IP Diagnosis Guide

Version 1  Release 11
Eleventh Edition (September 2009)

This edition applies to Version 1 Release 11 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You may send your comments to the following address.
International Business Machines Corporation
Attn: z/OS Communications Server Information Development
Department AKCA, Building 501
P.O. Box 12195, 3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195

You can send us comments electronically by using one of the following methods:

Fax (USA and Canada):
1+919-254-1258
Send the fax to “Attn: z/OS Communications Server Information Development”

Internet e-mail:
comsvrcf@us.ibm.com

World Wide Web:

If you would like a reply, be sure to include your name, address, telephone number, or FAX number. Make sure to include the following in your comment or note:

- Title and order number of this document
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Steps for diagnosing problems with IP routing to a destination when using policy-based routing (PBR) ..... 34
Steps for verifying network interface operation .................................................. 36
Steps for verifying network access ...................................................................... 36
Tools for diagnosing network connectivity problems ............................................ 37
Using the Ping command .................................................................................... 37
Using the Netstat command .............................................................................. 41
Using the Traceroute command ......................................................................... 43
Using SNMP remote Ping command .................................................................. 43
Documentation for the IBM Support Center .......................................................... 43

Part 2. Traces and control blocks ................................................................. 45

Chapter 5. TCP/IP services traces and IPCS support ...................................... 47
Component trace ................................................................................................. 47
  Modifying options with the TRACE CT command ........................................... 48
IKE daemon trace (SYSTCPIK) ........................................................................ 60
Event trace (SYSTCPIP) for TCP/IP stacks and Telnet ..................................... 60
  Specifying trace options .................................................................................. 61
  Additional filters for SYSTCPIP ...................................................................... 70
  Formatting event trace records for TCP/IP stacks and Telnet ....................... 71
  Socket API traces ........................................................................................... 74
Packet trace (SYSTCPDA) for TCP/IP stacks ..................................................... 94
  The trace process ........................................................................................... 94
  Supported devices .......................................................................................... 95
  Starting packet trace ..................................................................................... 95
  Modifying options with VARY ....................................................................... 96
  Formatting packet traces using IPCS .............................................................. 97
Data trace (SYSTCPDA) for TCP/IP stacks ......................................................... 148
  Starting data trace ......................................................................................... 148
  Displaying data traces ................................................................................... 149
  Formatting data traces using IPCS ................................................................. 149
Intrusion Detection Services trace (SYSTCPIS) .............................................. 149
  Restrictions .................................................................................................. 150
  CTRACE keywords on SYSTCPIS ............................................................... 150
  SYSTCPIS OPTIONS ................................................................................... 151
  OPTIONS Keywords ..................................................................................... 155
  IDS reports ................................................................................................... 167
  OSAENTA trace (SYSTCPOT) ....................................................................... 184
    The OSAENTA trace process .................................................................... 184
    Starting OSAENTA trace ........................................................................ 185
Network security services (NSS) server trace (SYSTCPNS) ............................. 188
Defense Manager daemon (DMD) trace (SYSTCPDM) ..................................... 188
OMPROUTE trace (SYSTCPRT) ........................................................................ 188
RESOLVER trace (SYSTCPRE) ........................................................................ 188
Configuration profile trace .............................................................................. 188

Chapter 6. IPCS subcommands for TCP/IP .................................................. 191
Types of subcommands ..................................................................................... 191
TCPIPCS command ........................................................................................... 194
  Syntax ........................................................................................................... 194
  Parameters .................................................................................................. 195
  Symbols defined ........................................................................................... 197
TCPIPCS subcommands ..................................................................................... 197
  TCPIPCS API ............................................................................................. 197
  TCPIPCS CONFIG ...................................................................................... 199
  TCPIPCS CONNECTION .......................................................................... 200
  TCPIPCS COUNTERS .............................................................................. 202
  TCPIPCS DUA .......................................................................................... 204
  TCPIPCS DUCB ........................................................................................ 206
  TCPIPCS FRCA ......................................................................................... 210
Part 3. Diagnosing z/OS Communications Server components ............ 313

Chapter 7. Diagnosing problems with the z/OS Load Balancing Advisor ...... 315
Diagnostic data ........................................ 315
Diagnosing Advisor and Agent problems .................................. 316
Abends .................................................................. 316
  Workload not distributed to a particular application ................. 316
  Workload not distributed as expected ................................ 317
  Advisor or Agent appears to be hung ................................ 318
  Group names in displays are indecipherable ......................... 319
  Load balancer connection terminates unexpectedly ................. 319
  Agent-Advisor connection terminates unexpectedly ................. 319
  Automatic restart manager (ARM) registration failure .............. 319
Debug settings and corresponding syslogd priority levels .............. 320

Chapter 8. Diagnosing problems with the automated domain name registration
application (ADNR) ............................................ 323
Diagnostic data ............................................. 323
Diagnosing ADNR problems ...................................... 325
Abends .................................................................. 325
  ADNR fails to initialize ........................................ 325
  ADNR not communicating with the Global Workload Manager .... 326
  Automatic restart manager (ARM) registration failure .............. 327
  ADNR not updating zones in a DNS server ......................... 327
  DNS name servers managed by ADNR contain incorrect or outdated data 328
Diagnosing unresponsive zones ...................................... 328
  ADNR appears to be hung ...................................... 328
  ADNR connection to the GWM terminates unexpectedly .......... 330
Debug settings .................................................. 331

Chapter 9. Diagnosing IKE daemon problems .................................... 333
Overview of diagnosing IKE daemon problems ......................... 333
Diagnosing IKE daemon problems .................................... 334
  Initialization problems ........................................ 334
  Problems establishing security associations ....................... 334
IKE daemon debug information ................................... 349
  Obtaining syslog debug information for the IKE daemon .......... 349
  Obtaining debug information using PagentSyslogLevel ........... 349
TCP/IP services component trace for the IKE daemon ............... 350
  Using CTRACE ................................................ 350
  Formatting IKE daemon trace records ........................... 354

Chapter 10. Diagnosing network security services (NSS) server problems .... 355
Overview of diagnosing NSS server problems ......................... 355
  Common NSS server initialization problems ....................... 355
  NSS client connection problems ................................ 356
  NSS XMLAppliance client API return codes and reason codes ... 357
Network security services server debug information ................. 362
  Obtaining syslog debug information for the network security service server 362
TCP/IP services component trace for the network security services (NSS) server 368
  Using CTRACE ................................................ 368
Steps for enabling the CTRACE at network security service (NSS) server startup ............................................. 369
Steps for enabling the CTRACE at network security services server startup .................................................. 371
Steps for disabling the CTRACE at network security services server startup ................................................ 371
Step for enabling the CTRACE after the network security services server has started ..................................... 371
Step for disabling the CTRACE after the network security services server has started .................................... 371
Step for displaying the CTRACE status .............................................................................................................. 371
Enabling CTRACE after network security services server initialization ....................................................... 372
Formatting network security services server trace records ..................................................................................... 372

Chapter 11. Diagnosing dynamic VIPA and sysplex problems ................................................................. 373
Overview of diagnosing sysplex distributor problems ..................................................................................... 373
Steps for diagnosing sysplex problems .......................................................................................................... 374
Steps for diagnosing problems using DVIPAs in source IP address selection for TCP connections problems ........ 385
Steps for diagnosing problems with the SYSPLEX-wide ephemeral port assignment for distributed DVIPAs 387
Diagnosing problems with the SYSPLEX-wide ephemeral port assignment for EXPLICITBINDPORTRANGE processing ............................................................ 389
Steps for determining an optimal range for the EXPLICITBINDPORTRANGE parameter 390
Steps for diagnosing problems with EXPLICITBINDPORTRANGE processing .............................................. 391
Diagnosing SYSPLEX-wide security association (SWSA) problems .......................................................... 393
Steps for diagnosing sysplex-wide security association (SWSA) problems .................................................. 393
Steps for diagnosing sysplex routing problems .......................................................................................... 397
Steps for diagnosing Tier 1 z/OS sysplex distribution problems .................................................................... 401
Steps for diagnosing Tier 1 non-z/OS sysplex distribution problems .......................................................... 401

Chapter 12. Diagnosing access control problems ......................................................................................... 407
Overview of access control support .................................................................................................................. 407
Diagnosing multilevel security consistency check messages (EZD1215-EZD1234) ........................................ 409
Steps for verifying the configuration .................................................................................................................. 409

Chapter 13. Diagnosing line print requester and daemon (LPR and LPD) problems ........................................ 411
Diagnosing LPR client and LPD server problems .............................................................................................. 411
Abends ................................................................................................................................................. 412
Steps for diagnosing timeouts, hangs, and waits ............................................................................................. 412
Incorrect output .................................................................................................................................. 413
LPR client traces ........................................................................................................................................ 415
Step for activating LPR client traces ........................................................................................................... 415
Step for creating client trace output ............................................................................................................. 415
LPD server traces ....................................................................................................................................... 421
Step for activating server traces .................................................................................................................... 421
Step for creating server trace output ............................................................................................................. 422

Chapter 14. Diagnosing File Transfer Protocol (FTP) problems ............................................................ 435
FTP server ........................................................................................................................................... 435
Structural overview .................................................................................................................................. 435
Definitions and setup ................................................................................................................................. 436
Error exit codes ..................................................................................................................................... 436
Name considerations for z/OS UNIX FTP .................................................................................................. 436
Translation and data conversion support .................................................................................................... 437
DB2 query support ................................................................................................................................ 439
JES support ........................................................................................................................................... 441
Logging FTP server activity .......................................................................................................................... 442
Common z/OS UNIX FTP problems ........................................................................................................... 443
Diagnosing FTP server problems with traces .............................................................................................. 456
Documenting server problems ...................................................................................................................... 464
FTP client .................................................................................................................................................. 465
Execution environments ............................................................................................................................. 465
Setup ....................................................................................................................................................... 465
Naming considerations ............................................................................................................................... 466
Directing the client to exit when an error occurs ............................................................................................. 466
Translation and data conversion support ..................................................................................................... 466
Chapter 19. Diagnosing SNALINK LU0 problems ................................. 547
Definitions ................................................................................. 547
Problem diagnosis ................................................................. 547
   Abends .............................................................................. 548
   Session hangs .................................................................... 548
   Session outages .................................................................. 550
Traces ....................................................................................... 551
   Using IP packet trace ....................................................... 551
   SNALINK LU0 DEBUG trace ............................................ 551

Chapter 20. Diagnosing SNALINK LU6.2 problems ............................. 555
Steps for setting up a SNALINK LU6.2 network ................................ 556
Common configuration mistakes .................................................. 557
Diagnosing problems ............................................................... 558
   Quick checklist for common problems ............................... 558
   Problems starting the SNALINK LU6.2 address space .......... 559
   DLC connection problems ............................................... 561
   Network connection establishment problems .................... 563
   Network connection loss problems .................................... 565
   Data loss problems .......................................................... 566
   Data corruption problems .................................................. 568
Documentation references for problem diagnosis ...................... 568
   Using NETSTAT .............................................................. 569
   Using the SNALINK LU6.2 subcommand ........................... 569
   Useful VTAM operations .................................................... 570
Traces ....................................................................................... 573
   Using SNALINK LU6.2 internal traces ............................... 573
   Using IP packet trace ....................................................... 573
   TCP/IP internal traces ...................................................... 576
   VTAM buffer traces .......................................................... 577
   Finding abend and sense code documentation ..................... 578
   Finding error message documentation ............................... 578

Chapter 21. Diagnosing name server problems ................................ 579
Identifying name server problems .............................................. 579
   Checking messages sent to the operator’s console ............... 579
   Checking the log messages ................................................. 580
   Tools for querying the name server .................................... 581
   Using the debug option with the name server ...................... 582
   Debugging with a resolver directive ................................... 583
   Using the remote name daemon control (rndc) program ........ 584
   Using name server signals ............................................... 584
   Statistics file for the Bind 9 name server ............................. 585
   Using the nsupdate command .......................................... 585
   Using component trace ..................................................... 585

Chapter 22. Diagnosing REXEC, REXECD, and RSH problems ........... 587
General information about REXEC and RSH ............................... 587
   Documentation for REXEC problem diagnosis .................... 587
   TSO console log ............................................................... 588
   Activating the REXEC debug trace .................................... 588
   REXEC trace example and explanation ............................... 588
   RSH trace example and explanation ................................... 588
General information about REXECD ......................................... 588
Chapter 23. Diagnosing z/OS UNIX REXEC, RSH, REXECD, and RSHD problems . . . 595
Setting up the inetd configuration file ........................................... 595
Diagnosing z/OS UNIX REXEC ......................................................... 596
Activating the z/OS UNIX REXEC debug trace ............................... 596
z/OS UNIX REXEC trace example and explanation ......................... 596
Diagnosing z/OS UNIX RSH ......................................................... 597
Step for activating the z/OS UNIX RSH debug trace ....................... 597
Step for invoking z/OS UNIX RSH trace ....................................... 597
Diagnosing z/OS UNIX REXECD ..................................................... 598
Activating the z/OS UNIX REXECD debug trace ............................ 598
z/OS UNIX REXECD trace example and explanation ....................... 598
Diagnosing z/OS UNIX RSHD ....................................................... 598
Step for activating the z/OS UNIX RSHD debug trace ..................... 598
z/OS UNIX RSHD trace example and explanation ........................... 599

Chapter 24. Diagnosing X Window System and Motif problems . . . . . . . . 601
Trace output when XWTRACE=2 ................................................... 601
Trace output when XWTRACE=2 ................................................... 602

Chapter 25. Diagnosing Simple Network Management Protocol (SNMP) problems . . . 605
Overview .................................................................................. 605
Management information base (MIB) ............................................. 605
PDUs ...................................................................................... 605
Functional components .............................................................. 606
Definitions ............................................................................. 607
z/OS UNIX snmp command .......................................................... 607
SNMP agent ........................................................................... 607
TCP/IP subagent .................................................................... 608
OMPROUTE subagent ............................................................... 608
Network SLAPM2 subagent ....................................................... 609
TN3270E Telnet subagent .......................................................... 609
SNMP socket call settings .......................................................... 609
Trap forwarder daemon .............................................................. 609
Diagnosing SNMP problems ....................................................... 609
Abends .................................................................................... 610
SNMP connection problems ....................................................... 610
Incorrect output ..................................................................... 617
No response from the SNMP agent ............................................. 623
Report received from SNMP agent ............................................. 624
0.0.0.0 address in traps from the SNMP agent ......................... 625
I/O error for SNMP PING ............................................................ 625
Traps not forwarded by trap forwarder daemon ......................... 626
Incorrect address in forwarded trap ............................................ 627
SNMP traces ............................................................................ 627
Starting manager traces ............................................................. 628
Starting SNMP agent traces ....................................................... 628
Starting TCP/IP subagent traces ................................................. 629
Starting OMPROUTE subagent traces ......................................... 630
Starting Network SLAPM2 subagent traces ............................... 630
Starting TN3270E Telnet subagent traces .................................. 630
Starting TRAPFWD traces .......................................................... 631
Trace examples and explanations ............................................... 631
### Chapter 26. Diagnosing Policy Agent problems

Overview: 653

- QoS policy 654
- QoS policy scope 654
- Configuration file import services 655
- Gathering diagnostic information 655
- Diagnosing Policy Agent problems
  - Initialization problems 657
  - Policy definition problems 659
  - Policy client connection problems 664
  - Policy client retrieval problems 667
  - Import requestor connection problems 669
  - Import requestor retrieval problems 673
  - LDAP object retrieval problems 673
  - LDAP object storage problems 675
  - Policy Agent and Sysplex distribution problems 677
  - Memory allocation/leakage problems 678

### Chapter 27. Diagnosing RSVP agent problems

Overview: 681

- Reservation types, styles, and objects 682
- Policies and RSVP processing 683
- Gathering diagnostic information 684
- Diagnosing RSVP agent problems
  - Initialization problems 684
  - Application problems 685
  - Policy problems 685
  - Example log file 686

### Chapter 28. Diagnosing intrusion detection problems

Overview: 695

- Diagnosing IDS policy problems 695
  - Step for determining which IDS policies are active in Policy Agent 695
  - Step for determining how your IDS policies have been mapped by the stack 695
- Diagnosing IDS output problems 696
  - Steps for determining why IDS syslogd output is missing 696
  - IDS console output 696
  - IDS packet trace output 697
  - Unusual conditions 697
- Diagnosing TRMD problems 699
- Documentation for the IBM Software Support Center 700

### Chapter 29. Diagnosing Application Transparent Transport Layer Security (AT-TLS)

- Common AT-TLS startup errors 701
- Steps for diagnosing AT-TLS problems 701
- AT-TLS traces 703
  - Sample AT-TLS trace 704
- AT-TLS return codes 706
- SIOCTTLSCTL ioctl return codes 712

### Chapter 30. Diagnosing Defense Manager daemon problems

Overview of diagnosing Defense Manager daemon problems 715

- Defense Manager daemon debug information 718
  - Abends during DMD processing 718
  - DMD error codes 719
- TCP/IP services component trace for the Defense Manager daemon 719
  - CTRACE options 719
- Enabling CTRACE at Defense Manager daemon startup 720
  - Steps for enabling the CTRACE at Defense Manager daemon startup 722
  - Steps for disabling the CTRACE at Defense Manager daemon startup 722
Step for enabling the CTRACE after the Defense Manager daemon has started ................................................................. 722
Step for disabling the CTRACE after the Defense Manager daemon has started ............................................................... 722
Displaying the CTRACE status ........................................................................................................................................ 722
Enabling CTRACE after Defense Manager daemon initialization ..................................................................................... 722
Formatting Defense Manager daemon trace records ........................................................................................................ 723

Chapter 31. Diagnosing IP security and defensive filter problems ................................................................................. 725
Overview of diagnosing IP security and defensive filter problems .................................................................................. 725
Steps for diagnosing IP security problems ......................................................................................................................... 726
Steps for diagnosing defensive filter problems .................................................................................................................. 727
Steps for diagnosing the cause for missing IP security or defensive filter syslogd output .................................................. 729
Steps for verifying IP security and defensive filter operation ............................................................................................. 732
  Steps for verifying manual IPSec protection ....................................................................................................................... 735
  Steps for verifying dynamic IPSec protection .................................................................................................................... 737
  Steps for verifying IP security policy or defensive filter enforcement ............................................................................. 740
Steps for verifying IPSec processing on zIIP ......................................................................................................................... 744
  Determining the Workload Manager service class associated with IPSec workload being processed on zIIP ............ 744
Tools for diagnosing IP security and defensive filter problems ......................................................................................... 744
  Using the ipsec command .................................................................................................................................................. 744
  Using the pasearch command ........................................................................................................................................... 747
  Using syslog messages ......................................................................................................................................................... 748

Chapter 32. Diagnosing OMPROUTE problems ............................................................................................................ 749
Overview .............................................................................................................................................................................. 749
Definitions ............................................................................................................................................................................. 752
Diagnosing OMPROUTE problems ...................................................................................................................................... 752
  Abends ............................................................................................................................................................................... 752
  OMPROUTE connection problems .................................................................................................................................. 752
  Routing failures ................................................................................................................................................................ 752
  Adjacency failures ............................................................................................................................................................. 754
OMPROUTE traces and debug information .......................................................................................................................... 759
  Starting OMPROUTE tracing and debugging from the z/OS UNIX System Services shell .............................................. 759
  Starting OMPROUTE tracing and debugging from an MVS cataloged procedure or AUTOLOG .................................. 760
  Starting OMPROUTE tracing and debugging using the MODIFY command ................................................................. 760
  Destination of OMPROUTE trace and debug output ......................................................................................................... 761
Sample OMPROUTE trace output ........................................................................................................................................ 762
TCP/IP services component trace for OMPROUTE .................................................................................................................. 772
  Specifying trace options ........................................................................................................................................................ 772
  Formatting OMPROUTE trace records .................................................................................................................................. 776
Commands to enable, disable, and display the status of the OMPROUTE CTRACE ............................................................. 776
  Steps for enabling the CTRACE at OMPROUTE startup ................................................................................................. 776
  Steps for disabling the CTRACE at OMPROUTE startup ............................................................................................... 776
  Steps for enabling the CTRACE after OMPROUTE has started .................................................................................... 776
  Steps for disabling the CTRACE after OMPROUTE has started ................................................................................... 777
  Step for displaying the CTRACE status ............................................................................................................................. 777

Chapter 33. Diagnosing NCPROUTE problems ............................................................................................................ 779
Definitions .............................................................................................................................................................................. 782
Diagnosing NCPROUTE problems ....................................................................................................................................... 783
  Abends ............................................................................................................................................................................... 783
  Connection problems ........................................................................................................................................................ 783
  Analyzing routing failures ................................................................................................................................................. 787
  Incorrect output ............................................................................................................................................................... 789
  Session outages ................................................................................................................................................................ 790
NCPROUTE traces .................................................................................................................................................................. 792
  Activating NCPROUTE global traces ............................................................................................................................... 792
  Activating NCPROUTE selective traces ............................................................................................................................ 792
  NCPROUTE trace example and explanation ..................................................................................................................... 793

Chapter 34. Diagnosing X.25 NPSI problems .................................................................................................................. 805
### Chapter 35. Diagnosing IMS problems.

- Steps for setting up the IMS TCP/IP services socket interface system... 817
- Common configuration mistakes... 819
- Quick checklist for common problems... 819
  - Component problems...
  - Connection problems...
  - Error message and return code problems...
  - Socket data protocol problems...
  - IMS transaction build problems...
  - IMS database problems...
- Documentation references for problem diagnosis... 833
  - Traces... 833
  - Using NETSTAT... 834
  - Where to find return code documentation... 835
  - Where to find error message documentation... 836

### Chapter 36. Diagnosing VMCF/TNF/IUCV problems

- Diagnosing restartable VMCF/TNF problems... 837
  - VMCF or TNF fail to initialize... 837
  - Abends 0D5 and 0D6... 837
  - Steps for diagnosing no response to commands... 837
  - VMCF or TNF does not stop... 838
- Diagnosing VMCF/IUCV problems with the TSO MVPXDISP command... 838

### Chapter 37. Diagnosing problems with IP CICS sockets

- Diagnostic data... 843
- Initialization problems... 844
  - Steps for diagnosing CICS socket interface not initialized... 844
  - Steps for diagnosing CICS listener not initialized... 844
  - No CICS sockets messages issued... 845
  - Steps for diagnosing TCP/IP clients unable to connect... 845
  - Steps for diagnosing child-server transactions not starting... 846
- CICS sockets application problems... 846
  - Steps for diagnosing hung CICS tasks... 846
  - Hung CICS region... 847
  - Errors on socket calls... 847
  - CICS shutdown hangs... 847
- CICS sockets control blocks... 847
  - Task interface element... 847
  - Global work area... 848
- CICS trace... 848
  - Steps for displaying the internal trace... 848
Chapter 38. Diagnosing problems with Express Logon ................................. 849
  Analyzing start problems with the DCAS ........................................ 850
  Analyzing client interface problems ............................................ 851

Chapter 39. Diagnosing resolver problems ....................................... 853
  Steps for resolving the hostname ............................................... 853
    Steps for resolving caching problems .................................... 855
    TRACE RESOLVER .................................................................. 857
      Specifying the Trace Resolver output location ......................... 859
      Interpreting the Trace Resolver output .................................. 861
    CTRACE — RESOLVER ...................................................... 876

Chapter 40. Diagnosing Simple Network Time Protocol (SNTP) problems .... 879
  Activating the SNTPD debug trace ............................................. 879
  Abends .................................................................................. 879
  Steps for stopping SNTPD ....................................................... 879
  Sample SNTPD debug output ................................................... 880

Chapter 41. Diagnosing Communications Server SMTP application problems .. 881
  Gathering diagnostic information ............................................. 881
    Steps for gathering log information ....................................... 881
  Resolving initialization or logging problems .............................. 882
  Resolving SMTPNOTE CLIST problems ................................... 883
  Diagnosing and resolving Resolver problems ............................. 883
  Resolving problems from the JES spool data set ......................... 885
  Resolving mail problems ....................................................... 888
    Target server problems ...................................................... 888
    Mail problems .................................................................. 889
  Resolving MODIFY command problems ................................... 894
  Diagnosing checkpoint problems ............................................ 895
  Monitoring resources used ..................................................... 896

Chapter 42. Diagnosing storage abends and storage growth ..................... 899
  Storage definitions ............................................................... 899
  Monitoring storage utilization ................................................ 900
  Limiting TCP/IP common and private storage utilization ............. 901
  Limiting CSM storage utilization ............................................. 901
  Storage messages ................................................................. 902
    Sysplex Problem Detection and Recovery (SPDR) storage messages 903
  Abends .................................................................................. 903
  Problem determination .......................................................... 904
    Steps to take when reviewing a storage problem ....................... 904
    Collecting documentation to submit to the IBM Support Center .... 905

Appendix A. First Failure Support Technology (FFST) ............................. 907
  FFST probe index ................................................................. 907
  FFST probe information ......................................................... 907
  FFST probe naming conventions .............................................. 908
  FFST probe descriptions ....................................................... 908

Appendix B. Overview of internetworking .......................................... 917
  Maximum transmission unit (MTU) ............................................. 918
  Fiber Distributed Data Interface (FDDI) .................................... 919
  Token-Ring IEEE 802.5 ......................................................... 920
  IEEE 802.3 .................................................................... 921
  Ethernet — DIX V2 ................................................................ 921
  Subnetwork Access Protocol (SNAP) ......................................... 922
  IP routing ........................................................................ 922
  Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) 923
Figures

1. Overview of the diagnosis procedure ............................................. 4
2. Example of output from the IPCS SYSTRACE command ............................ 27
3. Overview of verifying server operation ........................................... 31
4. Overview of verifying IP routing to a destination ................................ 33
5. Overview of verifying network interface operation ............................... 36
6. Overview of verifying network access .............................................. 37
7. Example of DUMP command for TCP/IP stack .................................... 52
8. Example of DUMP command for OMPROUTE ....................................... 52
9. Example of DUMP command for RESOLVER ..................................... 53
10. Example of DUMP command for TELNET ......................................... 53
11. IPCS CTRACE ................................................................................. 56
12. SYS1.PARMLIB member CTIEZB00 .................................................. 62
13. Start of component trace full format ............................................... 73
14. Component trace full format showing character interpretation of fields ... 74
15. TCP/IP networking API relationship on z/OS .................................... 75
16. Data trace record ............................................................................. 93
17. SOCKAPI trace record ..................................................................... 94
18. Control and data flow in the IP packet tracing facility ......................... 95
19. Example of a SUMMARY report ....................................................... 122
20. Format report example .................................................................... 129
21. Data trace: Single entry .................................................................. 149
22. Data trace with multiple entries ....................................................... 149
23. SYS1.PARMLIB member CTIIIDS00 ................................................ 150
24. Control and data flow in the OSAENTA tracing facility ....................... 185
25. TCPIPCS IPSEC subcommand sample output ..................................... 221
26. IPCS primary option menu ............................................................... 310
27. IPCS subcommand entry panel with a TCP/IP IPCS subcommand entered 311
28. Main menu for TCP/IP IPCS subcommands ....................................... 312
29. NAT keep alive message ................................................................. 348
30. SYS1.PARMLIB member CTIIKE00 ................................................ 352
31. Netstat VIPADCFG/-f example ......................................................... 376
32. Netstat CONFIG/-f example ............................................................. 377
33. D WLM,SYSTEMS example ............................................................. 378
34. Netstat VIPADYN/-v example ............................................................ 378
35. Netstat VIPADyn/-v example ............................................................. 379
36. Sysplex VIPADyn example ............................................................... 380
37. Netstat VDPT/-O example ............................................................... 383
38. d tcpip,tcps,net,vCRT,detail example .............................................. 385
39. Netstat ALLConn/-a example with IPAddr/-I filter value of 201.2.1.12 ..... 385
40. Netstat CONFIG/-f example ............................................................. 386
41. Netstat SRCIP/-f example ............................................................... 386
42. Diagnosing SYSPLEXPORTS problems ........................................... 387
43. VTAM NET,STATS example ............................................................. 389
44. VTAM NET,STATS,LIST=0 example ................................................ 390
45. ipsec -f example ............................................................................ 393
46. netstat,config example ................................................................... 394
47. ipsec -y example ............................................................................ 395
48. ipsec -y display -s example ............................................................. 396
49. Netstat VIPADyn example ............................................................... 398
50. Netstat VCRH/-V detail example ....................................................... 399
51. Example of LPR trace output ........................................................... 416
52. Example of LPR trace with filter x option .......................................... 418
53. Example of LPR output with unknown printer .................................... 419
54. Example of LPR trace with JNUM, LANDSCAPE, and TRACE options 420
55. Example of LPR trace with XLATE option ......................................... 421
<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>Example of LPD trace specified with the DEBUG option</td>
</tr>
<tr>
<td>57</td>
<td>Example of an LPD server trace of a failing job</td>
</tr>
<tr>
<td>58</td>
<td>Example of an LPD server trace for a remote print request</td>
</tr>
<tr>
<td>59</td>
<td>Trace between the Telnet client, parent, and child</td>
</tr>
<tr>
<td>60</td>
<td>z/OS UNIX Telnet trace using -t -D all</td>
</tr>
<tr>
<td>61</td>
<td>Telnet client trace</td>
</tr>
<tr>
<td>62</td>
<td>SMTP environment</td>
</tr>
<tr>
<td>63</td>
<td>Example of RESOLVER trace output</td>
</tr>
<tr>
<td>64</td>
<td>Example of a SNALINK LU0 DEBUG trace</td>
</tr>
<tr>
<td>65</td>
<td>Components of a SNALINK LU6.2 connection on MVS</td>
</tr>
<tr>
<td>66</td>
<td>Sample MVS System Console Messages on SNALINK LU6.2 Address Space Startup</td>
</tr>
<tr>
<td>67</td>
<td>NETSTAT DEVLINKS output example</td>
</tr>
<tr>
<td>68</td>
<td>LIST MODIFY subcommand output example</td>
</tr>
<tr>
<td>69</td>
<td>DISPLAY subcommand output example for connectable LU</td>
</tr>
<tr>
<td>70</td>
<td>DISPLAY subcommand output example for active LU</td>
</tr>
<tr>
<td>71</td>
<td>SNALINK LU6.2 internal trace output</td>
</tr>
<tr>
<td>72</td>
<td>A CTRACE formatted packet trace record</td>
</tr>
<tr>
<td>73</td>
<td>Remote execution protocol principle</td>
</tr>
<tr>
<td>74</td>
<td>Example of an REXEC trace</td>
</tr>
<tr>
<td>75</td>
<td>Example of an RSH trace</td>
</tr>
<tr>
<td>76</td>
<td>Example of an REXECD trace of a client using a SEND command</td>
</tr>
<tr>
<td>77</td>
<td>Example of a trace of an RSH client using a SEND command</td>
</tr>
<tr>
<td>78</td>
<td>Example trace to the JES spool file of the server</td>
</tr>
<tr>
<td>79</td>
<td>Adding applications to /etc/inetd.conf</td>
</tr>
<tr>
<td>80</td>
<td>Setting traces in /etc/inetd.conf</td>
</tr>
<tr>
<td>81</td>
<td>Example of X Application trace output when XWTRACE=2</td>
</tr>
<tr>
<td>82</td>
<td>Example of X Application trace output when XWTRACELC=2</td>
</tr>
<tr>
<td>83</td>
<td>SNMP agent response trace</td>
</tr>
<tr>
<td>84</td>
<td>SNMP agent trace of unsuccessful initialization</td>
</tr>
<tr>
<td>85</td>
<td>SNMP messages and agent trace for nonmatching key</td>
</tr>
<tr>
<td>86</td>
<td>SNMP messages and agent trace when data not in defined view</td>
</tr>
<tr>
<td>87</td>
<td>SNMP subagent trace</td>
</tr>
<tr>
<td>88</td>
<td>SNMP query engine traces</td>
</tr>
<tr>
<td>89</td>
<td>SNMP IUCV communication traces</td>
</tr>
<tr>
<td>90</td>
<td>TRAPFWD trace</td>
</tr>
<tr>
<td>91</td>
<td>RSVP Agent processing log</td>
</tr>
<tr>
<td>92</td>
<td>Example trace of a generic server processing</td>
</tr>
<tr>
<td>93</td>
<td>SYS1.PARMLIB member CTIDMD00</td>
</tr>
<tr>
<td>94</td>
<td>Overview of verifying IP security operation</td>
</tr>
<tr>
<td>95</td>
<td>Overview of verifying manual IPSec protection</td>
</tr>
<tr>
<td>96</td>
<td>Overview of verifying dynamic IPSec protection</td>
</tr>
<tr>
<td>97</td>
<td>Overview of verifying IP security policy enforcement</td>
</tr>
<tr>
<td>98</td>
<td>Sample OMPROUTE Trace Output</td>
</tr>
<tr>
<td>99</td>
<td>Sample IPv6 OMPROUTE Trace Output</td>
</tr>
<tr>
<td>100</td>
<td>SYS1.PARMLIB member CTIORA00</td>
</tr>
<tr>
<td>101</td>
<td>NCPROUTE environment</td>
</tr>
<tr>
<td>102</td>
<td>NCPROUTE trace</td>
</tr>
<tr>
<td>103</td>
<td>X.25 NPSI environment</td>
</tr>
<tr>
<td>104</td>
<td>Components of the IMS TCP/IP services socket interface system</td>
</tr>
<tr>
<td>105</td>
<td>MVPXDISP sample output using the userid parameter</td>
</tr>
<tr>
<td>106</td>
<td>MVPXDISP sample output using the ISAQ parameter</td>
</tr>
<tr>
<td>107</td>
<td>Error report example</td>
</tr>
<tr>
<td>108</td>
<td>Example of undeliverable mail notification</td>
</tr>
<tr>
<td>109</td>
<td>Routers and bridges within an internet</td>
</tr>
<tr>
<td>110</td>
<td>Relationship of MTU to frame size</td>
</tr>
<tr>
<td>111</td>
<td>Format of an IEEE 802.5 token-ring frame</td>
</tr>
<tr>
<td>112</td>
<td>Format of an IEEE 802.3 frame</td>
</tr>
<tr>
<td>113</td>
<td>Format of an Ethernet V2 frame</td>
</tr>
<tr>
<td>114</td>
<td>SNAP header</td>
</tr>
<tr>
<td>115</td>
<td>Classes of IPv4 addresses</td>
</tr>
<tr>
<td>116</td>
<td>Determining the class of an IPv4 address</td>
</tr>
</tbody>
</table>
117. Routing and bridging .................................. 925
118. General IP routing algorithm .................................. 927
119. Subnetting scheme .................................. 928
120. Routing algorithm with subnets .................................. 930
121. Example of resolving a subnet route .................................. 930
122. Main mode exchange .................................. 934
123. Aggressive mode exchange .................................. 936
124. Interpreting phase 1 SA states in Main mode .................................. 937
125. Interpreting phase 1 SA states in Aggressive mode .................................. 938
126. Quick mode exchange messages .................................. 939
127. Quick mode exchange with commit-bit support .................................. 940
128. Quick (phase 2) SA states without commit-bit support .................................. 942
129. Quick (phase 2) SA states with commit-bit support .................................. 943
130. IKE packet with and without the non-ESP marker .................................. 947
## Tables

1. Selecting a dump ................................ 8
2. Selecting a trace ................................ 8
3. Selecting a service aid .............................. 12
4. Description of dumps ................................ 13
5. Description of traces ................................ 14
6. Description of service aids ............................ 18
7. TCP/IP component name and release level ............. 23
8. Types of abends ................................ 25
9. Diagnostic commands ................................ 30
10. Diagnosis of a timeout .............................. 41
11. Trace options and groups ............................. 64
12. Data types .................................. 70
13. IP address and port filtering effect on different types of socket API calls ..................... 79
14. Flags that apply to IP or SNA packets ................. 109
15. TCP/IP IPCS commands .............................. 191
16. Available debug levels and associated syslogd priority levels .................................. 320
17. Establishing security associations problems ............ 335
18. Common problems when IKED, running as an NSS client, is unable to obtain services from the NSS server 343
19. IKE daemon trace options ............................ 350
20. Common NSS server initialization problems .......... 355
22. Common NSS client request failures .................... 357
23. NSS XMLAppliance client API return codes and reason codes .......................... 358
24. NSS IPSec client API return codes and reason codes .... 363
25. NSS server trace options ............................. 368
26. SQL problems generating 55x replies (FTP Server) .... 440
27. Other SQL problems (FTP Server) ..................... 441
28. SQL problems generating 55x replies (FTP Client) ... 472
29. Other SQL problems (FTP Client) ................. 473
30. Debug trace options ................................ 490
31. Telnet login problems .............................. 506
32. Incorrect output types for Telnet ....................... 510
33. Telnet commands from RFC 854 ....................... 523
34. Telnet command options from RFC 1060 ............. 523
35. Format of Record 1 of an SMTP ADDRBLOK data set .................................. 531
36. Format of Record 2 (for an unresolved from record) of an SMTP ADDRBLOK data set ........ 532
37. Format of Record 2 (for a resolved from record) of an SMTP ADDRBLOK data set ........ 532
38. Format of Record 3 (for an unresolved from record) of an SMTP ADDRBLOK data set ........ 533
39. Debugging switches by category ....................... 537
40. qf File code letters ................................ 540
41. Format of a SNALINK trace table entry ............... 552
42. Common SNALINK LU6.2 address space startup problems .................................. 559
43. Common DLC connection problems ..................... 562
44. SNALINK LU6.2 address space network establishment problems .................................. 564
45. Outage problem ................................ 566
46. Common SNALINK LU6.2 data loss problems ........ 567
47. Logging categories ................................ 581
48. BIND 9 debug information ............................ 582
49. `rndc` commands ................................ 584
50. BIND 9 name server signals ......................... 584
51. Configuration files and security types .............. 608
52. Common Policy Agent initialization problems ....... 658
53. Policy definition problems ......................... 659
54. Common policy client connection problems .......... 664
55. Common policy client retrieval problems .............. 668

© Copyright IBM Corp. 1994, 2009
56. Common import requestor connection problems ..................... 670
57. Common import requestor retrieval problems ...................... 673
58. LDAP object retrieval problems ................................... 674
59. LDAP object storage problems .................................... 676
60. Common RSVP initialization problems .......................... 684
61. RSVP application problems ...................................... 685
62. RSVP policy problems ............................................ 686
63. AT-TLS trace levels ............................................... 703
64. Common System SSL return codes ................................ 706
65. AT-TLS return codes ............................................... 708
66. SIOCTTLSCTL error codes ........................................ 712
67. Common defense manager daemon (DMD) problems ............... 716
68. Defense manager daemon (DMD) trace options .................. 719
69. GUI-level object mapping ......................................... 725
70. IPSec messages logged by TRMD .................................. 729
71. Defensive filter messages logged by TRMD ....................... 729
72. ipsec -f display command options ................................ 745
73. ipsec -F display command options ................................ 745
74. ipsec -F update or delete command options ..................... 746
75. pasearch commands ................................................. 747
76. OMPROUTE command terms ........................................ 753
77. Neighbor event code associated with adjacency failures ....... 756
78. Neighbor state associated with adjacency failures ............. 756
79. OMPTRACE options .................................................. 775
80. Types of RIP messages ............................................. 781
81. MIB variables registered for use by NCPROUTE ................. 782
82. NCPROUTE connection problems .................................. 784
83. NCPROUTE command terms ....................................... 787
84. NCPROUTE routing failures ....................................... 788
85. NCPROUTE incorrect output ....................................... 790
86. Symptoms of session outages ..................................... 791
87. Component problems ............................................... 820
88. Connection problems ............................................... 821
89. Error message and return code problems ......................... 825
90. Socket data protocol problems ................................... 826
91. IMS transaction build problems .................................. 830
92. IMS database problems ............................................ 830
93. Common initialization and logging problems .............. 882
94. Problems using SMTPNOTE CLIST to create mail messages on the JES spool data set 883
95. Diagnosing and resolving Resolver problems ............................. 883
96. JES spool data set problems ..................................... 885
97. Common messages indicating target server status .............. 888
98. Common mail text problems ..................................... 893
99. Dead letters problems ............................................. 894
100. Common MODIFY command ....................................... 894
101. Common messages indicating checkpoint status .................. 895
102. Common messages used for monitoring resources ............... 896
103. Commands for various types of dumps .......................... 905
104. FFST probes ....................................................... 907
105. FFST naming conventions ......................................... 908
106. IOCTL enablement probes ....................................... 908
107. Infrastructure services probes .................................. 908
108. FFST probes for Pascal API ...................................... 909
109. PFS IOCTL probes ................................................ 913
110. Telnet transform probes ......................................... 914
111. FFST probes for Telnet SRV ..................................... 914
112. Configuration services probes .................................. 914
113. TCP/IP Base probes .............................................. 914
114. Transmission Control Protocol probes .......................... 915
115. Update Datagram Protocol Layer probes ......................... 915
116. Streams probes .................................................... 915
117. Raw IP Layer probes ................................................. 916
118. FFST probes for Internet Protocol .......................... 916
119. XCF probes ......................................................... 916
120. Relationship between RC field and maximum I-field value ............................................. 921
121. Vendor ID ............................................................. 945
About this document

This document tells you how to diagnose and report problems occurring in the IBM® z/OS® TCP/IP. Additional information is provided for diagnosing problems with selected applications that are part of z/OS Communications Server.

The information in this document includes descriptions of support for both IPv4 and IPv6 networking protocols. Unless explicitly noted, descriptions of IP protocol support concern IPv4. IPv6 support is qualified within the text.

Use this document to perform the following tasks:
- Diagnose and solve problems in a z/OS Communications Server installation.
- Describe problems to the IBM Software Support Center and document the problems appropriately.

This document refers to Communications Server data sets by their default SMP/E distribution library name. Your installation might, however, have different names for these data sets where allowed by SMP/E, your installation personnel, or administration staff. For instance, this document refers to samples in SEZAINST library as simply in SEZAINST. Your installation might choose a data set name of SYS1.SEZAINST, CS390.SEZAINST or other high level qualifiers for the data set name.

Who should read this document

System programmers can use this document to diagnose problems with TCP/IP or to diagnose problems with z/OS Communications Server components.

How this document is organized

The z/OS Communications Server: IP Diagnosis Guide is divided into the following parts:

Part 1, “General diagnosis information” describes how to diagnose a problem suspected to be caused by z/OS Communications Server, select diagnostic tools, and apply diagnostic techniques.

Part 2, “Traces and control blocks” describes selected procedures for TCP/IP Services component trace, packet trace, Socket API trace, and the subcommands (installation, entering, and execution).

Part 3, “Diagnosing z/OS Communications Server components” gives detailed diagnostic information for z/OS Communications Server components.

The appendixes provide additional information for this document.

How to use this document

To use this document, you should be familiar with z/OS TCP/IP Services and the TCP/IP suite of protocols.
This book contains various traces and code examples. In many cases, these examples contain non-release specific information; they are included for illustrative purposes. Actual examples and traces depend on your environment.

**Determining whether a publication is current**

As needed, IBM updates its publications with new and changed information. For a given publication, updates to the hardcopy and associated BookManager® softcopy are usually available at the same time. Sometimes, however, the updates to hardcopy and softcopy are available at different times. The following information describes how to determine if you are looking at the most current copy of a publication:

- At the end of a publication’s order number there is a dash followed by two digits, often referred to as the dash level. A publication with a higher dash level is more current than one with a lower dash level. For example, in the publication order number GC28-1747-07, the dash level 07 means that the publication is more current than previous levels, such as 05 or 04.
- If a hardcopy publication and a softcopy publication have the same dash level, it is possible that the softcopy publication is more current than the hardcopy publication. Check the dates shown in the Summary of Changes. The softcopy publication might have a more recently dated Summary of Changes than the hardcopy publication.
- To compare softcopy publications, you can check the last two characters of the publication’s file name (also called the book name). The higher the number, the more recent the publication. Also, next to the publication titles in the CD-ROM booklet and the readme files, there is an asterisk (*) that indicates whether a publication is new or changed.

**How to contact IBM service**


Most problems can be resolved at this Web site, where you can submit questions and problem reports electronically, as well as access a variety of diagnosis information.

For telephone assistance in problem diagnosis and resolution (in the United States or Puerto Rico), call the IBM Software Support Center anytime (1-800-IBM-SERV). You will receive a return call within 8 business hours (Monday – Friday, 8:00 a.m. – 5:00 p.m., local customer time).

Outside the United States or Puerto Rico, contact your local IBM representative or your authorized IBM supplier.

If you would like to provide feedback on this publication, see “Communicating Your Comments to IBM” on page 999.

**Conventions and terminology that are used in this document**

Commands in this book that can be used in both TSO and z/OS UNIX® environments use the following conventions:

- When describing how to use the command in a TSO environment, the command is presented in uppercase (for example, NETSTAT).
• When describing how to use the command in a z/OS UNIX environment, the command is presented in bold lowercase (for example, netstat).
• When referring to the command in a general way in text, the command is presented with an initial capital letter (for example, Netstat).

All of the exit routines described in this document are installation-wide exit routines. You will see the installation-wide exit routines also called installation-wide exits, exit routines, and exits throughout this document.

The TPF logon manager, although shipped with VTAM®, is an application program. Therefore, the logon manager is documented separately from VTAM.

Samples used in this book might not be updated for each release. Evaluate a sample carefully before applying it to your system.

For definitions of the terms and abbreviations that are used in this document, you can view the latest IBM terminology at the IBM Terminology Web site.

Clarification of notes

Information traditionally qualified as Notes is further qualified as follows:

Note Supplemental detail
Tip Offers shortcuts or alternative ways of performing an action; a hint
Guideline Customary way to perform a procedure
Rule Something you must do; limitations on your actions
Restriction Indicates certain conditions are not supported; limitations on a product or facility
Requirement Dependencies, prerequisites
Result Indicates the outcome

How to read a syntax diagram

This syntax information applies to all commands and statements that do not have their own syntax described elsewhere.

The syntax diagram shows you how to specify a command so that the operating system can correctly interpret what you type. Read the syntax diagram from left to right and from top to bottom, following the horizontal line (the main path).

Symbols and punctuation

The following symbols are used in syntax diagrams:

Symbol
Description
→ Marks the beginning of the command syntax.
▷ Indicates that the command syntax is continued.
Marks the beginning and end of a fragment or part of the command syntax.

Marks the end of the command syntax.

You must include all punctuation such as colons, semicolons, commas, quotation marks, and minus signs that are shown in the syntax diagram.

**Commands**

Commands that can be used in both TSO and z/OS UNIX environments use the following conventions in syntax diagrams:

- When describing how to use the command in a TSO environment, the command is presented in uppercase (for example, NETSTAT).
- When describing how to use the command in a z/OS UNIX environment, the command is presented in bold lowercase (for example, netstat).

**Parameters**

The following types of parameters are used in syntax diagrams.

**Required**
Required parameters are displayed on the main path.

**Optional**
Optional parameters are displayed below the main path.

**Default**
Default parameters are displayed above the main path.

Parameters are classified as keywords or variables. For the TSO and MVS™ console commands, the keywords are not case sensitive. You can code them in uppercase or lowercase. If the keyword appears in the syntax diagram in both uppercase and lowercase, the uppercase portion is the abbreviation for the keyword (for example, OPERand).

For the z/OS UNIX commands, the keywords must be entered in the case indicated in the syntax diagram.

Variables are italicized, appear in lowercase letters, and represent names or values you supply. For example, a data set is a variable.

**Syntax examples**

In the following example, the USER command is a keyword. The required variable parameter is user_id, and the optional variable parameter is password. Replace the variable parameters with your own values.

```
USER user_id [password]
```

**Longer than one line**

If a diagram is longer than one line, the first line ends with a single arrowhead and the second line begins with a single arrowhead.
Required operands

Required operands and values appear on the main path line. You must code required operands and values.

Optional values

Optional operands and values appear below the main path line. You do not have to code optional operands and values.

Selecting more than one operand

An arrow returning to the left above a group of operands or values means more than one can be selected, or a single one can be repeated.

Nonalphanumeric characters

If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code OPERAND=(001,0.001).

Blank spaces in syntax diagrams

If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code OPERAND=(001 FIXED).
Default operands

Default operands and values appear above the main path line. TCP/IP uses the default if you omit the operand entirely.

Variables

A word in all lowercase italics is a variable. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.

Syntax fragments

Some diagrams contain syntax fragments, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram.

Prerequisite and related information

z/OS Communications Server function is described in the z/OS Communications Server library. Descriptions of those documents are listed in “Bibliography” on page 983, in the back of this document.

Required information

Before using this product, you should be familiar with TCP/IP, VTAM, MVS, and UNIX System Services.

Softcopy information

Softcopy publications are available in the following collections.

<table>
<thead>
<tr>
<th>Titles</th>
<th>Order Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS V1R11 Collection</td>
<td>SK3T-4269</td>
<td>This is the CD collection shipped with the z/OS product. It includes the libraries for z/OS V1R11, in both BookManager and PDF formats.</td>
</tr>
</tbody>
</table>
### Other documents

For information about z/OS products, refer to *z/OS Information Roadmap* (SA22-7500). The Roadmap describes what level of documents are supplied with each release of z/OS Communications Server, as well as describing each z/OS publication.

Relevant RFCs are listed in an appendix of the IP documents. Architectural specifications for the SNA protocol are listed in an appendix of the SNA documents.

The following table lists documents that might be helpful to readers.

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNA Formats</td>
<td>GA27-3136</td>
</tr>
<tr>
<td>TCP/IP Tutorial and Technical Overview</td>
<td>GG24-3376</td>
</tr>
<tr>
<td>Understanding LDAP</td>
<td>SG24-4986</td>
</tr>
<tr>
<td>z/OS Cryptographic Services System SSL Programming</td>
<td>SC24-5901</td>
</tr>
<tr>
<td>z/OS Integrated Security Services LDAP Server Administration and Use</td>
<td>SC24-5923</td>
</tr>
<tr>
<td>z/OS JES2 Initialization and Tuning Guide</td>
<td>SA22-7532</td>
</tr>
<tr>
<td>z/OS Problem Management</td>
<td>G325-2564</td>
</tr>
<tr>
<td>z/OS MVS Diagnosis: Reference</td>
<td>GA22-7588</td>
</tr>
</tbody>
</table>
Redbooks

The following Redbooks might help you as you implement z/OS Communications Server.

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communications Server for z/OS V1R10 TCP/IP Implementation, Volume 1: Base Functions, Connectivity, and Routing</td>
<td>SG24-7696</td>
</tr>
<tr>
<td>Communications Server for z/OS V1R10 TCP/IP Implementation, Volume 2: Standard Applications</td>
<td>SG24-7697</td>
</tr>
<tr>
<td>Communications Server for z/OS V1R10 TCP/IP Implementation, Volume 3: High Availability, Scalability, and Performance</td>
<td>SG24-7698</td>
</tr>
<tr>
<td>Communications Server for z/OS V1R10 TCP/IP Implementation, Volume 4: Security and Policy-Based Networking</td>
<td>SG24-7699</td>
</tr>
<tr>
<td>IBM Communication Controller Migration Guide</td>
<td>SG24-6298</td>
</tr>
<tr>
<td>IP Network Design Guide</td>
<td>SG24-2580</td>
</tr>
<tr>
<td>Managing OS/390 TCP/IP with SNMP</td>
<td>SG24-5866</td>
</tr>
<tr>
<td>Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender</td>
<td>SG24-5957</td>
</tr>
<tr>
<td>SecureWay™ Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements</td>
<td>SG24-5631</td>
</tr>
<tr>
<td>SNA and TCP/IP Integration</td>
<td>SG24-5291</td>
</tr>
<tr>
<td>TCP/IP in a Sysplex</td>
<td>SG24-5235</td>
</tr>
<tr>
<td>TCP/IP Tutorial and Technical Overview</td>
<td>GG24-3376</td>
</tr>
<tr>
<td>Threadsafe Considerations for CICS</td>
<td>SG24-6351</td>
</tr>
</tbody>
</table>

Where to find related information on the Internet

z/OS

This site provides information about z/OS Communications Server release availability, migration information, downloads, and links to information about z/OS technology

http://www.ibm.com/systems/z/os/zos/

z/OS Internet Library

Use this site to view and download z/OS Communications Server documentation
IBM Communications Server product
The primary home page for information about z/OS Communications Server

IBM Communications Server product support
Use this site to submit and track problems and search the z/OS Communications Server knowledge base for Technotes, FAQs, white papers, and other z/OS Communications Server information

IBM Communications Server performance information
This site contains links to the most recent Communications Server performance reports.
http://www.ibm.com/support/docview.wss?uid=swg27005524

IBM Systems Center publications
Use this site to view and order Redbooks, Redpapers, and Technotes
http://www.redbooks.ibm.com/

IBM Systems Center flashes
Search the Technical Sales Library for Techdocs (including Flashes, presentations, Technotes, FAQs, white papers, Customer Support Plans, and Skills Transfer information)
http://www.ibm.com/support/techdocs/atsmastr.nsf

RFCs
Search for and view Request for Comments documents in this section of the Internet Engineering Task Force Web site, with links to the RFC repository and the IETF Working Groups Web page
http://www.ietf.org/rfc.html

Internet drafts
View Internet-Drafts, which are working documents of the Internet Engineering Task Force (IETF) and other groups, in this section of the Internet Engineering Task Force Web site
http://www.ietf.org/ID.html

Information about Web addresses can also be found in information APAR II11334.

Note: Any pointers in this publication to Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

DNS Web sites
For more information about DNS, see the following USENET news groups and mailing addresses:

USENET news groups
comp.protocols.dns.bind
BIND mailing lists
http://www.isc.org/ml-archives/

BIND Users

- Subscribe by sending mail to bind-users-request@isc.org.
- Submit questions or answers to this forum by sending mail to bind-users@isc.org.

BIND 9 Users (This list might not be maintained indefinitely.)

- Subscribe by sending mail to bind9-users-request@isc.org.
- Submit questions or answers to this forum by sending mail to bind9-users@isc.org.

The z/OS Basic Skills Information Center

The z/OS Basic Skills Information Center is a Web-based information resource intended to help users learn the basic concepts of z/OS, the operating system that runs most of the IBM mainframe computers in use today. The Information Center is designed to introduce a new generation of Information Technology professionals to basic concepts and help them prepare for a career as a z/OS professional, such as a z/OS system programmer.

Specifically, the z/OS Basic Skills Information Center is intended to achieve the following objectives:

- Provide basic education and information about z/OS without charge
- Shorten the time it takes for people to become productive on the mainframe
- Make it easier for new people to learn z/OS

To access the z/OS Basic Skills Information Center, open your Web browser to the following Web site, which is available to all users (no login required):
http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/index.jsp

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this document or any other z/OS Communications Server documentation, do one of the following:

- Go to the z/OS contact page at http://www.ibm.com/systems/z/os/zos/webqs.html. You can enter and submit your comments in the form provided at this Web site.
- Send your comments by e-mail to comsvrcf@us.ibm.com. Be sure to include the name of the document, the part number of the document, the version of z/OS Communications Server, and, if applicable, the specific location of the text that you are commenting on (for example, a section number, a page number or a table number).
Summary of changes

Summary of changes
for GC31-8782-10
z/OS Version 1 Release 11

This document contains information previously presented in GC31-8782-09, which supports z/OS Version 1 Release 10.

New information

- Verbose ping, see "Using the Ping command" on page 37.
- Network management interface enhancements - stack configuration data, see "Sample output of the TCPIPCS PROFILE subcommand" on page 229.
- Sysplex Distributor support for both z/OS targets and non-z/OS targets, see
  - "TCPIPCS ROUTE" on page 237
  - "TCPIPCS TREE" on page 277
  - "Steps for diagnosing Tier 1 z/OS sysplex distribution problems" on page 400
  - "Steps for diagnosing Tier 1 non-z/OS sysplex distribution problems" on page 401
- NSS private key and certificate services for XML appliances, see Table 22 on page 357
- Customizable pre-logon banner for otelnetd, see "Debug trace examples (-t -D all)" on page 490.
- Remote execution server enhancements, see "Example trace to the JES spool file of the server" on page 592.
- Resolver DNS cache, see "Steps for resolving caching problems" on page 855.
- New SMTP client for sending Internet mail, see Chapter 41, "Diagnosing Communications Server SMTP application problems," on page 881.
- Improved responsiveness to storage conditions, see "Steps to take when reviewing a storage problem" on page 904.
- IBM Health Checker for z/OS RFC4301 compliance and for z/OS server check, see Appendix D, "IBM Health Checker for z/OS," on page 951.

Deleted information

- Support for NDB, the DHCP server, BINL, and BIND 4.9.3 is removed from the z/OS V1R11 Communications Server product; information describing this support has been deleted.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

You might notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.
Summary of changes
for GC31-8782-09
z/OS Version 1 Release 10

This document contains information previously presented in GC31-8782-08, which supports z/OS Version 1 Release 9.

New information

- Packet trace enhancements, see "OPTIONS syntax" on page 98.
- Allow multiple sockets to share the same FRCA cache, see "TCPIPCS FRCA" on page 210.
- Netstat enhancements, see "Sample output of the TCPIPCS PROFILE subcommand" on page 229.
- IPCS formatter enhancements, see:
  - "TCPIPCS RAW" on page 235
  - "TCPIPCS STORAGE" on page 264
  - "TCPIPCS TCB" on page 268
  - "TCPIPCS TELNET" on page 270
  - "TCPIPCS UDP" on page 288.
- Subplex support for Load Balancing Advisor, see "Diagnosing Advisor and Agent problems" on page 316.
- DataPower integration with NSS SAF access service, see Chapter 10, "Diagnosing network security services (NSS) server problems," on page 355.
- FTP enhancements, see "FTP connection stops during FILETYPE=JES processing" on page 442.
- Defensive filtering, see Chapter 30, "Diagnosing Defense Manager daemon problems," on page 715.
- Security options for centralized policy server connections, see Chapter 26, "Diagnosing Policy Agent problems," on page 653.
- Configuration Assistant: import of policy configuration data, see "Policy client connection problems" on page 664.
- OMPROUTE enhancements, see "Adjacency failures" on page 754.
- Allow DNS BIND 9 server to run with BPX.SUPERUSER authorization and resolver support for ENDS0, see "TRACE RESOLVER" on page 857.
- IBM Health Checker for z/OS enhancements and IBM Health Checker for z/OS migration check support, see Appendix D, "IBM Health Checker for z/OS," on page 953.

Deleted information

- Removed support for the traffic regulation policy in Policy Agent.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

You might notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.
Summary of changes
for GC31-8782-08
z/OS Version 1 Release 9

This document contains information previously presented in GC31-8782-07, which supports z/OS Version 1 Release 8.

The information in this document includes descriptions of support for both IPv4 and IPv6 networking protocols. Unless explicitly noted, descriptions of IP protocol support concern IPv4. IPv6 support is qualified within the text.

New information
- IPSec network security services, see:
  - Selecting a trace, Table 2 on page 8
  - Chapter 10, “Diagnosing network security services (NSS) server problems,” on page 355
  - Establishing security associations problems, Table 17 on page 335
  - “Using the ipsec command” on page 744
- Policy-based routing (PBR), see Chapter 4, “Diagnosing network connectivity problems,” on page 29
- Ping command detection of network MTU, see “Using the Ping command” on page 37
- IPv6 scoped address architecture API, see “Using the Ping command” on page 37
- OSA-Express2 network traffic analyzer enhancements, see:
  - “OPTIONS syntax” on page 98
  - “Starting OSAENTA trace” on page 185
  - “Modifying options with VARY commands” on page 186
- IPSec network management interface support, see “Initialization problems” on page 334.
- Source IP (SRCIP) enhancements, see Chapter 11, “Diagnosing dynamic VIPA and sysplex problems,” on page 373
- Support for WLM routing service enhancements for zAAP and zIIP, see “Steps for diagnosing sysplex problems” on page 374
- Allow the TN3270E Telnet server only in a separate address space, see Chapter 16, “Diagnosing Telnet problems,” on page 505
- Policy distribution services, see “Diagnosing Policy Agent problems” on page 657
- AT-TLS API enhancements, see:
  - AT-TLS return codes, Table 65 on page 708
  - SIOCCTTLSCTL error codes Table 66 on page 712
- IPSec processing on the zIIP, see:
  - “Steps for verifying IPSec processing on zIIP” on page 744
  - “Determining the Workload Manager service class associated with IPSec workload being processed on zIIP” on page 744
- CICS sockets enhancements, see:
  - “Steps for diagnosing CICS listener not initialized” on page 844
  - “Steps for diagnosing TCP/IP clients unable to connect” on page 845
  - “CICS shutdown hangs” on page 847
• Enhance Netstat ALL/-A report to indicate sockets storage use, see “Problem determination” on page 904.
• Health Checker enhancements, see Appendix D, “IBM Health Checker for z/OS,” on page 951

**Changed information**
• The TCPIPCS ROUTE subcommand has been changed to add new parameters for policy-based routing (PBR).

**Deleted information**
• Removed of QoS and IDS LDAPv2 schema
• The APPC Application Suite is removed from the z/OS V1R9 Communications Server product and therefore documentation describing APPC Application Suite support has been deleted.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

You might notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.
Part 1. General diagnosis information
Chapter 1. Overview of diagnosis procedure

To diagnose a problem suspected to be caused by z/OS Communications Server, first identify the problem, then determine if it is a problem with TCP/IP. If the problem is TCP/IP-related, gather information about the problem so that you can report the source of the problem to the IBM Software Support Center.

With this information, you can work with IBM Software Support Center representatives to solve the problem. This document helps you identify the source of the problem.

Figure 1 on page 4 summarizes the procedure to follow to diagnose a problem. The text following the figure provides more information about this procedure.
Steps for diagnosing problems

Before you begin: You need to know if the source of the problem is TCP/IP.

Perform the following steps to diagnosis a problem.

1. Check sources for diagnostic information.

   Various messages appearing in the console log or in the SYSPRINT or SYSERROR data sets, together with alerts and diagnostic aids, provide information that helps you to find the source of a problem. You should also check syslog output, and syslog daemon messages, and be prepared to provide this information to the IBM Software Support Center. If the problem is with TCP/IP, go to Step 3; otherwise, go to Step 2.
2. Check appropriate books.
   Refer to the diagnosis guide of the hardware device or software application that has the problem.

3. Gather information.
   See Chapter 2, “Selecting tools and service aids,” on page 7, for a detailed explanation of diagnostic procedures and how to collect information relevant to the problem.

4. Try to solve the problem.
   If you cannot solve the problem, go to Step 6.

5. The diagnosis task is completed.
   The problem has been solved.

6. Report the problem to the IBM Software Support Center.
   After you have gathered the information that describes the problem, report it to the IBM Software Support Center. If you are an IBMLink™ user, you can perform your own RETAIN® searches to help identify problems. Otherwise, a representative uses your information to build keywords to search the RETAIN database for a solution to the problem. Alternatively, go to http://www.ibm.com/software/network/comms/serv/.
   The object of this keyword search using RETAIN is to find a solution by matching the problem with a previously reported problem. When IBM develops a solution for a new problem, it is entered into RETAIN with a description of the problem.

7. Work with IBM Support Center representatives.
   If a keyword search matches a previously reported problem, its solution might also correct this problem. If so, go to Step 8. If a solution to the problem is not found in the RETAIN database, the IBM Software Support Center representatives continue to work with you to solve the problem. Go to Step 8.

8. Create an APAR.
   If the IBM Software Support Center does not find a solution, they create an authorized program analysis report (APAR) in the RETAIN database.

9. A solution is developed by the IBM Software Support Center.
   Using information supplied in the APAR, IBM Software Support Center representatives determine the cause of the problem and develop a solution for it.

10. Apply the solution.
Apply the corrective procedure supplied by the IBM Software Support Center to correct the problem.

Go to Step 4 to verify that the problem is corrected. You know you are done when the problem is corrected.
Chapter 2. Selecting tools and service aids

This topic introduces the tools and service aids that z/OS Communications Server provides for diagnosis. As used in this document, the term tools includes dumps and traces, while the term service aids includes all other facilities provided for diagnosis.

For example:
- SVC dump and system trace are tools.
- LOGREC data set and IPCS are service aids.

The following information is discussed in this topic:
- “How do I know which tool or service aid to select?” lists problem types and matches them with the appropriate tool or service aid. Use this topic to select the tool or service aid you need for a particular problem.
- “Overview of available tools and service aids” on page 13 describes each tool and service aid, including when to use it for diagnosis. Use this topic when you need an overview of tools and service aids, or to find the appropriate time to use a particular tool or service aid.
- “Submitting documentation through mailed tape” on page 19 describes the guidelines for submitting machine-readable documentation.
- “Methods for submitting documentation” on page 20 describes how to send documentation electronically to IBM using FTP or e-mail.
- “Necessary documentation” on page 22 lists the documentation you need to gather before contacting the IBM Software Support Center.

How do I know which tool or service aid to select?

This section describes the criteria for selecting a tool or service aid.

Your choice depends on one of the following:

<table>
<thead>
<tr>
<th>Problem or need</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selecting a dump</td>
<td>Table 1 on page 8</td>
</tr>
<tr>
<td>Selecting a TCP/IP services component trace</td>
<td>Table 2 on page 8</td>
</tr>
<tr>
<td>Selecting a service aid</td>
<td>Table 3 on page 12</td>
</tr>
</tbody>
</table>

The tables show the problem, the corresponding tool or service aid, and the topic or document that covers it in more detail. Use these tables to find a tool or service aid quickly.

See “Submitting documentation through mailed tape” on page 19 for information about submitting dumps and traces to the IBM Software Support Center.

Tip: The traces given in this document are only examples. Traces in your environment can differ from these examples because of different options selected.

Selecting a dump

Base your choice of dumps on the criteria given in Table 1 on page 8.
Table 1. Selecting a dump

<table>
<thead>
<tr>
<th>If the problem is ...</th>
<th>Then use this type of dump</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abnormal end of an authorized program or a problem program.</td>
<td>ABEND dump</td>
</tr>
<tr>
<td></td>
<td>See “Analyzing abends” on page 25 for detailed information.</td>
</tr>
<tr>
<td>TCP/IP server or client address space stops processing or is stopped by the operator because of slowdown or looping condition.</td>
<td>SVC dump</td>
</tr>
<tr>
<td></td>
<td>The SVC dump is created using the DUMP command.</td>
</tr>
<tr>
<td></td>
<td>See “Analyzing loops” on page 26 for detailed information.</td>
</tr>
</tbody>
</table>

You can now perform the steps for the decision you have made.

Selecting a trace

Base your choice of traces on the criteria given in Table 2.

Table 2. Selecting a trace

<table>
<thead>
<tr>
<th>If the problem is ...</th>
<th>Then use this type of trace or command</th>
<th>Trace output location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load balancing using the z/OS Load Balancing Advisor</td>
<td>Log file</td>
<td>syslogd</td>
</tr>
<tr>
<td>See Chapter 7, “Diagnosing problems with the z/OS Load Balancing Advisor,” on page 315 for more information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network connectivity</td>
<td>Ping, Netstat ARP/-R</td>
<td>Not applicable</td>
</tr>
<tr>
<td>See Chapter 4, “Diagnosing network connectivity problems,” on page 29 for detailed information.</td>
<td>For information on Ping, see “Using the Ping command” on page 37. For information on Netstat ARP/-R, see “Netstat ARP/-R” on page 43.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Packet trace</td>
<td>CTRACE managed data set</td>
</tr>
<tr>
<td></td>
<td>See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47 for detailed information.</td>
<td></td>
</tr>
<tr>
<td>Dynamic VIPA or Sysplex Distributor</td>
<td>Component Trace (SYSTCPIP) XCF option</td>
<td>TCP/IP address space or external writer</td>
</tr>
<tr>
<td>TCP/IP socket application</td>
<td>Component Trace (SYSTCPIP) SOCKAPI option</td>
<td>TCP/IP address space or external writer</td>
</tr>
<tr>
<td>See “Socket API traces” on page 74 for detailed information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPR client</td>
<td>LPR command with the TRACE option</td>
<td>sysout</td>
</tr>
<tr>
<td>See “LPR client traces” on page 415 for detailed information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>If the problem is ...</td>
<td>Then use this type of trace or command</td>
<td>Trace output location</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td><strong>LPD server</strong></td>
<td>See “LPD server traces” on page 421 for ways to activate traces.</td>
<td><strong>SYSPRINT</strong></td>
</tr>
<tr>
<td><strong>z/OS UNIX FTP server</strong></td>
<td><strong>z/OS UNIX FTP server trace</strong></td>
<td><strong>Server traces appear on the console if syslogd is not started. If it is started, traces appear in the file designated in the syslog.conf file. Refer to the z/OS Communications Server: IP Configuration Guide for detailed information about syslogd.</strong></td>
</tr>
<tr>
<td><strong>z/OS UNIX Telnet</strong></td>
<td><strong>z/OS UNIX Telnet traces</strong></td>
<td><strong>syslogd</strong></td>
</tr>
<tr>
<td><strong>TN3270E Telnet server</strong></td>
<td>Telnet traces</td>
<td><strong>Telnet address space or external writer</strong></td>
</tr>
<tr>
<td><strong>SMTP</strong></td>
<td><strong>Resolver Trace (see also “Debugging with a resolver directive” on page 583)</strong></td>
<td><strong>Job log output</strong></td>
</tr>
<tr>
<td><strong>Popper</strong></td>
<td><strong>Popper Messages</strong></td>
<td><strong>syslogd</strong></td>
</tr>
<tr>
<td><strong>SNALINK LU0</strong></td>
<td><strong>IP Packet Trace</strong></td>
<td><strong>CTRACE managed data set</strong></td>
</tr>
<tr>
<td><strong>SNALINK LU6.2</strong></td>
<td><strong>TRACE DETAIL ALL</strong></td>
<td><strong>SYSPRINT</strong></td>
</tr>
<tr>
<td><strong>SNALINK LU6.2</strong></td>
<td><strong>IP Packet Trace</strong></td>
<td><strong>CTRACE managed data set</strong></td>
</tr>
<tr>
<td><strong>SNALINK LU6.2</strong></td>
<td><strong>TCP/IP Internal Trace</strong></td>
<td><strong>CTRACE managed data set</strong></td>
</tr>
<tr>
<td><strong>SNALINK LU6.2</strong></td>
<td><strong>VTAM Buffer Trace</strong></td>
<td><strong>GTF managed data set, refer to z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures for detailed information.</strong></td>
</tr>
<tr>
<td><strong>Dynamic domain name system (DDDNS)</strong></td>
<td><strong>Error messages</strong></td>
<td><strong>syslogd</strong></td>
</tr>
<tr>
<td><strong>Dynamic domain name system (DDDNS)</strong></td>
<td><strong>Resolver Trace</strong></td>
<td><strong>Job log output</strong></td>
</tr>
<tr>
<td><strong>Dynamic domain name system (DDDNS)</strong></td>
<td><strong>TCP/IP component trace</strong></td>
<td><strong>CTRACE managed data set</strong></td>
</tr>
</tbody>
</table>
Table 2. Selecting a trace (continued)

<table>
<thead>
<tr>
<th>If the problem is ...</th>
<th>Then use this type of trace or command</th>
<th>Trace output location</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS UNIX REXEC</td>
<td>z/OS UNIX REXEC debug trace</td>
<td>syslogd</td>
</tr>
<tr>
<td>See Chapter 23, “Diagnosing z/OS UNIX REXEC, RSH, REXECD, and RSHD problems,” on page 595</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z/OS UNIX REXECD</td>
<td>z/OS UNIX REXECD debug trace</td>
<td>syslogd</td>
</tr>
<tr>
<td>See Chapter 23, “Diagnosing z/OS UNIX REXEC, RSH, REXECD, and RSHD problems,” on page 595</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z/OS UNIX RSHD</td>
<td>z/OS UNIX RSHD debug trace</td>
<td>syslogd</td>
</tr>
<tr>
<td>See Chapter 23, “Diagnosing z/OS UNIX REXEC, RSH, REXECD, and RSHD problems,” on page 595</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X Windows® and Motif</td>
<td>XWTRACE and XWTRACEC (environment variables)</td>
<td>stderr</td>
</tr>
<tr>
<td>SNMP</td>
<td>Manager Traces</td>
<td>Console (snmp) or SYSPRINT (NetView® SNMP)</td>
</tr>
<tr>
<td></td>
<td>• SNMP Agent Traces</td>
<td>syslogd</td>
</tr>
<tr>
<td></td>
<td>• TCP/IP Subagent Traces</td>
<td>syslogd</td>
</tr>
<tr>
<td></td>
<td>• OMPROUTE Subagent Traces</td>
<td>syslogd</td>
</tr>
<tr>
<td></td>
<td>• Network SLAPM2 Subagent Traces</td>
<td>syslogd</td>
</tr>
<tr>
<td></td>
<td>• TN3270E Telnet Subagent Traces</td>
<td>syslogd</td>
</tr>
<tr>
<td></td>
<td>• TRAPFWD Traces</td>
<td>syslogd</td>
</tr>
<tr>
<td>Policy Agent</td>
<td>Log file</td>
<td>Refer to the z/OS Communications Server: IP Configuration Guide for detailed information.</td>
</tr>
<tr>
<td>RSVP Agent</td>
<td>Log file</td>
<td>Refer to the z/OS Communications Server: IP Configuration Guide for detailed information.</td>
</tr>
<tr>
<td>Traffic Regulator Management Daemon (TRMD)</td>
<td>Log file</td>
<td>syslogd</td>
</tr>
</tbody>
</table>
### Table 2. Selecting a trace (continued)

<table>
<thead>
<tr>
<th>If the problem is ...</th>
<th>Then use this type of trace or command</th>
<th>Trace output location</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IKE daemon</strong></td>
<td>Component trace</td>
<td>CTRACE managed data set</td>
</tr>
<tr>
<td>See Chapter 9, “Diagnosing IKE daemon problems,” on page 333.</td>
<td>For detailed information about IKE daemon component trace, see “TCP/IP services component trace for the IKE daemon” on page 330.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Log file</td>
<td>syslogd</td>
</tr>
<tr>
<td></td>
<td>For detailed information about obtaining IKE daemon debug log information, see “ Obtaining syslog debug information for the IKE daemon” on page 349.</td>
<td></td>
</tr>
<tr>
<td><strong>Network security services (NSS) server</strong></td>
<td>Component trace</td>
<td>CTRACE managed data set</td>
</tr>
<tr>
<td>See Chapter 10, “Diagnosing network security services (NSS) server problems,” on page 355.</td>
<td>For detailed information about network security services (NSS) server component trace, see “TCP/IP services component trace for the network security services (NSS) server” on page 368.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Log file</td>
<td>syslogd</td>
</tr>
<tr>
<td></td>
<td>For detailed information about obtaining network security services (NSS) server debug log information, see “ Obtaining syslog debug information for the network security server” on page 362.</td>
<td></td>
</tr>
<tr>
<td><strong>OMPROUTE</strong></td>
<td>Component trace</td>
<td>CTRACE managed data set</td>
</tr>
<tr>
<td>See Chapter 32, “Diagnosing OMPROUTE problems,” on page 749.</td>
<td>For detailed information about OMPROUTE Component Trace, see “TCP/IP services component trace for OMPROUTE” on page 772.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OMPROUTE Trace</td>
<td>stdout</td>
</tr>
<tr>
<td></td>
<td>For detailed information, see “OMPROUTE traces and debug information” on page 793.</td>
<td></td>
</tr>
<tr>
<td><strong>NCPROUTE</strong></td>
<td>NCPROUTE Traces</td>
<td>SYSPRINT</td>
</tr>
<tr>
<td><strong>X.25 NPSI</strong></td>
<td>Server activity log</td>
<td>SYSPRINT</td>
</tr>
</tbody>
</table>
### Table 2. Selecting a trace (continued)

<table>
<thead>
<tr>
<th>If the problem is ...</th>
<th>Then use this type of trace or command</th>
<th>Trace output location</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS™</td>
<td>IP Packet Trace</td>
<td>CTRACE managed data set</td>
</tr>
<tr>
<td></td>
<td>TCP/IP Internal Trace</td>
<td>CTRACE managed data set</td>
</tr>
<tr>
<td></td>
<td>IMS Trace</td>
<td>Refer to the <a href="https://example.com">IMS Version 8: Utilities Reference: System</a> for detailed information.</td>
</tr>
<tr>
<td>CICS®</td>
<td>CICS external trace data set (auxtrace)</td>
<td>Refer to the <a href="https://example.com">CICS/ESA® 5.2 Problem Determination Guide</a> for detailed information.</td>
</tr>
<tr>
<td></td>
<td>TCP/IP Internal trace</td>
<td>CTRACE managed data set</td>
</tr>
<tr>
<td>Express Logon</td>
<td>Log file</td>
<td>syslogd</td>
</tr>
<tr>
<td>Resolver</td>
<td>Trace Resolver</td>
<td>SYSPRINT or stdout</td>
</tr>
<tr>
<td></td>
<td>Resolver Internal trace</td>
<td>CTRACE managed data set</td>
</tr>
</tbody>
</table>

You can now perform the steps for the decision you have made.

### Selecting a service aid

Base your choice of service aid on the criteria given in Table 3.

### Table 3. Selecting a service aid

<table>
<thead>
<tr>
<th>If the problem is...</th>
<th>Then use this type of service aid</th>
</tr>
</thead>
<tbody>
<tr>
<td>System or hardware problem: need a starting point for diagnosis or diagnosis requires an overview of system and hardware events in chronological order.</td>
<td>LOGREC data set or EREP</td>
</tr>
<tr>
<td>Information about the contents of load modules and program objects or a problem with modules on the system.</td>
<td>AMBLIST</td>
</tr>
<tr>
<td>Diagnosis requires a trap to catch problem data while a program is running. The DISPLAY TCPIP,STOR command can be used to help set a SLIP trap.</td>
<td>Service Level Indication Processing (SLIP)</td>
</tr>
<tr>
<td>Diagnosis requires formatted output of problem data, such as a dump or trace.</td>
<td>IPCS</td>
</tr>
</tbody>
</table>

You can now perform the steps for the decision you have made.
Overview of available tools and service aids

This section provides an overview of the tools and service aids in detail. The sections that follow contain a brief description of each tool or service aid, reasons why you would use it, and a reference to the topic or document that covers the tool or service aid in detail. (Most of the detailed information on tools and service aids is in this document.)

A description of tools and service aids are included in the following sections:

- Dumps, see Table 4
- Traces, see Table 5 on page 14
- First Failure Support Technology™, see “First Failure Support Technology (FFST)” on page 16
- Display commands, see “Display commands” on page 18
- System service aids, see Table 6 on page 18

In the tables that follow, the dumps, traces, or service aids are listed by frequency of use.

Tip: The traces given in this document are only examples. Traces in your environment can differ from these examples because of different options selected.

Dumps

Table 4 describes the types of available dumps.

Table 4. Description of dumps

<table>
<thead>
<tr>
<th>Type of dump</th>
<th>Description</th>
</tr>
</thead>
</table>
| ABEND dumps  | Use an ABEND dump when ending an authorized program or a problem program because of an uncorrectable error. These dumps show:  
  - The virtual storage for the program requesting the dump.  
  - System data associated with the program.  
  The system can produce three types of ABEND dumps—SYSABEND, SYSMDUMP, and SYSUDUMP. Each one dumps different areas. Select the dump that gives the areas needed for diagnosing your problem. The IBM-supplied defaults for each dump are:  
  - SYSABEND dumps. The largest of the ABEND dumps, containing a summary dump for the failing program plus many other areas useful for analyzing processing in the failing program.  
  - SYSMDUMP dumps. Contains a summary dump for the failing program, plus some system data for the failing task. In most cases, SYSMDUMP dumps are recommended, because they are the only ABEND dumps that are formatted with IPCS.  
  - SYSUDUMP dumps. The smallest of the ABEND dumps, containing only data and areas about the failing program.  
  Reference: Refer to z/OS MVS Diagnosis: Tools and Service Aids for more information about ABEND |

Chapter 2. Selecting tools and service aids
Table 4. Description of dumps (continued)

<table>
<thead>
<tr>
<th>Type of dump</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVC dumps</td>
<td>SVC dumps can be used in two different ways:</td>
</tr>
<tr>
<td></td>
<td>• Most commonly, a system component requests an SVC dump when an unexpected system error occurs, but the system can continue processing.</td>
</tr>
<tr>
<td></td>
<td>• An authorized program or the operator can also request an SVC dump when diagnostic data is needed to solve a problem.</td>
</tr>
<tr>
<td></td>
<td>SVC dumps contain a summary dump, control blocks, and other system code, but the exact areas dumped depend on whether the dump was</td>
</tr>
<tr>
<td></td>
<td>requested by a macro, command, or SLIP trap. SVC dumps can be analyzed using IPCS.</td>
</tr>
<tr>
<td></td>
<td><strong>Reference:</strong> Refer to <a href="#">z/OS MVS Diagnosis: Tools and Service Aids</a> for detailed information.</td>
</tr>
<tr>
<td></td>
<td>If a console dump or SLIP is requested:</td>
</tr>
<tr>
<td></td>
<td>• Capture the OMVS and (if applicable) affected application address spaces as well as TCP/IP.</td>
</tr>
<tr>
<td></td>
<td>• Include all TCP/IP data spaces.</td>
</tr>
<tr>
<td></td>
<td>• SDATA specification should contain the RGN, TRT, PSA, SUM, CSA and SQA keywords (at minimum).</td>
</tr>
<tr>
<td>FFST™ dumps</td>
<td>FFST dumps fall into two categories: SDUMPs (full dumps) and FFST minidumps (partial dumps). The type of dump produced depends on the</td>
</tr>
<tr>
<td></td>
<td>characteristics of the probe that produced it.</td>
</tr>
<tr>
<td></td>
<td>• FFST uses the operating system SDUMP macroinstruction to provide a full dump of the address space where the problem occurred.</td>
</tr>
<tr>
<td></td>
<td>• If the SDUMP option has not been coded for the probe triggering the dump, an FFST minidump is written to the output data set. The probe</td>
</tr>
<tr>
<td></td>
<td>output data for the TCP/IP minidumps are found in data sets that were allocated when FFST was installed.</td>
</tr>
<tr>
<td>Stand-alone dumps</td>
<td>Use a stand-alone dump when:</td>
</tr>
<tr>
<td></td>
<td>• The system stops processing.</td>
</tr>
<tr>
<td></td>
<td>• The system enters a wait state with or without a wait state code.</td>
</tr>
<tr>
<td></td>
<td>• The system enters an instruction loop.</td>
</tr>
<tr>
<td></td>
<td>• The system is processing slowly.</td>
</tr>
<tr>
<td></td>
<td>These dumps show central storage and some paged-out virtual storage occupied by the system or stand-alone dump program that failed.</td>
</tr>
<tr>
<td></td>
<td>Stand-alone dumps can be analyzed using IPCS.</td>
</tr>
<tr>
<td></td>
<td>See <a href="#">Analyzing loops</a> on page 26 for detailed information.</td>
</tr>
</tbody>
</table>

**Traces**

Table 5 describes the types of available traces.

Table 5. Description of traces

<table>
<thead>
<tr>
<th>Type of trace</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component trace</td>
<td>Use a component trace when you need trace data to report a client/server component problem to the IBM Software Support Center. Component tracing shows processing between the client and server.</td>
</tr>
<tr>
<td></td>
<td><strong>Reference:</strong> See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47 for detailed information.</td>
</tr>
</tbody>
</table>
Table 5. Description of traces (continued)

<table>
<thead>
<tr>
<th>Type of trace</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data trace</td>
<td>Use a data trace to trace socket data (transforms) into and out of the physical file structure (PFS). Refer to <a href="#">Data trace (SYSTCPDA) for TCP/IP stacks</a> for detailed information.</td>
</tr>
<tr>
<td>GTF trace</td>
<td>Use a Generalized Trace Facility (GTF) trace to show system processing through events occurring in the system over time. The installation controls which events are traced. Use GTF when you are familiar enough with the problem to pinpoint the one or two events required to diagnose your system problem. GTF can be run to an external data set. Refer to <a href="#">z/OS MVS Diagnosis: Tools and Service Aids</a> for more information about GTF.</td>
</tr>
<tr>
<td>Master trace</td>
<td>Use the master trace to show the messages to and from the master console. Master trace is useful because it provides a log of the most recently issued messages. These can be more pertinent to your problem than the messages accompanying the dump itself. You can either accept a dump or write this trace to GTF. Refer to <a href="#">z/OS MVS Diagnosis: Tools and Service Aids</a> for detailed information.</td>
</tr>
<tr>
<td>OSAENTA trace</td>
<td>Use a OSA-Express network traffic analysis trace to obtain traces of IP packets flowing from and into TCP/IP on a z/OS Communications Server host. The OSAENTA statement lets you copy IP packets as they enter or leave OSA-Express adapter, and then examine the contents of the copied packets. While the packet trace collects data records that flow over the links, the OSAENTA trace collects data records that flow from the network through the OSA adapter. Refer to <a href="#">Chapter 5, “TCP/IP services traces and IPCS support,” on page 47</a> for detailed information.</td>
</tr>
<tr>
<td>Packet trace</td>
<td>Use a packet trace to obtain traces of IP packets flowing from and into TCP/IP on a z/OS Communications Server host. The PKTTRACE statement lets you copy IP packets as they enter or leave TCP/IP, and then examine the contents of the copied packets. While the component trace function collects event data about TCP/IP internal processing, packet trace collects data records that flow over the links. Refer to <a href="#">Chapter 5, “TCP/IP services traces and IPCS support,” on page 47</a> for detailed information.</td>
</tr>
<tr>
<td>System trace</td>
<td>Use system trace to see system processing through events occurring in the system over time. System tracing is activated at initialization and, typically, runs continuously. It records many system events, with minimal details about each. The events traced are predetermined, except for branch tracing. You can either take a dump or write this trace to GTF. Refer to <a href="#">z/OS MVS Diagnosis: Tools and Service Aids</a> for detailed information.</td>
</tr>
</tbody>
</table>
### Table 5. Description of traces (continued)

<table>
<thead>
<tr>
<th>Type of trace</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTAM trace</td>
<td>VTAM traces contain entries for many TCP/IP events, especially I/O and storage requests. Refer to z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT for detailed information.</td>
</tr>
<tr>
<td>z/OS UNIX applications</td>
<td>z/OS UNIX applications send debug and trace output to syslogd. For more information on individual components, such as z/OS UNIX FTP or z/OS UNIX SNMP, refer to those topics in this manual. ITRACE initiated from TCPIP PROFILE processing</td>
</tr>
</tbody>
</table>

**First Failure Support Technology (FFST)**

First Failure Support Technology (FFST) is a licensed program that captures information about a potential problem when it occurs. See Appendix A, “First Failure Support Technology (FFST),” on page 907 for descriptions of the various FFST probes contained in TCP/IP.

Note: For a complete description of FFST commands, refer to the FFST/MVS FFST/VM Operations Guide.

When a problem is detected, a software probe is triggered by TCP/IP. FFST then collects information about the problem and generates output to help solve the problem. Based on the options active for the probe, you get a dump and a generic alert. See “Generic alert” on page 17 for information on generic alerts. You also get the FFST “EPW” message group.

**FFST dumps**

Each TCP/IP Services FFST probe can trip up to five times in five minutes before it is automatically turned off. Only one of the five dumps is produced, thereby limiting the number of dumps that you get if a recurring problem triggers a probe.

You get either an SDUMP (full dump) or an FFST minidump (partial dump) depending on the characteristics of the probe that is triggered.

FFST saves the TCP/IP minidump on a dynamically allocated sequential data set. The TCP/IP Services FFST full dump (SDUMP) is saved on SYSLDUMPx data sets. You must specify the volume serial number and the UNIT identification information for this data set. Provide this information to FFST on a DD statement in the FFST installation procedure or in the FFST startup command list installed at system installation. A startup command list contains MVS commands to control FFST.

**SDUMP**

The SDUMP option is coded in the probe; FFST uses the operating system SDUMP macroinstruction to provide a full dump of the address space where the potential problem occurred.
Formatting an SDUMP
Use the standard IPCS dump formatting and viewing facilities to access the dump. If you use the EPWDMPFM clist to format a full dump, message EPW9561E, NOT A VALID FFST DUMP is issued.

FFST minidump
If the SDUMP option has not been coded for the probe triggering the dump, an FFST minidump is written to the output data set. The probe output data for the TCP/IP minidumps are found in the data sets that were allocated when FFST was installed.

Formatting an FFST minidump
Use the dump formatting CLIST, EPWDMPFM, to format your TCP/IP Services FFST minidump. EPWDMPFM formats your minidump and writes it to a data set you can view online or print using the IEBPTPCH utility program.

Generic alert
A software generic alert is built from the symptom record and routed to the NetView program, if installed. The generic alert contains the following:
• Date and time that the probe was triggered
• System name from the CVTSNAME field
• Product name (TCP)
• Component identifier and the release number of the product triggering the probe
• Hardware identification information:
  – Machine type
  – Serial number
  – Model number
  – Plant code
• Dump data set and volume, if a dump was taken
• Probe statement
• Statement description
• Probe statement severity level

The symptom string
The primary symptom string contains the following data supplied by TCP/IP:
• PIDS/component IP. The TCP/IP component identifier.
• LVLS/level. The TCP/IP specification for the product level.
• PCSS/Probe ID. From the probe that was triggered.
• PCSS/FULL or MINI. Type of dump taken.
• RIDS. Module name from the probe that was triggered.

FFST console
The following is a sample for a console listing for FFST. In this sample, the FFST program console message group “EPW” shows information that a probe has been triggered and that the data is being collected. The EPW0404I message contains the primary symptom string for TCP/IP.
Display commands

Display commands can be useful tools and service aids. This section provides a brief description of the DISPLAY TCPIP,STOR command. For detailed information about this command, refer to the z/OS Communications Server: IP System Administrator’s Commands.

DISPLAY TCPIP,STOR

Use the DISPLAY TCPIP,STOR command to display the location and level of a TCP/IP stack module, which verifies that the load module has the appropriate service level.

System service aids

Table 6 lists the service aids supported by z/OS Communications Server.

<table>
<thead>
<tr>
<th>Type of service aid</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMBLIST</td>
<td>Use AMBLIST when you need information about the contents of load modules and program objects or you have a problem related to the modules on your system. AMBLIST is a program that provides extensive data about modules in the system, such as a listing of the load modules, map of the CSECTs in a load module or program object, list of modifications in a CSECT, map of modules in the LPA, and a map of the contents of the DAT-on nucleus. Reference: Refer to the z/OS MVS Diagnosis: Tools and Service Aids for more information about AMBLIST.</td>
</tr>
<tr>
<td>Common storage tracking</td>
<td>Use common storage tracking to collect data about requests to obtain or free storage in CSA, ECSA, SQA, and ESQA. This is useful to identify jobs or address spaces using an excessive amount of common storage or ending without freeing storage. Use Resource Measurement Facility* (RMF®) or the IPCS VERBEXIT VSMDATA subcommand to display common storage tracking data. References: • Refer to the z/OS RMF User’s Guide for more information about RMF®. • Refer to the z/OS MVS Initialization and Tuning Guide for detailed information about requesting common storage tracking. • Refer to the VSM topic of the z/OS MVS IPCS User’s Guide for information about the IPCS VERBEXIT VSMDATA subcommand.</td>
</tr>
</tbody>
</table>
### Table 6. Description of service aids (continued)

<table>
<thead>
<tr>
<th>Type of service aid</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dump suppression</strong></td>
<td>Dump Suppression allows an installation to control dump analysis and elimination (DAE) processing, which suppresses dumps that it considers unnecessary because they duplicate previously taken dumps. DAE suppresses ABEND dumps that would be written to a SYSMDUMP data set (SYSMDUMPs), Transaction dumps (IEATDUMP), and SVC dumps, when the symptom data of a dump duplicates the symptom data of a dump of the same dump type previously taken. DAE uses the ADYSETxx parmlib member to determine the actions DAE is to perform.</td>
</tr>
<tr>
<td><strong>Tip:</strong> Consider the SUPPRESSALL statement in ADYSETxx, if dumps are to be considered for suppression. Do this because the Communications Server IP Recovery Routines do not always specify the VRADAE Key in the SDWA (system diagnostic work area) when requesting a dump.</td>
<td></td>
</tr>
<tr>
<td><strong>Reference:</strong> Refer to the <a href="https://www.ibm.com/support/docview.ws/?uid=swg21461571">z/OS MVS Initialization and Tuning Guide</a> for more information about requesting dump suppression.</td>
<td></td>
</tr>
<tr>
<td><strong>IPCS</strong></td>
<td>Use IPCS to format and analyze dumps, traces, and other data. IPCS produces reports that can help in diagnosing a problem. Some dumps, such as SNAP, SYSABEND, and SYSUDUMP ABEND dumps, are preformatted and are not formatted using IPCS.</td>
</tr>
<tr>
<td><strong>Reference:</strong> Refer to the <a href="https://www.ibm.com/support/docview.ws/?uid=swg21460312">z/OS MVS IPCS User’s Guide</a> for detailed information.</td>
<td></td>
</tr>
<tr>
<td><strong>LOGREC data set</strong></td>
<td>Use the LOGREC data set as a starting point for problem determination. The system records hardware errors, selected software errors, and selected system conditions in the LOGREC data set. LOGREC information gives you an idea of where to look for a problem, supplies symptom data about the failure, and shows the order in which the errors occurred.</td>
</tr>
<tr>
<td><strong>Reference:</strong> Refer to the <a href="https://www.ibm.com/support/docview.ws/?uid=swg21462619">z/OS MVS Diagnosis: Tools and Service Aids</a> for detailed information.</td>
<td></td>
</tr>
<tr>
<td><strong>SLIP traps</strong></td>
<td>Use serviceability level indication processing (SLIP) to set a trap to catch problem data. SLIP can intercept program event recording (PER) or error events. When an event that matches a trap occurs, SLIP performs the problem determination action that you specify:</td>
</tr>
<tr>
<td>• Requesting or suppressing a dump</td>
<td></td>
</tr>
<tr>
<td>• Writing a trace or a LOGREC data set record</td>
<td></td>
</tr>
<tr>
<td>• Giving control to a recovery routine</td>
<td></td>
</tr>
<tr>
<td>• Putting the system in a wait state</td>
<td></td>
</tr>
<tr>
<td><strong>Reference:</strong> Refer to the SLIP command in <a href="https://www.ibm.com/support/docview.ws/?uid=swg21461732">z/OS MVS System Commands</a> for detailed information.</td>
<td></td>
</tr>
</tbody>
</table>

---

**Submitting documentation through mailed tape**

Submitting documentation electronically is preferred whenever possible. If, after talking to the IBM Support Center representative about a problem, it is decided that documentation should be submitted to the TCP/IP support team, and electronic submission is not possible, documentation can be submitted on a tape. Documentation on tape can be handled most efficiently by the IBM Support Center if it conforms to the following guidelines.

**Tip:** Trace data and dumps created by TCP/IP can contain user IDs, passwords, and other sensitive information. The trace data files should be protected to prevent disclosure. As an example, packet trace of the FTP port 21 used to control FTP
sessions contains user IDs and passwords in clear text. However, a customer can use Secure Socket Layer for FTP and for TELNET. The Packet Trace (VC
TCP/IP, PKTTRACE) command can be RACF® protected.

**Guidelines:** When preparing documentation on tape for submission in an MVS environment, follow these guidelines:

- Submit the dumps and traces in their original format.
  - For dumps:
    - Dump data should not be formatted in any way prior to or during the transfer of the dump data set.
    - The DCB parameters of the dump data set should not be changed. The DCB parameters should be:
      - LRECL=4160, BLKSIZE= n*4160, RECFM=FBS (for z/OS CS) - where n is 1 to 7.
  - For external CTRACE, IP packet trace, and data trace:
    - CTRACE data should not be formatted in any way prior to or during the transfer of the data set. DCB parameters of the CTRACE data set should not be changed.
    - The IPCS commands COPYDUMP and COPYTRC can also be used. For more information, refer to the [z/OS MVS IPCS Commands](#).
  - For GTF traces:
    - GTF trace data should be copied using IEBGENER only.
    - DCB parameters of the GTF data set should not be changed. A GTF trace should be RECFM=VB(A).

For both traces and dumps, do not reblock the data (that is, do not use a different BLKSIZE value) when moving the information.

**Tip:** Use of any other utility (IBM or non-IBM) to transfer dump or trace data to tape might result in a processing delay and could result in the APAR being returned to the customer (closed RET), due to the inability of the change team to process the tape.

- Submit other types of information (such as TCP/IP traces, configuration files, console logs, and so forth) in machine readable format (preferred) or on paper. If submitted on tape, the data should be written to tape using IEBGENER only. The DCB parameters used when writing this type of data to tape should be the same as the input data set (that is, the same DCB parameters as the source of the data).

### Methods for submitting documentation

You can send documentation to IBM using:

- File Transfer Protocol (FTP)
- e-mail
- TCP/IP active storage or the location and level of a TCP/IP stack module.

**Tip:** If you use FTP, compress all dumps and traces with the AMATERSE (MVS terse) program, and send the data in BINARY mode.

**Requirement:** AMATERSE is prerequisite for PUTDOC.

To obtain PUTDOC and detailed instruction on its use, follow these steps:
Steps for obtaining PUTDOC

Perform the following steps to obtain PUTDOC:


2. Log in using anonymous as the user ID and your e-mail address as the password.

3. Change directories (cd) to /s390/mvs/tools/putdoc/, where you find three files: PUTDOC.BIN, PUTDOC.HTML and PUTDOC.SRC.

4. Read the PUTDOC.HTML file for detailed instructions.

Using AMATERSE

AMATERSE is an application that prepares diagnostic materials, such as z/OS dumps and traces, for transmission to IBM and vendor sites. When the materials arrive, AMATERSE also provides a means to create similar data sets to support diagnosis of problems.

If you have previously used the TRSMAIN utility, you will find that the following changes have been made to prepare AMATERSE for formal inclusion in z/OS:

- AMATERSE is used as the preferred application program name rather than TRSMAIN. TRSMAIN is shipped as an alias entry point to AMATERSE.
- The ddnames INFILE and OUTFILE that were required by the TRSMAIN utility are replaced by SYSUT1 and SYSUT2 respectively. When the TRSMAIN entry point of AMATERSE is invoked, ddnames INFILE and OUTFILE remain as the defaults.
- AMATERSE is placed into MIGLIB, a library that is part of the link list. No STEPLIB ddname is needed to invoke AMATERSE.
- You can use AMATERSE, the TRSMAIN utility, and VM terse interchangeably in nearly all cases.

Starting AMATERSE

The following sample JCL can be used to start AMATERSE. Lower case text reflects the data that you must alter.

```
  //jobname  JOB ... 
  //stepname EXEC  PGM=AMATERSE,PARM=aaaaa 
  //SYSPRINT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=12901)
  //SYSUT1 DD DISP=bbb,DSN=your.input.dataset.name 
  //SYSUT2 DD DISP=ccc,DCB=ddd,DSN=your.output.dataset.name 
  // SPACE=space_parameters
```

For more information on how to use AMATERSE and any restrictions on its use, see [z/OS MVS Diagnosis: Tools and Service Aids](#).

Using electronic transfer through e-mail attachments

Smaller documents can be sent as attachments to an e-mail message. This can include cut and paste of user output or downloading of the file to a workstation for inclusion. Displayable text can be downloaded using ASCII transfer; all others should be processed by the AMATERSE utility and transferred in BINARY. E-mail
systems usually have limits on how much data can be included, so FTP transfers should be used for any significant amounts (the IBM mail system limit is 10M).

**Transferring data sets using tape**

Tapes that are submitted to the TCP/IP support team can be standard label (SL) or nonlabel (NL). Each tape should contain an external label to identify the tape and its contents in some way. The problem number or APAR number should appear on the label. If multiple tapes, or multiple files on one tape, are used, a separate explanation should be included, itemizing the contents of each tape or file.

Include the output from the job used to create each tape with the tapes. It is very important that the IBM Software Support Center have the output from the job that created the tape (not simply the JCL that was used) to verify that the tape was created correctly and that the job completed normally.

**Necessary documentation**

Before you call the IBM Support Center, have the following information available:

**Customer number**
- The authorization code that allows you to use the IBM Support Center.
- Your account name, your TCP/IP license number, and other customer identification should also be available.

**Problem number**
- The problem number previously assigned to the problem. If this is your first call about the problem, the support center representative assigns a number to the problem.

**Operating system**
- The operating system that controls the execution of programs (such as z/OS), include the release level.

**Language Environment® run-time library**
- The release level of the link edit run-time library is also needed if you are compiling user-written applications written in C or C++.

**Component ID**
- A number that is used to search the database for information specific to TCP/IP. If you do not give this number to the support center representative, the amount of time taken to find a solution to your problem increases.

**Release number**
- A number that uniquely identifies each TCP/IP release.

Table 7 on page 23 lists the specific information that you should provide to the IBM Support Center.
Table 7. TCP/IP component name and release level

<table>
<thead>
<tr>
<th>Component name and release level</th>
<th>System maintenance program</th>
<th>Field maintenance identifier/CLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS Communications Server V1R10</td>
<td>SMP/E</td>
<td>The following identifiers are associated with this stack:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIP61A0 (Base)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• JIP61AK (Security Level 3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• JIP61AX (X Window System X11R4)</td>
</tr>
</tbody>
</table>

The following are component ID numbers for z/OS Communications Server:

**Licensed IBM program**

z/OS

**Component ID number**

5694–A01

A complex problem might require you to talk to several people when you report your problem to the IBM Support Center. Therefore, you should keep all the information that you have gathered readily available. You might want to keep the items that are constantly required, such as the TCP/IP component ID, in a file for easy access.
Chapter 3. Diagnosing abends, loops, and hangs

This topic contains information about abends, loops, and hangs.

This topic contains the following sections:
- “Analyzing abends”
- “Analyzing loops” on page 26
- “Steps for analyzing hangs” on page 27

Analyzing abends

An abend is an abnormal end.

Table 8 describes the types of abends that can occur.

Table 8. Types of abends

<table>
<thead>
<tr>
<th>Type of abend</th>
<th>Description</th>
<th>Where to find help</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Abends</td>
<td>User abends are generated by C run-time routines. They usually start with U409x.</td>
<td>Refer to the <a href="link">z/OS Communications Server: IP and SNA Codes</a></td>
</tr>
<tr>
<td>Platform abends</td>
<td>Abend 3C5 and abend 4C5 are internal abends generated by TCP/IP. Note the reason code stored in register 15 and check the IBM database for known problems.</td>
<td>Refer to the <a href="link">z/OS Communications Server: IP and SNA Codes</a></td>
</tr>
<tr>
<td>System abends</td>
<td>0C4, 0C1, and 878 are system abends. 0D6/0D4/0C4 abends can occur when an application is removed from VMCF/TNF with the F VMCF/TNF, REMOVE command, or if VMCF is not active when an application or command, which requires it is started or issued.</td>
<td>Refer to the <a href="link">z/OS MVS System Codes</a> Can occur when an application is removed from VMCF/TNF with the F VMCF/TNF, REMOVE command. It can also occur when an application or command, which requires it is started or issued. The following TCP/IP commands will abend if VMCF is not active: HOMETEST, RSH, REXEC, and TELNET.</td>
</tr>
<tr>
<td>CEEDUMPs</td>
<td>Language Environment produces certain types of abends detected for z/OS UNIX applications such as z/OS UNIX Telnet. CEEDUMPs are usually written to the current working directory in the hierarchical file structure.</td>
<td>Refer to the <a href="link">z/OS Language Environment Debugging Guide</a> publication.</td>
</tr>
</tbody>
</table>

A dump is usually produced when TCP/IP or a TCP/IP component address space experiences an abend. If an abend occurs and no dump is taken, the dump files or
spools might be full or a SYSMDUMP DD statement might not have been specified in the failing procedure. If TCP/IP or a TCP/IP component was not able to complete the dump, gather a console dump of TCP/IP or the failing TCP/IP component, the TCP/IP trace data space or external trace data set, and system log as soon as possible. Otherwise, you must re-create the abend or wait for it to occur again.

For more information about debugging the abends and the system abends (for example, abends 0C4, 0C1, and 878), refer to the [z/OS Problem Management](#).

## Analyzing loops

If processing stops or if TCP/IP does not respond to commands, TCP/IP might be in a loop. Some indicators of a loop are:

- Slow response time
- No response at all
- Inordinately high CPU utilization by TCP/IP

## Steps for collecting documentation

If the problem is a loop, perform the following steps to collect documentation.

1. Get dump output.

   - **Enabled**
     
     Get an SVC dump of TCP/IP or the looping TCP/IP component by issuing the DUMP command from the MVS system console, or press the Program Restart key. Refer to the [z/OS MVS Diagnosis: Tools and Service Aids](#) for more information about the DUMP command.

     **Guidelines:** Ensure that the following storage areas are dumped because they might be needed to diagnose the TCP loop:
     
     - TCP/IP and VTAM address spaces.
     - SDATA options RGN, CSA, LSQA, NUC, PSA, and LPA.
     - TCP/IP data space TCPIPDS1, which contains the TCP/IP component trace records.
     - CSM dataspace. To find the name of the CSM dataspace, issue the DISPLAY net,CSM command. Specify the CSM dataspace name in the DUMP command as DSPNAME=(1.dddddddd) where dddddddd is the name of the CSM dataspace.

     For examples of the DUMP command, see Chapter 5, "TCP/IP services traces and IPCS support," on page 47 and Chapter 42, "Diagnosing storage abends and storage growth," on page 899.

   - **Disabled**
     
     If the loop is disabled, the MVS system console is not available for input. Try the following:
     
     - Use a PSW RESTART to terminate a looping task. This process creates a LOGREC entry with a completion code of X'071'. Use the LOGREC record and the RTM work area to locate the failing module. Depending on the PSW bit 32, the last three bytes (24-bit mode) or four bytes (31-bit mode) contain the address being executed at the time of the dump. Scan the dump output to find the address given in the PSW. For more information on using PSW RESTART, refer to [z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures](#).
     - Take a stand-alone dump. Refer to [z/OS MVS Diagnosis: Tools and Service Aids](#) for information about stand-alone dumps.
2. Get the MVS system console log (SYSLOG), the job log from the started procedure, and the LOGREC output.
   The MVS system console log might contain information, such as error messages, that can help you diagnose the problem. Also, print the LOGREC file.

   Use the LOGDATA option to print the in-core LOGREC buffers. Refer to z/OS MVS Diagnosis: Tools and Service Aids or z/OS MVS IPCS Commands for more information about the LOGDATA option.

   Tip: The SYSERROR data set might contain additional information to help you diagnose the problem.

3. Determine whether there are any messages associated with the loop, such as a particular message always preceding the problem, or the same message being issued repeatedly. If so, add the message IDs to your problem documentation.

4. Examine the trace entries using IPCS.
   By examining all of the trace entries in the system trace table, you might be able to determine whether there is a loop. The most obvious loops would be a module or modules getting continual control of the TCP/IP system.

   Use the PSW to determine the names of the modules in the loop. Refer to the z/OS MVS IPCS User’s Guide for information about using IPCS.

   In the output shown in Figure 2, the CLKC entries indicate an enabled loop. The PSW addresses on the CLKCs identify the looping program. Use the WHERE subcommand to locate the responsible program.

   

   Figure 2. Example of output from the IPCS SYSTRACE command

---

Steps for analyzing hangs

If the problem is a hang, perform the following steps to collect documentation:

1. Determine the extent of the hung state in the operation of the TCP/IP network.
   Determine whether all TCP/IP processing stopped or only processing with respect to a single device, or something in between. Also determine what, if any, recovery action was taken by the operator or user at the time the hang was encountered. Some information about the activity that immediately preceded the hang might be available on the system log or in application program transaction logs.
2. Determine whether TCP/IP responds to commands, such as Ping or Netstat HOME/-h. If TCP/IP does not respond to these commands, take an SVC dump of TCP/IP address space and contact the IBM Software Support Center. If TCP/IP does respond to the commands, it is not hung.

3. Determine whether a particular application (such as z/OS UNIX FTP or a user-written application) is hung.
Take a dump of the OMVS address space, the TCP/IP address space, and the application address space.
Chapter 4. Diagnosing network connectivity problems

This topic describes the diagnosis process for network connectivity problems and contains the following sections:

- “Communicating through the correct stack” on page 30
- “Steps for diagnosing problems connecting to a server” on page 30
- “Steps for verifying server operation” on page 31
- “Steps for verifying IP routing to a destination when not using policy-based routing (PBR)” on page 32
- “Steps for diagnosing problems with IP routing to a destination when using policy-based routing (PBR)” on page 34
- “Steps for verifying network access” on page 36
- “Tools for diagnosing network connectivity problems” on page 37
- “Documentation for the IBM Support Center” on page 43

Overview

Interconnectivity between network hosts encompasses the physical layer or hardware layer, the protocols such as TCP and IP, the IP security services, and the applications that use the services of TCP and IP. To understand interconnectivity, you should first understand internetworking. For detailed information on internetworking, see Appendix B, “Overview of internetworking,” on page 917.

Isolating network problems is an essential step in successful implementation of a network application. This topic introduces commands and techniques you can use to diagnose network connectivity problems.

The following diagnostic commands are available for either the z/OS UNIX environment or the TSO environment:

- Ping
- Netstat
- Traceroute

Netstat reports are also available from the console environment by invoking the DISPLAY TCPIP,NETSTAT command. For complete descriptions of these commands and examples of their output, refer to z/OS Communications Server: IP System Administrator’s Commands

When referring to these commands and their options throughout this section, both the TSO and z/OS UNIX shell command options are listed, separated by a slash. For example, the recommendation to use Netstat to view the stack’s HOME list of IP addresses appears as “use Netstat HOME/-h.”

MVS-style data sets are written in capital letters (for example, hlq.TCPIP.DATA). Files names in the z/OS UNIX file system are written in lowercase (for example, /etc/hosts).
Table 9 lists the name of the commands in each environment.

Table 9. Diagnostic commands

<table>
<thead>
<tr>
<th>UNIX command</th>
<th>TSO command</th>
<th>Refer to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ping/oping</td>
<td>PING</td>
<td>“Using the Ping command” on page 37</td>
</tr>
<tr>
<td>netstat/onetstat</td>
<td>NETSTAT</td>
<td>“Using the Netstat command” on page 41</td>
</tr>
<tr>
<td>traceroute/otracer</td>
<td>TRACERTE</td>
<td>“Using the Traceroute command” on page 43</td>
</tr>
</tbody>
</table>

Guideline: Do not use the resolver and domain name server functions, which translate symbolic names to IP addresses, when diagnosing network problems. Use the host IP address instead.

Communicating through the correct stack

If you are running multiple stacks, the first question to ask is whether the application is communicating through the correct stack. To identify the stack an application is using, you can look at the keyword TCPIPjobname in the TCPIP.DATA file. An application can also select a stack using the SETIBMOPT socket API.

You can use the Netstat parameter TCP/-p to specify the TCP/IP stack name for which you want Netstat report output. This lets you determine the characteristics of a particular stack.

Using the information provided by Netstat, you can change, if necessary, the hlq.PROFILE.TCPIP data set or the application configuration file. Alternatively, the application might need to communicate through another stack.

It is also helpful to understand the search order for configuration information used by z/OS Communications Server. Refer to z/OS Communications Server: IP Configuration Reference, “Understanding search orders of configuration information”, for more information.

Steps for diagnosing problems connecting to a server

Perform the following steps to determine the source of problems connecting to a server.

1. Verify that TCP/IP is running correctly on your host. Use Ping loopback, then Ping one of your home addresses. For information about the Ping command, refer to z/OS Communications Server: IP System Administrator’s Commands.

2. Verify that the server application is operational. See “Steps for verifying server operation” on page 31 for more information.

3. Verify IP routing to the server or the client. If you are not using policy-based routing, see “Steps for verifying IP routing to a destination when not using”
4. Use the DISPLAY TCPIP,,NETSTAT,ACCESS,NETWORK command to determine whether network access has been configured on the TCP/IP stack. Refer to z/OS Communications Server: IP System Administrator’s Commands for more information about this command. If network access control is enabled, then the server might not be permitted to send or receive data on a socket. See “Steps for verifying network access” on page 36 to determine whether network access controls are impacting the server application.

5. Verify IP security protection for the server. If IP security is enabled, then IP traffic to or from the server might not be permitted to flow. See “Steps for diagnosing IP security problems” on page 726 to determine whether IP security controls are impacting the server application.

Steps for verifying server operation

Figure 3 shows the decisions involved for verifying server operation.

Verify Server Operation

Before you begin: Identify the job name and port of the server to be verified.

Perform the following steps to verify server operation.

1. Ensure that the server is started. If not, start the server.
2. Use the Netstat SOCKETS/-s command to determine which port the server is listening on, filtered on the application's job name (-E option for z/OS UNIX, CLIENT keyword for TSO and Operator commands). For example:

```
NETSTAT SOCKETS (CLIENT SMTP
```

If the server is not listening on the correct port, configure it correctly. For basic information about the Netstat SOCKETS/-s command, refer to z/OS Communications Server: IP System Administrator's Commands for details. For details on server configuration, refer to z/OS Communications Server: IP Configuration Reference.

3. Ensure that there is a PORT statement in the TCP/IP profile data set, to reserve the port for the server. If the server is started but not using the correct port, then a PORT statement might be missing. Refer to z/OS Communications Server: IP Configuration Reference for more information about the PORT statement.

4. Use the Netstat SOCKETS/-s command to determine whether a different server is using the port filtered on the port number (-p option for z/OS UNIX, PORT keyword for TSO and Operator commands). Unless the SHAREPORT keyword is specified on the PORT statement, only one server can be listening on a given TCP port. Refer to z/OS Communications Server: IP Configuration Reference for more information about the PORT statement.

5. Check the PORT statement for the server to determine whether the SAF keyword has been specified. If so, then port access control is in effect for the port. Refer to z/OS Communications Server: IP Configuration Guide for more information about port access control. Ensure that the user ID associated with the server is permitted to the security resource name represented by the SAF keyword value. See the description of the PORT statement SAF keyword in the z/OS Communications Server: IP Configuration Reference for information on the security resource name. If the SAF keyword was not specified on the PORT statement, and the server belongs to the z/OS Communications Server product, refer to z/OS Communications Server: IP Configuration Reference for configuration information that is specific to the server.

---

**Steps for verifying IP routing to a destination when not using policy-based routing (PBR)**

Figure 4 on page 33 shows the decisions involved for verifying IP routing to a destination.
Before you begin: Identify the destination IP address for which a route is to be verified.

Perform the following steps to verify IP routing to a destination.

1. Use the Ping command to determine whether there is connectivity to the identified IP address. For information about the Ping command, refer to \textit{z/OS Communications Server: IP System Administrator’s Commands}.

2. If the Ping command fails immediately, there might not be a route to the destination. Use the Netstat ROUTE/-r command to display routes to the network. Verify whether or not TCP/IP has a route to the destination. For information about the Netstat ROUTE/-r command, refer to \textit{z/OS Communications Server: IP System Administrator’s Commands}.
   If there is no route, proceed to step 3. If a route exists, proceed to step 4.

3. If there is no route to the destination, problem resolution depends on whether static or dynamic routing is being used. Refer to \textit{z/OS Communications Server: IP Configuration Guide} for more information about static and dynamic routing.
4. If a route exists, verify that the route is correct for the destination. If multipath routing is in effect for the destination, use the Ping command to determine whether there is connectivity to the IP address over any route. Invoke the Netstat CONFIG/-f command and check the value in the output report field, MultiPath, to determine whether multipath routing is in effect and what kind of multipath routing is active.

Determine whether there is a gateway identified for the route to the destination. If there is no gateway, then the destination address is presumed to be directly connected. In this case, proceed to step 5.

If a gateway is identified for the route, use the Ping command to confirm connectivity to the gateway. Do one of the following:

- If the gateway responds to a Ping, then there is a network problem at the gateway or beyond. Use the Traceroute command with the final destination address to determine at which hop in the route the failure is occurring. For information about using the Traceroute command, refer to z/OS Communications Server: IP System Administrator’s Commands.
- If the gateway does not respond to a Ping, proceed to step 5.

5. Determine which network interface is associated with the route to the destination. If the network interface operation has not been verified for this interface, verify it now. See “Steps for verifying network interface operation” on page 36 for more information.

6. Use the DISPLAY TCPIP,,NETSTAT,ACCESS,NETWORK command to see whether network access control is enabled. If it is enabled, see “Steps for verifying network access” on page 36 for more information.

7. Use the Netstat CONFIG/-f command to determine whether IP security is enabled. If the output report field, IpSecurity, contains the value Yes, then IP security is enabled. See “Steps for verifying IP security and defensive filter operation” on page 732 for information about how to verify that IP security is correctly configured. If the problem still exists, see “Documentation for the IBM Support Center” on page 43 to determine what problem documentation you need, and then call the IBM Support Center.

Steps for diagnosing problems with IP routing to a destination when using policy-based routing (PBR)

Perform the following steps to diagnose problems with IP routing to a destination when using policy-based routing.

1. While the application is active and attempting to connect to the destination, use the Netstat ALL/-A report to determine the policy rule that is assigned to the connection and the route table being used to perform a route lookup.

For information about the Netstat command, see z/OS Communications Server: IP System Administrator’s Commands.

- If no policy rule is listed and the connection is not expected to use policy-based routing, see “Steps for verifying IP routing to a destination when not using policy-based routing (PBR)” on page 32.
- Continue to the following step if one of the following is true:
– A policy rule is not listed and the connection is expected to use policy-based routing
– A policy rule is listed and the connection is not expected to use policy-based routing
– A policy rule is listed, but it is not the expected policy rule
  • Otherwise, continue with step 3.

2. For information on how to map a connection to the correct policy rule, refer to the 'Policy-based routing' section in z/OS Communications Server: IP Configuration Guide.

3. Use pasearch to find the policy rule and the corresponding action. For information about the pasearch command, refer to “Displaying policy based networking information” section of the z/OS Communications Server: IP System Administrator’s Commands. The policy action will list all the possible route tables that can be used for the connection. Perform steps 4 through 6 on each of the route tables listed in the action.

4. Use the Netstat ROUTE/-r PR command to display routes in the route table. Verify whether TCP/IP has a route to the destination/network in the route table. For information about the Netstat ROUTE/-r command, refer to z/OS Communications Server: IP System Administrator’s Commands.
  • If there is no route to the destination/network and no route is expected to be found in the route table, repeat step 4 using the next route table in the policy action.
  • If there is no route to the destination/network and a route was expected in the route table, refer to z/OS Communications Server: IP Configuration Guide for information on setting up static and dynamic routing for policy-based routing tables.
  • If a route was found, verify that the route is marked active (has the U flag). If the route is not active, refer to z/OS Communications Server: IP Configuration Guide for information on route states.
  • If an active route is found, verify that the route table name matches the route table name displayed on the Netstat ALL/-A report for the connection. If it does not, continue to step 9. Otherwise, continue to step 5.

5. Determine whether there is a gateway identified for the route to the destination. If there is no gateway, then the destination address is presumed to be directly connected. In this case, proceed to step 6. If a gateway is identified for the route, use the Ping command to confirm connectivity to the gateway.
  • If the gateway responds to a Ping, then there is a network problem at the gateway or beyond.
  • If the gateway does not respond to a Ping, proceed to step 6.

6. Determine which network interface is associated with the route to the destination. If the network interface operation has not been verified for this interface, verify it now. See “Steps for verifying network interface operation” on page 36 for more information.

7. Use the DISPLAY TCPIP,,NETSTAT,ACCESS,NETWORK command to determine if network access control is enabled. If it is enabled, see “Steps for verifying network access” on page 36 for more information.

8. Use the Netstat CONFIG/-f command to determine if IP security is enabled. If the output report field IpSecurity contains the value Yes, then IP security is enabled. If it is enabled, see “Steps for verifying IP security and defensive filter operation” on page 732 for information about how to verify that IP security is correctly configured.
9. See “Documentation for the IBM Support Center” on page 43 to determine what problem documentation you need, and then call the IBM Support Center.

Steps for verifying network interface operation

Figure 5 shows the decisions involved for verifying network interface operation.

Verify Network Interface Operation

Before you begin: Identify the network interface to be verified.

Perform the following steps to verify network interface operation.

1. Use the Netstat DEVLINKS/-d command to check the interface status. If the interface status is Ready, check the physical connectivity from the interface to the network and check for configuration errors in the network. For example, if you are using VLAN, verify that you have configured the proper VLAN IDs throughout the network. If the interface status is not Ready, try to start the interface by using the VARY TCPIP,START command, and proceed to 2.

2. Use the Netstat DEVLINKS/ -d command again to determine whether the interface is ready after being started. If the interface is not ready, check the system console for error messages issued from TCPIP, VTAM or IOS and respond as suggested in the documentation for the messages that appear.

Steps for verifying network access

Figure 6 on page 37 shows the decisions involved for verifying network access.
Before you begin: Identify the IP address, subnet, or prefix for which network access is to be verified.

Perform the following steps to verify network access.

1. Invoke the DISPLAY TCPIP,,NETSTAT,ACCESS,NETWORK,,ipaddress command, specifying the IP address for which access is to be verified. If the command output indicates that network access control is in effect for the IP address, proceed to 2.

2. Verify that the server or client application is permitted access to the IP resource. See Chapter 12, “Diagnosing access control problems,” on page 407 for more information on verifying this access.

Tools for diagnosing network connectivity problems

This section describes tools used to diagnose network connection problems.

Using the Ping command

The packet Internet groper (Ping) command sends an Internet Control Message Protocol (ICMP/ICMPv6) Echo Request to a host, gateway, or router with the expectation of receiving a reply. You can invoke the Ping function by using the TSO PING command or the z/OS UNIX shell ping or oping command.

For a complete description of the Ping command and examples of Ping output, refer to the z/OS Communications Server: IP System Administrator’s Commands.

The Ping command does not use the ICMP/ICMPv6 header sequence number field (icmp_seq or icmp6_seq) to correlate requests with ICMP/ICMPv6 Echo Replies. Instead, it uses the ICMP/ICMPv6 header identifier field (icmp_id or icmp6_id).
plus an 8-byte TOD time stamp field to correlate requests with replies. The TOD
time stamp is the first 8-bytes of data after the ICMP/ICMPv6 header. When you
specify the Verbose/-v parameter, the ICMP/ICMPv6 header sequence numbers
sent in the ICMP/ICMPv6 echo requests are displayed in the verbose report of
detailed ICMP/ICMPv6 echo replies. Use these sequence numbers to detect the
out-of-order and lost packets, based on missing sequence numbers

When the PMTU/-P parameter with a value of yes or ignore is specified on the
command, Ping will ensure that the outbound echo request packets are not
fragmented. As a result, ICMP/ICMPv6 error messages may be received by the
Ping command if the echo request packet is too large to be sent out by the stack or,
forwarded at some point in the network. In this case the Ping command uses both
the ICMP/ICMPv6 header identifier and sequence number fields to correlate
requests with the error messages. For IPv6 Ping requests, the Ping command will
also use the 8-byte TOD time stamp returned in the ICMPv6 Packet Too Big error
message.

Ping can be used in the following ways:

- **Pinging loopback is essentially used to verify the installation of TCP/IP in the
  z/OS Communications Server environment.**

  The Ping loopback is essentially an internal software test. The command
eamples below use the IPv4 standard loopback address, 127.0.0.1, or the IPv6
standard loopback address, ::1. An IP packet is not sent to a physical device.

  ```
  ping 127.0.0.1
  ```

  For IPv6

  ```
  ping ::1
  ```

- **Ping a home address to verify the information from the Netstat HOME/-h
  command.**

  This is an internal software test. An IP packet is not sent to a physical device.

  ```
  ping 9.67.113.58
  ```

- **Ping a host on a directly attached network to verify the following:**

  - If equal-cost multipath routes exist in the IP routing table for outbound IP
    traffic to reach a remote host, use the Ping INTF/-i option to select a routing
    interface with the attached equal-cost multipath route. Alternatively, for
    routing interfaces associated with an IPv6 link-local address, the name of the
    routing interface can be appended as scope information to the IPv6 link-local
    address of the remote host. When running multiple TCP/IP stacks on the
    same MVS image, specify the TCP/-p parameter, along with the scope, to
    indicate the stack to which the routing interface is configured. Whenever
    applicable, use either of these options to test connectivity. For more
    information about using scope, see the section on support for scope in [z/OS

    - The directly attached network is defined correctly.
    - The device is properly connected to the network.
    - The device is able to send and receive packets on the network.
    - The remote host is able to receive and send packets.

  ```
  ping 9.67.43.101 (intf eth1
  ping fe80::9:67:43:104%ethipv61 -p tcpip1
  ```

- **Ping a host on a remote network to verify the following:**

  - If equal-cost multipath routes exist in the IP routing table for outbound IP
    traffic to reach the remote host, use the Ping INTF/-i option to select a
    routing interface with the attached equal-cost multipath route. Whenever
    applicable, use this option to test connectivity.

  ```
  ```
- The route to the remote network is defined correctly.
- The router is able to forward packets to the remote network.
- The remote host is able to send and receive packets on the network.
- The remote host has a route back to the local host.

```
ping -i eth1 mvs1
```

**Restriction:** Ping commands to a remote host might fail if there is a firewall between the two systems, even if the host is reachable using other commands.

**Display details of echo replies and obtain summary statistics**

You can use the Ping command with the Verbose/-v parameter to obtain detailed echo replies and summary statistics regarding the round trip times based on the response times in the received echo replies. The detailed echo replies can be used to identify lost echo reply packets based on their sequence numbers and to identify how many hops the echo requests have traveled based on their values of time-to-live (TTL) or maximum number of hops (hop limits).

The Verbose/-v parameter provides the following output information:

- Number of bytes for the ICMP data portion
- Echo reply details (for each echo reply received):
  - from: echo reply sender’s IP address
  - seq: ICMP/ICMPv6 sequence number
  - ttl: time-to-live for number of hops (if IPv4)
  - hoplim: maximum hop limit (if IPv6)
  - time: response time
- Ping statistics summary:
  - Packets history:
    - Sent: total number of echo requests sent
    - Received: total number of echo replies received
    - Lost: total number of packets lost plus percentage of packet loss
  - Approximate round trip times in milliseconds:
    - Minimum: minimum value among all RTTs
    - Maximum: maximum value among all RTTs
    - Average: average RTT among all RTTs
    - StdDev for standard deviation among all RTTs

```
ping mvs1 (verbose ping -v mvs1)
```

For examples of Ping verbose reports, see z/OS Communications Server: IP System Administrator’s Commands.

**Determine the path MTU size to a host:**

- Use the Ping PMTU/-P parameter with the values yes or ignore, to prevent fragmentation of the outbound echo request packets and specify what type of path MTU discovery support you want. If the outbound packet needs to be fragmented, Ping will display the host name and IP address of the host where the fragmentation is needed.

  "yes" Specifies that the outbound echo request packets will not be fragmented at the local host or in the network, and that you want to use the MTU value, determined by path MTU discovery for the destination.

  - If path MTU discovery is enabled and has already determined an MTU value for the destination, and the length of the Ping echo request packet is larger than this MTU size, then the local TCP/IP stack will not send out the packet. In this case, Ping displays one of
the local stack’s IP addresses as the address of the host where fragmentation is needed, and the next-hop MTU value displayed by Ping is the current path MTU value to the destination. For Ping commands to IPv4 destinations, the Ping command processing will not cause path MTU discovery support to be triggered for the destination. For IPv4, only TCP processing causes path MTU discovery support to be triggered.

- If path MTU discovery is not enabled, or has not already determined a path MTU value for the destination, and the Ping echo request packet exceeds the configured route MTU selected for this packet, then the local TCP/IP stack will not send out the packet. In this case, Ping will display one of the local stack’s IP addresses as the address of the host where fragmentation is needed, and the next-hop MTU value displayed by Ping is that of the route selected for the Ping packet.

- If the Ping request fails because the echo request packet needs to be fragmented at some point in the network, Ping will display the IP address where fragmentation needs to occur and will display the next-hop MTU value, if it was provided.

**ignore**

Specifies that the outbound echo request packets will not be fragmented at the local host or in the network, and that any MTU values determined by path MTU discovery for the destination, will be ignored.

- If path MTU discovery has determined an MTU value for the destination, and the length of the Ping echo request packet is larger than this MTU size, specifying a value of **ignore** causes the TCP/IP stack to ignore the path MTU value and attempt to send out the packet. As long as the echo request packet length does not exceed the configured route MTU selected for this packet, you can use the ignore value to determine where in the network the original MTU problem occurred. In this case, Ping displays the IP address where fragmentation needs to occur and will display the path MTU value, if it was provided.

- If the Ping echo request packet exceeds the configured route MTU selected for this packet, then the local TCP/IP stack will not send out the packet. In this case, Ping displays one of the local stack’s IP addresses as the address of the host where fragmentation is needed, and the next-hop MTU value displayed by Ping is that of the route selected for the Ping packet.

**MULTIPATH PERPACKET considerations:**

When MULTIPATH PERPACKET is in effect, and equal-cost routes are configured to the Ping destination host, the smallest MTU value of all the equal-cost routes is used as the largest packet size which can be sent, even if some of the equal-cost routes could support a larger packet size.

- Specify the NONAME/-n parameter to request that Ping only display the IP address of the host, and not attempt to resolve the IP address to a host name. This saves a name server address-to-name lookup. If this host also returned the next-hop MTU size, the size is also displayed.

- Vary the length of the outbound packet to determine where the packet needs to be fragmented. The Length/-l parameter on the Ping command, specifies the number of data bytes for the echo request.
- For IPv4 destinations, the total length of the outbound echo request packet includes the length of an IPv4 IP header (20 bytes), the length of an ICMP header (8 bytes), and the data length specified by the Length/-l parameter. Depending on your TCP/IP stack configuration, the TCP/IP stack might add additional IP header options to the IP header created by Ping, before the echo request packet is sent, thereby increasing the size of the packet.

- For IPv6 destinations, the total length of the outbound echo request packet includes the length of an IPv6 IP header (40 bytes), the length of an ICMPv6 header (8 bytes), and the data length specified by the Length/-l parameter. Depending on your TCP/IP stack configuration, the TCP/IP stack might add additional IPv6 extension headers to the packet created by Ping, before the echo request packet is sent, thereby increasing the size of the packet.

### Correcting timeout problems

A Ping timeout message can occur for many reasons, and various techniques can be used to identify whether the problem is the local z/OS server or a remote host or router.

Base your actions on the possible reasons for a timeout, as shown in Table 10.

<table>
<thead>
<tr>
<th>If the problem is ...</th>
<th>Then use these diagnostic techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>The device is not transmitting packets to the local network.</td>
<td>Use <code>Netstat DEVLINKS/-d</code> to collect information to help you diagnose the problem. (See <code>DEVLINKS/-d</code> report option in z/OS Communications Server: IP System Administrator’s Commands.)</td>
</tr>
<tr>
<td>The remote host is not receiving or transmitting packets on the network.</td>
<td>Use <code>Netstat ARP/-R</code> to display the IPv4 entry for the remote host. (See the <code>ARP/-R</code> report option in z/OS Communications Server: IP System Administrator’s Commands.) Use <code>Netstat ND/-n</code> to display the IPv6 entry for the remote host. (See the <code>ND/-n</code> report option in z/OS Communications Server: IP System Administrator’s Commands.)</td>
</tr>
<tr>
<td>The remote host does not have a route back to the local z/OS server.</td>
<td>Use <code>Netstat ROUTE/-r</code> on the remote host to make sure it has a route back. (See <code>ROUTE/-r</code> report option in z/OS Communications Server: IP System Administrator’s Commands.)</td>
</tr>
<tr>
<td>An intermediate router or gateway is not correctly forwarding IP packets.</td>
<td>Use a packet trace. (See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47.)</td>
</tr>
<tr>
<td>The IP reassembly timeout value might be set too low.</td>
<td>Refer to the TCP/IP Profile statements, IPCONFIG and IPCONFIG6, in z/OS Communications Server: IP Configuration Reference.</td>
</tr>
</tbody>
</table>

### Using the Netstat command

You can use the Netstat command to verify your TCP/IP configuration. The information provided in the output from the Netstat command should be checked against the values in your configuration data sets for the TCP/IP stack. Refer to the PROFILE DD statement in the TCP/IP started task procedure for the name of the configuration data sets.
Netstat can be invoked by using the TSO NETSTAT command, the z/OS UNIX shell netstat/onetstat command, or the console DISPLAY TCPIP,,NETSTAT command.

The following Netstat commands can be used to verify the state of those network resources that affect connectivity:

- Netstat HOME/-h
- Netstat DEVLINKS/-d
- Netstat ROUTE/-r
- Netstat ARP/-R
- Netstat ND/-n

For a complete description of the Netstat command and examples of Netstat output, refer to the z/OS Communications Server: IP System Administrator’s Commands.

**Netstat HOME/-h**

Use the Netstat HOME/-h command to verify the IP addresses defined for a TCP/IP stack, the names of the interfaces which are associated with the IP addresses, and the status of the IPv6 IP addresses. If any of the displayed information appears incorrect, check the HOME and INTERFACE statements in the PROFILE.TCPIP data set.

**Netstat DEVLINKS/-d**

Use the Netstat DEVLINKS/-d command to display the status and associated configuration values for a device and its defined interfaces, as coded in the PROFILE.TCPIP data set.

**Netstat ROUTE/-r**

The Netstat ROUTE/-r command displays the current routing tables for TCP/IP. In order to establish connectivity to a remote host, the remote host must also have a route back to the z/OS Communications Server.

The Netstat ROUTE/-r RSTAT command displays all of the static routes that are defined as replaceable.

The Netstat ROUTE/-r PR command displays all of the routes available in policy-based routing tables.

If there are any errors in the policy-based routing tables, check policy agent startup and configuration files for probable errors.

- Ensure that no error messages were generated during processing of either the initial profile or any subsequent VARY TCPIP,OBEYFILE commands. (For information about the VARY TCPIP,OBEYFILE command, refer to z/OS Communications Server: IP System Administrator’s Commands.)
- Check the PROFILE.TCPIP data set for the following:
  - Ensure that the HOME and INTERFACE statements have been coded correctly.
  - If static routing is provided using the BEGINROUTES or GATEWAY statement, ensure that each route in the statement correlates to a valid interface name.
  - If static routing is provided using the BEGINROUTES or GATEWAY statement, ensure that there are routes in the statement that correlate to the appropriate network and host addresses available on the network.
Netstat ARP/-R
Use the command Netstat ARP/-R to query the ARP cache for a given address.
Use Netstat ARP/-R ALL to query an entire ARP cache table. Ensure Netstat
ARP/-R displays an ARP entry for the remote hosts.

The ARP entry for the host on a remote network contains the IP address and the
MAC address for the router.

To ensure the host has a route back to the z/OS Communications Server, review
the routing tables on the remote host. The route back can be a host route or
network route. Intermediate routers must also be configured correctly.

Netstat ND/-n
Use Netstat ND/-n to display the Neighbor Discovery entries.

Using the Traceroute command
Traceroute displays the route that a packet takes to reach the requested target.
Traceroute starts at the first router and uses a series of UDP probe packets with
increasing IP time-to-live (TTL) or hop count values to determine the sequence of
routers that must be traversed to reach the target host. The Traceroute function can
be invoked by either the TSO TRACERTE command or the z/OS UNIX shell
traceroute/otracert command.

The packetSize option lets you increase the IP packet size to see how size affects
the route that the Traceroute packet takes. It also shows the point of failure if a
destination address cannot be reached.

If equal-cost multipath routes exist in the IP routing table for outbound IP traffic to
reach a remote host, use the Traceroute SRCIP/-s option or the INTF/-i option to
select a home IP address (for example, VIPA) for the source IP address and a
routing interface with the attached equal-cost multipath route. Alternatively, for
routing interfaces associated with an IPv6 link-local address, you can append the
name of the routing interface as scope information to the IPv6 link-local address of
the remote host. When running multiple TCP/IP stacks on the same MVS image,
specify the TCP/-a parameter, with the scope, to indicate the stack to which the
routing interface is configured. Whenever applicable, use one of these options to
test connectivity. For more information about using scope, see the information
about support for scope in z/OS Communications Server: IPv6 Network and
Application Design Guide

For the complete syntax of the TSO TRACERTE and z/OS UNIX
traceroute/otracert command and examples of command output, refer to the z/OS
Communications Server: IP System Administrator’s Commands

Using SNMP remote Ping command
Use the SNMP remote Ping command to determine the response time between two
remote hosts. For example, from Host A, you can determine the response time
(Ping) between Hosts B and C, assuming the SNMP agent and TCP/IP subagent
are running on Host B. Refer to the z/OS Communications Server: IP System
Administrator’s Commands for details.

Documentation for the IBM Support Center
In most cases, persistent error conditions indicate an installation or configuration
problem. Contact the local IBM branch office for installation assistance.
If a software defect is suspected, collect the following information before contacting the IBM Support Center:

- **PROFILE.TCPIP**
- **TCPIP.DATA**
- Output from Netstat commands. If using policy-based routing, collect Netstat ROUTE/-r output for all possible route tables involved in the failed routing.
- Output from Ping traces
- If using policy-based routing, output from **pasearch** commands
- Network diagram or layout
- Error messages received. Refer to [z/OS Communications Server: IP Messages Volume 4 (EZ/3, SNM)](http://www.ibm.com/support/docview.wss?uid=ssg1s1002720) for information about messages.
- Component traces, see Chapter 5, “TCP/IP services traces and IPCS support,” on page 47
- If using dynamic routing protocols for IP route table management, see the following for related information:
  - Chapter 32, “Diagnosing OMPROUTE problems,” on page 749
  - Chapter 33, “Diagnosing NCPROUTE problems,” on page 779
Part 2. Traces and control blocks
Chapter 5. TCP/IP services traces and IPCS support

This topic describes selected procedures for TCP/IP Services component trace, packet trace, and Socket API trace. The following sections are included:

- “Component trace”
- “Event trace (SYSTCPIP) for TCP/IP stacks and Telnet” on page 60
- “Packet trace (SYSTCPDA) for TCP/IP stacks” on page 94
- “Data trace (SYSTCPDA) for TCP/IP stacks” on page 148
- “Intrusion Detection Services trace (SYSTCPIS)” on page 149
- “OSAENTA trace (SYSTCPOT)” on page 184
- “Network security services (NSS) server trace (SYSTCPNS)” on page 188
- “Defense Manager daemon (DMD) trace (SYSTCPDM)” on page 188
- “OMPROUTE trace (SYSTCPRT)” on page 188
- “RESOLVER trace (SYSTCPRE)” on page 188
- “Configuration profile trace” on page 188

The TN3270E Telnet server uses a subset of the TCP/IP Services component trace. Specify the started procedure name of Telnet instead of TCP/IP to control component tracing in the Telnet address space.

Component trace

You typically use component trace when re-creating a problem.

Component trace performs the following functions:

- Captures trace requests.
- Adds trace records to an internal buffer.
- Writes the internal buffer to an external writer, if requested.
- Formats the trace records using the Interactive Problem Control System (IPCS) subcommand CTRACE.
- Provides a descriptor at the beginning of a trace record that specifies the address and length of each data area. Each data area in the trace record is dumped separately.
- Provides an optional identifier for the connection (UDP, TCP, and so on) as part of each record.

Tip: Trace data can contain user IDs, passwords, and other sensitive information. The trace data files should be protected to prevent disclosure. As an example, packet trace of the FTP port 21 used to control FTP sessions contains user IDs and passwords in the CLEAR. However, a customer can use Secure Socket Layer for FTP and for TELNET. The Packet Trace (V TCPIP,PKTTRACE) command can be RACF protected.

For detailed information, refer to the following information:

- z/OS MVS Diagnosis: Tools and Service Aids for information about component trace procedures.
- z/OS MVS Initialization and Tuning Reference for information about the component trace SYS1.PARMLIB member.
Modifying options with the TRACE CT command

After initialization, you must use the TRACE CT command to change the component trace options. Modifying options with the TRACE CT command can be done with or without the PARMLIB member. The component trace buffer size can be changed for the SYSTCPDA, SYSTCPIP, SYSTCPIS, and SYSTCPOT components.

Modifying with the PARMLIB member

Because TCP/IP, OMPROUTE, RESOLVER, IKE daemon, NSS server, DMD, and the trace command are accessing the PARMLIB data sets, they need to be authorized for read access to these data sets by RACF or another security product.

To change component trace options using a PARMLIB member, create a new SYS1.PARMLIB member and specify the component member on the PARM= keyword of the TRACE CT command.

Use the following syntax:

```
TRACE CT,ON,COMP=component_name,SUB=(procedure_jobname)
,PARM=parmlib_member
```

Following are descriptions of the parameters:

COMP

Indicates the component name:

SYSTCPDA

TCP/IP packet trace. There is no parmlib member. Options are specified by the VARY TCPIP,,PKTTRACE command. (see "Packet trace (SYSTCPDA) for TCP/IP stacks" on page 94).

SYSTCPDM

Defense Manager daemon, parmlib = CTIDMD00 (see "TCP/IP services component trace for the Defense Manager daemon" on page 719).

SYSTCPIK

IKE daemon, parmlib = CTIKE00 (see "TCP/IP services component trace for the IKE daemon" on page 350).

SYSTCPIP

TCP/IP event trace, parmlib = CTIEZBxx, where xx is any 2 alphanumeric characters (see "Event trace (SYSTCPIP) for TCP/IP stacks and Telnet" on page 60).

SYSTCPIS

TCP/IP intrusion detection service, parmlib = CTIIDSxx (see "Intrusion Detection Services trace (SYSTCPIS)" on page 149).

SYSTCPNS

Network security services server, parmlib = CTINSS00 (see "TCP/IP services component trace for the network security services (NSS) server" on page 368).
SYSTCPOT
TCP/IP OSA-Express Network Traffic Analyzer (OSAENTA) trace.
TCP/IP event trace, parmlib = CTINTA00, (see "OSAENTA trace
(SYSTCPOT)" on page 184). An alternate CTINTA00 member cannot be
specified on the EXEC statement of the TCPIP procedure. CTINTA00
will always be used when starting TCPIP. Only an alternate buffer size
or external writer procedure can be specified. All options are provided
by the OSAENTA command.

SYSTCPRE
Resolver, parmlib = CTIRESxx, (see Chapter 39, “Diagnosing resolver
problems,” on page 853).

SYSTCPRT
OMPROUTE, parmlib = CTIORA00 (see "TCP/IP services component
trace for OMPROUTE” on page 772).

Tip: An optional suffix, CTIORAx, is also available.

SUB
Indicates the started procedure name for TCP/IP, the OMPROUTE application,
the RESOLVER, the IKE daemon started task name, the network security
services (NSS) server started task name, the Defense Manager daemon (DMD)
started task name, or the Telnet started task name for which the trace is run. If
you use the S proname.jobname method to start TCP/IP, OMPROUTE, IKE
deamon, network security services (NSS) server, DMD, or Telnet, you must
specify the same value for the SUB parameter that is specified for the jobname
value. There can be as many as eight TCP/IP sessions and eight Telnet sessions
active in one system.

Restrictions:
• Only one OMPROUTE application can be active on each TCP/IP stack.
• Only one RESOLVER application can be active with each operating system.
• Only one IKE daemon application can be active with each operating system.
• Only one network security services (NSS) server application can be active
with each operating system.
• Only one Defense Manager daemon (DMD) application can be active with
each operating system.

PARM
Identifies the PARMLIB member that contains the trace options (see "COMP"
on page 48). All options can be respecified. However, the buffer size cannot be
changed if OMPROUTE, the IKE daemon, the NSS server, the DMD, or the
RESOLVER are running. If a different size is required, you must stop
OMPROUTE, IKE daemon, network security services server, DMD, or the
RESOLVER, and then restart it after modifying the PARMLIB member.

If the incorrect parmlib member is specified, one of the following messages might
be issued:
• An incorrect CTIEZBxx member is specified on the TRACE CT,ON command:

  IEE5381 CTIEZBxx MEMBER NOT FOUND IN SYS1.PARMLIB
  ITT010I COMPONENT TRACE PROCESSING FAILED FOR PARMLIB MEMBER=CTIEZBxx:
  PARMLIB MEMBER NOT FOUND.

• An incorrect CTIEZBxx member is specified on the CTRACE() keyword of the
EXEC statement of the TCP/IP started procedure:

  IEE5381 CTIEZBxx MEMBER NOT FOUND IN SYS1.PARMLIB

• An incorrect CTIORAx member is specified on the TRACE CT,ON command:
An incorrect CTINTA00 member is specified on the TRACE CT,ON command:

IEE5381 CTINTA00 MEMBER NOT FOUND in SYS1.PARMLIB
ITT01011 COMPONENT TRACE PROCESSING FAILED FOR PARMLIB MEMBER=CTINTA00:
PARMLIB MEMBER NOT FOUND

Modifying without the PARMLIB member
To change component trace options without using a PARMLIB member, issue the TRACE CT command without the PARM= parameter and specify the options on the reply. Though the SYSTCPDA component for packet or data trace does not have a parmlib member, SYSTCPDA can be used on the trace command without the PARMLIB member.

Use the following syntax:
TRACE CT,ON,COMP=component_name,SUB=(procedure_jobname)

After issuing the TRACE CT command, you are prompted to specify the trace options. Respond using the following syntax:

Reply nn
[.ASID=(asid-list)]
[.JOBNAME=(jobname-list)]
[.OPTIONS=(name[name]...)]
[.WTR=(membername|DISCONNECT)]
[.CONT|END]

Restriction: ASID and JOBNAME are not valid for OMPROUTE.

Reply nn
Specifies the identification number (in the range 0-9999) in the prompting message. For example, if the response is
06 IIT066A SPECIFY OPERAND(S) FOR TRACE CT COMMAND

You might reply
r 06,WTR=PTTCP,END

ASID
The ASID (address space identifiers) of the client whose TCP/IP requests are to be traced.

JOBNAME
The job name of the client whose TCP/IP requests are to be traced. The job name might be:

• The job name associated with a client application.
• The SNA LU associated with a TELNET session.

Restriction: Do not use the JOBNAME parameter with the TELNET ctrace option.
• The FTP user ID associated with a FTP data connection.

OPTIONS
Options valid for use with SYSTCPIP are listed in this topic; options valid for use with OMPROUTE are listed in Chapter 32, “Diagnosing OMPROUTE problems,” on page 749 and options for SYSTCPRE (the Resolver component) are listed in Chapter 39, “Diagnosing resolver problems,” on page 853.
Options valid for use with IKE daemon are listed in Chapter 9, “Diagnosing IKE daemon problems,” on page 333.

Options valid for use with the network security services (NSS) server are listed in Chapter 10, “Diagnosing network security services (NSS) server problems,” on page 355.

Options valid for use with the Defense Manager daemon (DMD) are listed in Chapter 30, “Diagnosing Defense Manager daemon problems,” on page 715.

**membername**

The member containing the source JCL that invokes the external writer. The membername in the WTR parameter must match the membername in a previous TRACE CT,WTRSTART command. (See “Steps for obtaining component trace data with an external writer” on page 53.)

**WTR=DISCONNECT**

Disconnects the component trace external writer and the trace. You must also specify a TRACE CT,WTRSTART or TRACE CT,WTRSTOP command to start or stop the writer.

**CONT or END**

CONT specifies that the reply continues on another line. Specify END to complete the response.

**Displaying component trace status**

To display information about the status of the component trace, issue the following command:

DISPLAY TRACE,COMP=component_name,SUB=(procedure_jobname)

See “COMP” on page 48 for more information about component_name.

This command displays information about the status of the component trace for one procedure. To display information about the status of the component trace for all active procedures, issue the following command:

DISPLAY TRACE,COMP=component_name,SUBLEVEL,N=8

For the TCP/IP CTRACE components, do not be misled by the line in the middle of the display showing the MODE is OFF. This part of the display always says the MODE is OFF because TCP/IP uses the subtrace for all tracing. The subtrace for TCPCS2 indicates the actual state of the trace. In the example shown in Figure 4 on page 54, the trace is active (MODE is ON) with an internal buffer size of 16 M, tracing all ASIDs and all JOBNAMES, using MINIMUM options, and using the external writer PTTCP. Another version of the DISPLAY TRACE command D TRACE,COMP=component_name,SUBLEVEL,N=8 shows all subtraces for the component.

**Modifying the trace buffer size:** To modify the amount of trace buffer in use for the SYSTCPIP, SYSTCPDA, SYSTCPIS and SYSTCPOT traces use the following command:

TRACE CT,nnnM,COMP=component_name,SUB=(procedure_jobname)

where nnnM is the new buffer size in mega bytes. The buffer size is subject to the minimum and maximum buffer size established for each component.

See “COMP” on page 48 for more information about component_name.
Stopping a component trace

To stop current tracing, issue the following TRACE CT command:

```
TRACE CT, OFF, COMP=component_name, SUB=(procedure_jobname)
```

See **“COMP” on page 48** for more information about component_name.

With the TRACE,CT,OFF command, TCP/IP discontinues recording of all trace data.

```
TRACE CT, ON, COMP=SYSTCPIP, SUB=(procedure_jobname)
```

TCP/IP continues to trace exception events.

Obtaining component trace data with a dump

You can request a dump to obtain component trace data for:

- TCP/IP stack
- OMPROUTE
- RESOLVER
- TELNET
- IKE daemon
- Network security services (NSS) server
- Defense Manager daemon (DMD)

TCP/IP stack: If an abend occurs in the TCP/IP address space or in a user’s address space, TCP/IP recovery dumps the home ASID, primary ASID, secondary ASID, and the TCPIPDS1 data space. TCPIPDS1 is the name of the data space for each TCP/IP in an MVS image. It contains the trace data for the SYSTCPIP, SYSTCPDA, SYSTCPIS and SYSTCPOT components.

To view the trace records for a problem where no abend has occurred, use the DUMP command. The following example illustrates a DUMP command:

```
DUMP COMM=(your dump title here)
R n, JOBNAME=(tcpipprocname), DSPNAME=('tcpipprocname'.TCPIPDS1), CONT
R n, SDATA=(NUC, CSA, LSQA, PSA, RGN, SQA, TRT), END
```

**Figure 7. Example of DUMP command for TCP/IP stack**

To generate a meaningful dump, specify (at a minimum):

- CSA
- LSQA
- RGN
- SQA

OMPROUTE: To obtain a dump of the OMPROUTE address space (which contains the trace table), use the DUMP command, as shown in the following example:

```
DUMP COMM=(enter your dump title here)
R n, JOBNAME=omproute_started_task_name, SDATA=(CSA, RGN, ALLPSA, SQA, SUM, TRT, ALLNUC), END
```

**Figure 8. Example of DUMP command for OMPROUTE**
RESOLVER: To obtain a dump of the RESOLVER, use the DUMP command, as shown in the following example:

DUMP COMM={enter your dump title here}
R n, JOBNAME=resolver_started_task_name, SDATA=(CSA, RGN, ALLPSA, SQA, SUM, TRT, ALLNUC), END

Figure 9. Example of DUMP command for RESOLVER

TELNET: To obtain a dump of TELNET, use the DUMP command, as shown in the following example:

DUMP COMM={enter your dump title here}
R n, JOBNAME=telnet_started_task_name, SDATA=(CSA, RGN, ALLPSA, SQA, SUM, TRT, ALLNUC), END

Figure 10. Example of DUMP command for TELNET

Steps for obtaining component trace data with an external writer
Perform the following steps to use an external writer to obtain component trace data for TCP/IP stacks, packet trace, OMPROUTE, and Telnet.

1. Enter the appropriate writer procedure in SYS1.PROCLIB, as shown in the following example. Use a separate external writer for each CTRACE component. You can have multiple procedures writing to as many as 16 TRCOUT files either on disk or tape.

   //PTTCP PROC
   //* REFER: SYS1.PROCLIB(PTTCP)
   //* COMPID: OPER
   //* DOC: THIS PROCEDURE IS THE IPCS CTRACE1 EXTERNAL WRITER PROCEDURE.
   //* USED BY TCP/IP.
   //* //IEFPROC EXEC PGM=ITTTRCWR,REGION=0K,TIME=1440
   //* TIME=1440 to prevent S322 abends
   //TRCOUT01 DD DSNAME=MEGA.IPCS.CTRACE1,UNIT=SYSDA,
   // VOL=SER=STORGE,
   // SPACE=(4096,(100,10),,,CONTIG),DISP=(NEW,CATLG),
   // DCB=(DSORG=PS)
   //
   // Restriction: Do not specify the DCB parameters RECFM, LRECL and BLKSIZE. The external writer defaults to an optimal blocking factor.

2. Start the external writer using the following command:

   TRACE CT, WTRSTART=procedure_name, WRAP

3. Turn the trace on and connect the external writer to the component either by specifying the external writer name in the PARMLIB member, or by specifying the external writer name in the TRACE command. When starting TCP/IP, because the SYSTCPDA component has no PARMLIB member, the PARMLIB option is not applicable for SYSTCPDA. For example, TRACE CT, ON, COMP=SYSTCPDA, SUB=(TCPCS), PARM=CTIEZBDA is a valid command. The PARMLIB member can specify a new buffer size or the name or a writer. To turn the trace on and connect the external writer to the component using a PARMLIB member, add the following TRACEOPTS option to the PARMLIB member:

   WTR(xxx)
where xxx is the procedure name of the external writer. Then use this
PARMLIB member when starting the program (TCP/IP, OMPROUTE,
TELNET, or the Resolver) or if the program is already executing, issue the
following command:

TRACE CT,ON,COMP=component_name,SUB=(procedure_name),PARM=parmlib_member

To turn the trace on and connect the external writer without using the
PARMLIB member, enter the following command:

TRACE CT,ON,COMP=component_name,SUB=(procedure_name)

When the system responds, enter the following command:

R n,WTR=procedure_name,END

where n is the response number issued by the system. Note that you can add
options to the response. The options vary for each component name. See
“Formatting component traces” on page 55 for references to the component
options.

4. Use the DISPLAY command to check the external writer status. Include a
sublevel.

D TRACE,COMP=SYSTCPDA,SUB=(TCPCS2)

IEE843I 11.33.06 TRACE DISPLAY 099
SYSTEM STATUS INFORMATION
ST=(ON,0064K,00064K) AS=ON BR=OFF EX=ON MT=(ON,064K)
TRACENAME
========
SYSTCPDA

MODE BUFFER HEAD SUBS
OFF HEAD 2
NO HEAD OPTIONS
SUBTRACE MODE BUFFER HEAD SUBS
---------------------------------------------
TCPPCS2 ON 0016M
ASIDS *NONE*
JOBNAMES *NONE*
OPTIONS MINIMUM
WRITER PTTCP

**Tip:** At this point, the external writer is active for packet and data.

5. Turn the trace off or disconnect the external writer. The following two
commands disconnect from the external writer, while leaving the trace running
internally.

TRACE CT,ON,COMP=component_name,SUB=(procedure_jobname)

When the system responds, enter the second command:

R nn,WTR=DISCONNECT,END

6. Stop the external writer using the following command:

TRACE CT,WTRSTOP=procedure_name

**Tips for using component trace external writer**
Consider the following when using the component trace external writer.
Do not use the same writer to trace more than one TCP/IP stack, TELNET, or OMPROUTE application. If you need to trace multiple stacks or applications, use separate writers.

If your external writer fills up and the wrap option is on, the writer overwrites itself. If the nowrap option is on, the writer stops.

Use REGION=0K on the trace writer procedure EXEC statement. This helps ensure that there is enough virtual memory for trace buffers.

Use TIME=1440 on the EXEC statement. This prevents S322 abends.

Use CONTIG on the disk space allocation of the trace data when using the WRAP option. For example: SPACE=(1024,(4096,100),,CONTIG). This ensures that the space for the trace data set is available.

Do not specify DCB parameters for trace data sets. The writer optimizes the logical record length and block size for new trace data sets.

Ensure that the dispatching priority of the writer is equal to or greater than the application that is being traced.

Using a VSAM linear data set:
1. Define a VSAM Linear data set. Using a VSAM linear data set for output trace data provides better performance than using a sequential data set does.
   ```
   //DEFINE EXEC PGM=IDCAMS
   //SYSPRINT DD SYSOUT=*  
   //SYSIN DD *  
   DELETE +  
     (hlq.CTRACE.LINEAR) +  
   CLUSTER  
   DEFINE CLUSTER(  
     NAME(hlq.CTRACE.LINEAR) +  
     LINEAR +  
     MEGABYTES(10) +  
     VOLUME(CPDLB0) +  
     CONTROLINTERVALSIZE(32768) +  
   ) +  
   DATA(  
     NAME(hlq.CTRACE.DATA) +  
   )  
   LISTCAT ENT(hlq.CTRACE.LINEAR) ALL
   ```
2. Update the Ctrace writer procedure:
   ```
   //IEFPROC EXEC PGM=ITTTRCWR
   //TRCDOUT01 DD DSNAMEn=hlq.CTRACE.LINEAR,,DISP=SHR 
   //SYSPRINT DD SYSOUT=*
   ```
3. Issue the COPYTRC command.
   The VSAM data set must be copied with COPYTRC to a sequential dataset before being sent to IBM Service.

Formatting component traces
You can format component trace records using IPCS panels or a combination of IPCS panels and the CTRACE command, either from a dump or from external-writer files. The code for the component trace record formater can be found in the SYS1.MIGLIB data set. This data set should be added as a concatenation to the STEPLIB data set. For details, refer to the z/OS MVS IPCS Commands and the z/OS MVS IPCS User's Guide.

Steps for formatting component traces using IPCS panels: To format component traces using only IPCS panels, follow these steps:
1. Log on to TSO.
2. Access IPCS.

3. Select option 2 (ANALYSIS) from the option list.

4. Select option 7 (TRACES) from the option list.

5. Select option 1 (CTRACE) from the option list.

6. Select option D (Display) from the option list.

You know you are done when the CTRACE DISPLAY PARAMETERS screen is displayed [Figure II], as shown below.

![Figure II. IPCS CTRACE](image)

Enter the component name in the COMPONENT field and as the value in COMP(xx). For descriptions of options, see the following sections:

- SYSTCPDA, see “COMP” on page 48.
- SYSTCPDM, see “TCP/IP services component trace for the Defense Manager daemon” on page 719.
- SYSTCPIK, see “TCP/IP services component trace for the IKE daemon” on page 350.
- SYSTCPNS, see “TCP/IP services component trace for the network security services (NSS) server” on page 368.
- SYSTCPIP, see “COMP” on page 48.
- SYSTCPI, see “COMP” on page 48.
- SYSTCPOT, see “OSAENTA trace (SYSTCPOT)” on page 184.
- SYSTCPRE, see Chapter 39, “Diagnosing resolver problems,” on page 853.
- SYSTCPR, see “TCP/IP services component trace for OMPROUTE” on page 772.
Steps for using the CTRACE command: Perform the following steps to format component traces using the CTRACE command.

1. Log on to TSO.

2. Access IPCS.

3. Select option 6 (COMMAND) from the option list.

4. Enter a CTRACE command and options on the IPCS command line.

Syntax: Following is the syntax of the IPCS CTRACE command:

**CTRACE syntax**

```
CTRACE Component selection Report type

Data selection Address space selection Setdef parameters
```

**Component Selection:**

```
QUERY(compname)-SUB((name))
COMP(compname)-SYSNAME(name)-SUB((name))
```

**Report Type:**

```
SHORT GMT SUMMARY LOCAL FULL TALLY
```

**Data Selection:**

```
START(mm/dd/yy, hh.mm.ss.ddddd) STOP(mm/dd/yy, hh.mm.ss.ddddd) EXCEPTION
LIMIT(nnnnnnnn) ENTIDLIST(entidlist) USEREXIT(exitname)
OPTIONS((component routine parameters))
```
Address Space Selection:

- ALL
- CURRENT
- ERROR
- TCPEERROR
- ASIDLIST(asidlist)
- JOBLIST(joblist)
- JOBNAME(joblist)

Setdef Parameters:

- DSNAME(dataset)
- DATASET(dataset)
- DDNAME(ddname)
- FILE(ddname)
- FLAG(severity)
- PRINT
- NOPRINT
- TERMINAL
- NOTERMINAL
- TEST
- NOTEST

Parameters: Refer to [z/OS MVS IPCS Commands](#) for details on the CTRACE parameters.

Keywords: You can use the following CTRACE keywords with TCP/IP component trace formatters:

**JOBLIST, JOBNAME**

Use the JOBLIST and JOBNAME keywords to select packet trace records with a matching link name. However, only the first eight characters of the link name are matched and no asterisks are accepted in the job name. Also, use them to match the job name in data trace records.

**ASIDLIST**

Use the ASIDLIST to select trace records only for a particular address space.

**GMT**

The time stamps are converted to GMT time.

**LOCAL**

The time stamps are converted to LOCAL time.

**SHORT**

If the OPTIONS string does not specify any reports, then format the trace records. Equivalent to the FORMAT option.

**FULL**

If the OPTIONS string does not specify any reports, then format and dump the trace records. Equivalent to the FORMAT and DUMP options.

**SUMMARY**

If the OPTIONS string does not specify any reports, then create a one line summary for each trace record. Equivalent to the SUMMARY option.

**TALLY**

If the OPTIONS string does not specify any reports, then count the trace records.
**START and STOP**

These keywords limit the trace records that are seen by the formatter. The STOP keyword determines the time when records are no longer seen by the packet trace report formatter.

**Rule:** CTRACE always uses the time the trace record was moved to the buffer for START and STOP times.

**LIMIT**

Determines the number of records the formatter is allowed to process.

**USEREXIT**

The CTRACE USEREXIT is called for TCP/IP formatters, except for the packet trace formatters. Therefore, the packet trace formatter calls the CTRACE USEREXIT before testing the records with the filtering criteria. If it returns a nonzero return code, then the record is skipped. The USEREXIT can also be used in the OPTIONS string. It is called after the record has met all the filtering criteria in the OPTIONS string. For details, see "Formatting packet traces using IPCS" on page 97.

**Examples of formatting component traces:** The following example shows the error message when the specified address space is not available in the dump.

```c
CTRACE QUERY(SYSCTCPIP) SUB((TCPSVT1)) FULL LOCAL
COMPONENT TRACE QUERY SUMMARY

ITT10003I There are no trace buffers in the dump for COMP(SYSCTCPIP)SUB((TCPSVT1))
```

The following example shows the results when the CTRACE QUERY command is issued for a dump when the address space is available.

```c
CTRACE QUERY(SYSCTCPIP) SUB((TCPSVT2)) FULL LOCAL
COMPONENT TRACE QUERY SUMMARY

COMP(SYSCTCPIP)SUBNAME((TCPSVT2))

START = MT 02/21/2001 15:25:49.432 LOCAL
STOP = 180 02/21/2001 15:51:16.8
Buffer size: 0050M

OPTIONS: CONFIG,CSOCKET, FIREWALL, IOCTL, MESSAGE, OETCP, OPCMDS, OPMGS, PASAPI, PING, SOCKAPI, TN, UDP, XCF, CLAW, INT
ERNET, LCS, VTAM, VTAMDATA

OPTIONS: MINIMUM
```

**Tip:** The first option is the relevant one (ignore the second options list). The buffer size and options list are displayed only for a dump data set, not an external writer data set.

**Formatting component traces using a batch job:** A component trace can also be formatted through the use of a batch job. The following is an example of JCL for a batch job:

```c
//jobname DD (accounting),pgmname,CLASS=A,MSGCLASS=A
//DUMP EXEC PGM=IKJEFT01
//STEP LIB DD DISP=SHR,DSN=SYS1.MIGLIB
//SYSPRINT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSTSPRT DD SYSOUT=* 
//PRINTER DD SYSOUT=* 
//SYSPROC DD DISP=SHR,DSN=SYS1.CLIST
// DD DISP=SHR,DSN=SYS1.SBLSCLI0
//IPCSPPARM DD DISP=SHR,DSN=SYS1.PARMLIB
```
IKE daemon trace (SYSTCPIK)

TCP/IP Services component trace is also available for use with the IKE daemon. See "TCP/IP services component trace for the IKE daemon" on page 350.

Event trace (SYSTCPIP) for TCP/IP stacks and Telnet

The TN3270E Telnet server running as its own procedure also uses the SYSTCPIP event trace.

Restrictions: All discussion that follows where TCP/IP is used as an example also pertains to the TN3270E Telnet server with the following exceptions:

- The TN3270E Telnet server does not use a dataspace for trace collection, it uses its own private storage.
- A subset of trace commands are used by Telnet. A default parmlib member, CTIEZBTN, is provided that indicates all trace options available. The default parmlib member can be overridden in the same manner as the TCP/IP parmlib can be overridden.
- A subset of IPCS commands are used by Telnet.

Event trace for TCP/IP stacks traces individual TCP/IP components (such as STORAGE, INTERNET, and so forth) and writes the information either to a data set (using an external writer), or internally to the TCP/IP dataspace (TCPIPDS1). To aid in first failure data capture, a minimal component trace is always started during TCP/IP initialization if you use the TCP/IP Component Trace SYS1.PARMLIB member, CTIEZBxx.

You can select trace records at run time by any of the following methods:

- JOBNAME
- Address space identifiers (ASID)
- Trace option
- IP address
- Port number
- Event identifier
Restriction: If using the TELNET options, do not specify the JOBNAME parm when starting CTRACE.

Specifying trace options
You can specify component trace options at TCP/IP initialization or after TCP/IP has initialized.

Specifying trace options at initialization
To start TCP/IP with a specific trace member, use the following command:

```
S tcpip_procedure_name,PARMS=CTRACE(CTIEZBxx)
```

where CTIEZBxx is the component trace SYS1.PARMLIB member.

You can create this member yourself, or you can update the default SYS1.PARMLIB member, CTIEZB00. For a description of trace options available in the CTIEZB00, see [Table 11 on page 64](#).

Tip: Besides specifying the desired TCP/IP traces, you can also change the component trace buffer size.

You can use IBM Health Checker for z/OS to check whether TCP/IP Event Trace (SYSTCPIP) is active with options other than the default options (MINIMUM, INIT, OPCMDS, or OPMSGS). For more details about IBM Health Checker for z/OS, see [Appendix D, “IBM Health Checker for z/OS,” on page 951](#).
TRACEDATA

ON OR OFF: PICK 1

ON

OFF

BUFSIZE: A VALUE IN RANGE 1M TO 256M

BUFSIZE(8M)

JOBNAME(jobname1,...)

ASID(Asid1,...)

WTR(wtr Procedure)

Note, the following groups of trace options are supported:

ALL = All options except MISC, PFMAIN, ROUTE, SERIAL,

SOCKAPI, STORAGE, TCPMAIN, and TIMER

CSOCKET = PF S + SOCKET

Figure 12. SYS1.PARMLIB member CTIEZB00 (Part 1 of 3)
Figure 12. SYS1.PARMLIB member CTCIEZB00 (Part 2 of 3)
A group activates multiple trace options. The group name identifies traces that should be activated for a specific problem area, and trace groups provide a way to collect trace data by problem type.

Table 11 describes the available trace options and groups.

<table>
<thead>
<tr>
<th>Trace Event</th>
<th>Description</th>
</tr>
</thead>
</table>
| ALL         | All types of records except MISC, PFSMIN, ROUTE, SERIAL, STORAGE, TCPMIN, and TIMER.  
**Slow Performance:** Using this option slows performance considerably, so use with caution.  
Also available for the TN3270E Telnet server running in its own address space. |
<table>
<thead>
<tr>
<th>Trace Event</th>
<th>Description</th>
</tr>
</thead>
</table>
| ALLMIN      | Turns on the following trace options:  
              • INIT  
              • OPCMDS  
              • OPMGS  
              • PFSMIN  
              • TCPMIN  
| ACCESS      | Trace creation, modification, and manipulation of the Network Access tree, along with results of all Network Access queries.  
| AFP         | Turns on trace for fast response cache accelerator.  
| ARP         | Shows address resolution protocol (ARP) cache management and ARP timer management. This option also shows all outbound and inbound ARP packets.  
            | **Tip:** The information provided differs depending on the type of device.  
            | **Guideline:** The ARP and ND options are aliases. If you turn one on, you turn on the other option, and if you turn one off, you turn off the other option. When formatting the trace, these options can be filtered separately.  
| CLAW        | Shows all control flows for a CLAW device.  
| CONFIG      | Turns on trace for configuration updates.  
| CSOCKET     | Turns on the following trace options:  
              • PFS  
              • SOCKET  
| DLC         | Turns on the following trace options:  
              • CLAW  
              • INTERNET  
              • LCS  
              • VTAM  
              • VTAMDATA  
| EID(list)   | Turns on trace by event identifier. The event identifiers are 8 hexadecimal digits. Up to 16 can be specified. Use only under the direction of IBM Support.  
| ENGINE      | Turns on trace for stream head management.  
            | **Guideline:** The ENGINE and QUEUE options are aliases. If you turn one on, you turn on all related options, and if you turn one off, you turn off all related options. These alias options are only for recording the trace. When formatting the trace, these options can be filtered separately.  
| FIREWALL    | Turns on trace for firewall events.  
            | **Tip:** Synonymous with IPSEC option.  
| ICMP        | Turns on trace for the ICMP protocol.  

<table>
<thead>
<tr>
<th>Trace Event</th>
<th>Description</th>
</tr>
</thead>
</table>
| IN          | Turns on the following trace options:  
|             | • CONFIG  
|             | • INIT  
|             | • IOCTL  
|             | • OPCMDS  
|             | • OPMGS  
| INIT        | Turns on trace for TCP/IP Initialization/Termination.  
|             | **Note:** The INIT, OPCMDS, and OPMGS options are aliases. If you turn one on, you turn on all related options, and if you turn one off, you turn off all related options. These alias options are only for recording the trace. When formatting the trace, these options can be filtered separately.  
|             | Also available for the TN3270E Telnet server running in its own address space.  
| INTERNET    | Turns on trace for Internet Protocol layer.  
|             | **Tip:** Using this option slows performance considerably, so use with caution.  
| IOCTL       | Turns on trace for IOCTL processing.  
| IPADDR(list) | Turns on trace by IP address.  
| IPSEC       | Turns on trace for IP security events.  
|             | **Tip:** Synonymous with FIREWALL option.  
| LATCH       | Turns on the following trace option:  
|             | • SERIAL  
| LCS         | Shows all control flows for an LCS device.  
| MESSAGE     | Turns on trace for message triple management.  
|             | **Tip:** Using this option slows performance considerably, so use with caution.  
|             | Also available for the TN3270E Telnet server running in its own address space.  
| MINIMUM     | Turns on the following trace options:  
|             | • INIT  
|             | • OPCMDS  
|             | • OPMGS  
| MISC        | Turns on trace for miscellaneous TCP/IP internal diagnostics.  
| NONE        | Turn off all traces but exception traces, which always stay on.  
|             | Also available for the TN3270E Telnet server running in its own address space.  
| ND          | Enable Neighbor Discovery trace option.  
|             | **Guideline:** The ARP and ND options are aliases. If you turn one on, you turn on the other option, and if you turn one off, you turn off the other option. When formatting the trace, these options can be filtered separately.  

---

66 z/OS V1R11.0 Comm Svr: IP Diagnosis Guide
Table 11. Trace options and groups  (continued)

<table>
<thead>
<tr>
<th>Trace Event</th>
<th>Description</th>
</tr>
</thead>
</table>
| OETCP       | Turns on the following trace options:  
|             | • ENGINE  
|             | • PFS  
|             | • QUEUE  
|             | • TCP  |
| OEUDP       | Turns on the following trace options:  
|             | • ENGINE  
|             | • PFS  
|             | • QUEUE  
|             | • UDP  |
| OPCMDS      | Turns on traces of operator commands.  
|             | **Guideline:** The INIT, OPCMDS, and OPMGS options are aliases. If you turn one on, you turn on all related options, and if you turn one off, you turn off all related options. These alias options are only for recording the trace. When formatting the trace, these options can be filtered separately. |
| OPMGS       | Turns on trace for console messages.  
|             | **Guideline:** The INIT, OPCMDS, and OPMGS options are aliases. If you turn one on, you turn on all related options, and if you turn one off, you turn off all related options. These alias options are only for recording the trace. When formatting the trace, these options can be filtered separately. |
| PASAPI      | Turns on traces for transforms that handle Pascal APIs. |
| PFS         | Turns on trace for the physical file system layer.  
|             | **Tip:** The PFS and PFSMIN options should not be specified together; the PFS option gathers all the information that the PFSMIN option gathers. |
| PFSMIN      | Turns on the minimum PFS trace option.  
|             | **Tip:** The PFS and PFSMIN options should not be specified together; the PFS option gathers all the information that the PFSMIN option gathers. |
| PING        | Turns on the following trace options:  
|             | • ARP  
|             | • ICMP  
|             | • RAW  |
| POLICY      | Trace the stack usage of Policy Rules and Actions. |
| PORT(list)  | Turns on trace by port number. |
| QUEUE       | Turns on trace for stream queue management.  
<p>|             | <strong>Guideline:</strong> The ENGINE and QUEUE options are aliases. If you turn one on, you turn on all related options, and if you turn one off, you turn off all related options. These alias options are only for recording the trace. When formatting the trace, these options can be filtered separately. |
| RAW         | Turns on trace for the RAW transport protocol. |
| ROUTE       | Trace manipulation of IP Routing Tree. |</p>
<table>
<thead>
<tr>
<th>Trace Event</th>
<th>Description</th>
</tr>
</thead>
</table>
| RW          | Turns on the following trace options:  
|             | • ENGINE  
|             | • PFS  
|             | • QUEUE  
|             | • RAW  
|             | • SOCKET  
| SERIAL      | Turns on trace for lock obtain and release.  
|             | Tip: Using this option slows performance considerably, so use with caution.  
|             | Also available for the TN3270E Telnet server running in its own address space.  
| SMTP        | Turns on the following trace options:  
|             | • ENGINE  
|             | • IOCTL  
|             | • PASAPI  
|             | • PFS  
|             | • QUEUE  
|             | • SOCKET  
|             | • TCP  
| SNMP        | Turns on trace for SNMP SET requests.  
| SOCKAPI     | Trace Macro and Call Instruction API calls (see "Socket API traces" on page 74)  
| SOCKET      | Turns on trace for the Sockets API layer.  
| STORAGE     | Turns on trace for storage obtain and release.  
|             | Tip: Using this option slows performance considerably, so use with caution.  
|             | Also available for the TN3270E Telnet server running in its own address space.  
| SYSTEM      | Turns on the following trace options:  
|             | • INIT  
|             | • OPCMDS  
|             | • OPMMSGS  
|             | • SERIAL  
|             | • STORAGE  
|             | • TIMER  
|             | • WORKUNIT  
| TC          | Turns on the following trace options:  
|             | • ENGINE  
|             | • PFS  
|             | • QUEUE  
|             | • SOCKET  
|             | • TCP  

Table 11. Trace options and groups (continued)
<table>
<thead>
<tr>
<th>Trace Event</th>
<th>Description</th>
</tr>
</thead>
</table>
| TCP        | Turns on trace for the TCP transport protocol.  
**Restriction:** The TCP and TCPMIN options should not be specified together; the TCP option gathers all the information that the TCPMIN option gathers.  
**Slow Performance:** Using this option slows performance considerably, so use with caution. |
| TCPMIN     | Turns on the minimum TCP trace option.  
**Slow Performance:** The TCP and TCPMIN options should not be specified together; the TCP option gathers all the information that the TCPMIN option gathers. The same is also true for the PFS and PFSMIN options. |
| TELNET     | Turns on trace for TELNET events.  
Only useful when used by the TN3270E Telnet server. |
| TELNVTAM (an alias for TELNET) | Turns on trace for TELNET events. |
| TIMER      | Turns on trace for TCP timers.  
**Slow Performance:** Using this option slows performance considerably, so use with caution.  
Also available for the TN3270E Telnet server running in its own address space. |
| TN         | Turns on the following trace options for TCP:  
- PFS  
- TCP  
Turns on the following trace option for the TN3270E Telnet server running in its own address space:  
- TELNET |
| UD         | Turns on the following trace options:  
- ENGINE  
- PFS  
- QUEUE  
- SOCKET  
- UDP |
| UDP        | Turns on trace for UDP transport protocol.  
**Slow Performance:** Using this option slows performance considerably, so use with caution. |
| VTAM       | Shows all of the non-data-path signaling occurring between IF and VTAM. |
| VTAMDATA   | Shows data-path signaling between IF and VTAM, including a snapshot of media headers and some data.  
**Slow Performance:** Using this option slows performance considerably, so use with caution. |
| WORKUNIT   | Turns on trace for work unit scheduling. |
| XCF        | Turns on trace for XCF events. |
Specifying trace options after initialization

After TCP/IP or Telnet initialization, you must use the TRACE CT command to change the component trace options. Each time a new component trace is initiated, all prior trace options are turned OFF, and the new traces are activated.

You can specify TRACE CT with or without the PARMLIB member.

You can use IBM Health Checker for z/OS to check whether TCP/IP Event Trace (SYSTCPIP) is active with options other than the default options (MINIMUM, INIT, OPCMD, or OPMSGS). For more details about IBM Health Checker for z/OS, see Appendix D, “IBM Health Checker for z/OS,” on page 951.

Additional filters for SYSTCPIP

The following additional trace filters for limiting the volume of trace data are available:

- The IPADD keyword filters by IP address
- The PORT keyword filters by port number
- The EID keyword filters by event identifier

The EID keyword specifies up to 16 trace event identifiers. Each identifier is eight hexadecimal characters. For example: EID(00010001,00090001,40030003). Use the EID keyword only with the direction of IBM service personnel.

To execute a trace on a particular IP address, use the IP address, port number, ASID, and JOBNAME as targets for filtering the records.

To use this function, start by issuing the TRACE command:

```
TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpip_procedure_name)
R 01,OPTIONS=(IPADD(12AB:0:0:CD30::/60),PORT(1012))
R 02,OPTIONS=(ENGINE,PFS),END
```

Trace records of type ENGINE or PFS for an IP address of 12AB:0:0:CD30::/60 and a port number of 1012 are captured. The IP address used is the foreign session partner IP address. The local port number is the local session partner port number. The choice of the IP and Port numbers is determined by the direction of the data.

When filters are used, the trace record must be accepted by each filter. Each filter can specify multiple values (up to 16), and the trace record must match one of the values.

Table 12 lists the data types and corresponding description.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound</td>
<td>Data received at the IP layer is considered inbound data. The source IP address and the destination port number are used.</td>
</tr>
<tr>
<td>Outbound</td>
<td>Data sent in the PFS layer is considered outbound data. The destination IP address and the source port number are used.</td>
</tr>
</tbody>
</table>

The following are five criteria for selecting trace records for recording:

- TYPE
- JOBNAME
Each criterion can specify one or more values. If a criterion has been specified, the record to be traced must match one of the values for that criterion. If a criterion has not been specified, the record is not checked and does not prevent the record from being recorded. However, the record must match all specified criteria.

In the above example, JOBNAME and ASID were not specified, so the value of JOBNAME and ASID in the record are not checked.

Restriction: IPADDR and PORT are exceptions. Some trace records do have an IP address or a port number. Therefore, the IP address is only checked if it is nonzero, and the port number is checked only if it is nonzero.

You can also specify a range of IP addresses to trace. For example,

```
TRACE CT,ON,COMP=SYSTCPIP,SUB=(TCP1P_PROC_NAME)
  R xx, OPTIONS = (IPADDR(nn.nn.nn.nn,{nn.nn.nn.nn/mm.mm.mm.mm}),PORT(pppp{,pppp}))
```

**IPADDR**

An IP address. Up to 16 addresses can be specified. IPv4 addresses are in dotted decimal notation, for example: 192.48.24.57. IPv6 addresses are in colon-hexadecimal notation or in a combination of both colon-hexadecimal and dotted decimal for IPv4-mapped IPv6 addresses, for example: beef::030:1839.

Use an IP address of 0 for trace records that do not have an IP address. A subnet mask is indicated by a slash (/) followed by the prefix length in decimal or by a dotted decimal subnet mask for IPv4 addresses. The prefix length is the number of one bits in the mask. For IPv4 addresses it might be in the range of 1–32; for IPv6 addresses it might be in the range of 1–128, for example: 192.48.24/24 or 2001:0DB8::0/10, respectively.

**PORT**

The list of port numbers to be filtered. Up to 16 port numbers can be specified.

The port numbers, specified in decimal, must be in the range 0–65535. A trace record with a zero port number is not subject to port number filtering.

You can specify the IPADDR and PORT keywords multiple times in an OPTIONS string. If you do, all the values are saved.

Restriction: All the values in the OPTIONS keyword must be specified in one trace command. The next trace command with an OPTIONS keyword replaces all the options specified.

### Formatting event trace records for TCP/IP stacks and Telnet

You can format event trace records using IPCS panels or a combination of IPCS panels and the CTRACE command. For a description of the relevant IPCS panels, see “Steps for formatting component traces using IPCS panels” on page 55.

For more information about other CTRACE options, refer to the z/OS MVS IPCS Commands.

When using an IPCS panel, enter the trace types in the following format:

```
option DUCB() CID()
```
Following is the syntax for the CTRACE command for TCP/IP stacks and Telnet. For more information on the command and IPCS, refer to the z/OS MVS IPCS User’s Guide.

**Type Name**
The name of a trace type. Only records of these types are formatted. For a list of types, see Table 11 on page 64.

**ADDR**
A control block address. Up to 16 control block addresses can be specified. Addresses in hexadecimal should be entered as x’hhhhhhhhh’.

**DUCB**
A process index for the thread of execution. Up to 16 indexes can be specified. The DUCB index values can be entered either in decimal (such as DUCB(18)) or hexadecimal (such as DUCB(X’12’)), but are displayed in hexadecimal format.

**CID**
A connection identifier. Up to 16 identifiers can be specified. The CID values can be entered in either decimal (such as CID(182)) or hexadecimal (such as CID(X’0006CE7E’)), but are displayed in hexadecimal. This is the same value that appears in the NETSTAT connections display.

**IPADDR**
An IP address. Up to 16 addresses can be specified. IPv4 addresses are in dotted decimal notation, for example: 192.48.24.57. IPv6 addresses are in colon-hexadecimal notation or in a combination of both colon-hexadecimal and dotted decimal for IPv4-mapped IPv6 addresses, for example: beef::c030:1839. Use an IP address of 0 for trace records that do not have an IP address. A subnet mask is indicated by a slash (/) followed by the prefix length in decimal or by a dotted decimal subnet mask for IPv4 addresses. The prefix
length is the number of one bits in the mask. For IPv4 addresses it might be in the range of 1–32; for IPv6 addresses it might be in the range of 1–128, for example: 192.48.24/24 or 2001:0DB8:0/10

**PORT**
A port number. Up to 16 port numbers can be specified. Note that the port numbers can be entered in decimal, such as PORT(53), or hexadecimal, such as PORT(x'35'), but are displayed in decimal. These are port numbers in the range 0–65535. Use a port number of 0 for trace records that do not have a port number.

**RECORD**
The record number can be specified as a single hexadecimal value (for example, x'hhhhhhhh') or as a range (for example, x'hhhhhhhh':x'hhhhhhhh'). The record number is assigned as the records are written and can be found on the line of equal signs (=) that separates each record.

Standard TSO syntax is used for the keywords and their values. For example, CID (1 2 3).

Figure 13 shows the beginning of the CTRACE formatted output. The CTRACE command parameters are followed by the trace date and column headings. Then, there is one TCP/IP CTRACE record with four data areas.

```
COMPONENT TRACE FULL FORMAT
COMP(SYSTCPIP)SUBNAME((TCPSVT))

**** 11/03/1999
SYSTYPE  MNAME   ENTRY ID  TIME STAMP  DESCRIPTION
--------  ---------  --------  -----------  ---------------
1  VIC1A2  PFS   60010018  14:57:59.207826  Socket IOCTL Exit
   HASID..001E  PASID..000E  SASID..001E  USER...OMPROUTE
2  TCB....007E9A68  MODID..EZBPF1OC  REG14..161D86C0  DUCB...0000000C
3  CID....0000003A  PORT.....0
   IPADDR. 3F98::D002:A521
4  IPADDR. 3F98::0002:4521
```

Figure 13. Start of component trace full format

The parts of the TCP/IP CTRACE record are:
Standard IPCS header line, which includes the system name (VIC142), TCP/IP option name (PFS), time stamp, and record description.

TCP/IP header line with address space and user (or job name) information.

TCP/IP header line with task and module information.

TCP/IP header line with session information (CID, IP address, and port number).

TCP/IP header line for a data area. This line has the address (first four bytes are the ALET), the length of data traced, and the data description. Following the description, the actual data is in dump format (hexadecimal offset, hexadecimal data, and EBCDIC data).

There are four data areas in this example. The third data area (Return Value Errno ErrnoJr) has an extra line. The ERRNO line is added only when the return value is -1 and the ERRNO indicates an error. In this example, the return code is hexadecimal 462 (decimal 1122). Refer to the z/OS Communications Server: IP and SNA Codes for more information.

TCP/IP trailer and separator line with the record sequence number (hexadecimal 573E).

Additional fields in CTRACE output

The ERRNO line in Figure 13 on page 73 is one of two cases in which the formatter extracts data and formats it in a special way. The other case is for "TCB CTRL" and "IUDR" data. Several fields are copied from the data and formatted with character interpretation of fields, such as converting values to decimal or dotted decimal. Figure 14 is an example. Note the additional fields (TcpState, TpiState, and others) following the hexadecimal data.

Figure 14. Component trace full format showing character interpretation of fields

Socket API traces

The SOCKAPI option, for the TCP/IP CTRACE component SYSTCPIP, is intended to be used for application programmers to debug problems in their applications.
The SOCKAPI option captures trace information related to the socket API calls that
an application might issue. The SOCKET option is primarily intended for use by
TCP/IP Service and provides information meant to be used to debug problems in
the TCP/IP socket layer, UNIX System Services, or the TCP/IP stack.

CTRACE is available only to users with console operator access. If the application
programmer does not have console access, someone must provide the CTRACE
data to the programmer. For security reasons, it is suggested that only the trace
data related to the particular application be provided. The following sections
explain how to obtain the trace data for a particular application, format it, and
save the formatted output. The application data can be isolated when recording the
trace, or when formatting it, or both.

z/OS provides several socket APIs that applications can use. Figure 15 on page 75
shows different APIs along with the high level flows of how they interact with the
TCP/IP stack.

The SOCKAPI trace output is captured in the Sockets Extended Assembler Macro
API (the Macro API). Given the structure of the TCP/IP APIs, this trace also covers
the Call Instruction API, the CICS Socket API, and the IMS socket API. Some of the
socket APIs based on the Macro API currently encapsulate some of the Macro API
processing.

For example, in a CICS TS environment, CICS sockets-enabled transactions do not
have to issue an SOCKAPI call. Rather, this is done automatically for the socket
API by the TCP/IP CICS TRUE (Task Related User Exit) component layer. If the
socket API trace is active, trace records for the SOCKAPI calls are created.

**Figure 15. TCP/IP networking API relationship on z/OS**

### Recommended options for the application trace

The CTRACE facility has flexibility such as filtering, combining multiple
concurrent applications and traces, and using an external writer.

**Guidelines:** Consider the following when using CTRACE:
- Although the CTRACE can be used to trace multiple applications at the same time and in conjunction with other trace options, it is not recommended. Multiple traces make problem determination more difficult.
- For performance reasons, the data being recorded should be filtered, to minimize the overhead of recording the trace, to make formatting faster, to save storage, and to minimize wrapping (overwriting of older trace records by new trace records).

Ideally, you should use the CTRACE facility to capture all the SOCKAPI trace records for one application. The trace can be filtered various ways when formatting. If necessary, you can limit the trace data collected by IP address or port number, but you risk some records not being captured. For example, the problem might be that the wrong IP address or port number was coded or used. Both the IP address and port number are formatting options.

**Guidelines:** The following are recommended options for optimally capturing the application data:

- **Trace only one application.** Use the job name or ASID option when capturing the trace to limit the trace data to one application.
- **Trace only the SOCKAPI option.** To get the maximum number of SOCKAPI trace records, specify only the SOCKAPI option.
  **Tip:** You also receive exception records. Exception records are always traced because they are considered unusual events.
- **Use an external writer.** The external writer is recommended to:
  - Separate the SOCKAPI trace records from other internal data that exist in a dump (for security and other reasons)
  - Avoid interrupting processing with a dump of the trace data
  - Keep the buffer size from limiting the amount of trace data
  - Avoid increasing the buffer size, which requires restarting TCP/IP
  - Handle a large number of trace records
- **Trace only one TCP/IP stack.** If you are running with multiple TCP/IP stacks on a single z/OS image, use the external writer for only one TCP/IP stack.
- **Activate the data trace only if more data is required.** The SOCKAPI trace contains the first 96 bytes of data sent or received, which is usually sufficient. If additional data is needed, the data trace records can be correlated with the SOCKAPI records.

**Collecting the SOCKAPI trace option**
This section describes how to collect the trace for use by application programmers.

The existing CTRACE facility for TCP/IP’s SYSTCPIP component is used for the SOCKAPI trace option. Collecting the trace is described generally in “Component trace” on page 47.

The trace can be started automatically when TCP/IP starts or can be started or modified while TCP/IP is executing. A CTRACE PARMLIB member is required for starting the trace automatically, and can optionally be used after TCP/IP has been started.

**CTRACE PARMLIB member CTIEZBxx:** Sample member CTIEZB00 is shipped with TCP/IP.
**TCP/IP start procedure:** The CTRACE PARMLIB member can be specified in the TCP/IP start procedure or on the START command. The sample TCPIPROC start procedure specifies member name CTIEZB00. Specifying the member name on the START command depends on how the TCP/IP start procedure is coded.

The following example illustrates overriding the PARMLIB member name using the sample TCPIPROC start procedure.

```
S TCPIPROC,PARM='CTRACE(CTIEZBAN)' 
```

Use the TRC option to specify the suffix of the SYS1.PARMLIB member for SYSTCPIP CTRACE initialization. The TRC option appends the two letters to CTIEZB. The full member name is CTIEZBxx. The default value is 00. In this example, the PARMLIB member for SYSTCPIP is CTIEZBAN, an equivalent command is

```
S TCPIPROC,PARM='TRC=AN'
```

Use the IDS option to specify the suffix of the SYS1.PARMLIB member for SYSTCPIP CTRACE initialization. The IDS option appends the two letters to CTIIDS. The full member name is CTIIDSxx. The default value is 00.

```
S TCPIPROC,PARM='IDS=AN'
```

You can specify multiple parameters. If you specify both the CTRACE and TRC parameters, the parameter that appears last in the parameter string is used.

**TRACE command:** Use the MVS TRACE command to start, modify, or stop the trace after TCP/IP has been started. The TRACE command replaces all prior settings except the buffer size. When modifying the options, be sure to specify the SOCKAPI option.

The examples below show how to start the trace.

The SUB option is the subtrace name, which for TCP/IP, is the job name of the stack (usually this is the TCP/IP start procedure name). In the following examples, the subtrace is TCPIPROC (the name of the sample procedure), and the variable fields are in lowercase.

To activate the trace with just the SOCKAPI option, code the following:

```
TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpiproc)
R n,JOBNNAME=(ezasokjs),OPTIONS=(sockapi),end
```

To specify a PARMLIB member, which contains the trace options, code the following:

```
TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpiproc),PARM=ctiezban
```

To stop the trace, either use the TRACE CT,OFF command or reissue TRACE CT,ON with different parameters.

The following is an example of the OFF option:

```
TRACE CT,OFF,COMP=SYSTCPIP,SUB=(tcpiproc)
```

When using the TRACE command, be sure to notice message ITT038I, which indicates whether the command was successful or not. The following is an example of ITT038I:

```
Chapter 5. TCP/IP services traces and IPCS support  77
```
14.11.29 ITT038I NONE OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE SUCCESSFULLY EXECUTED.

or

14.11.40 ITT038I ALL OF THE TRANSACTIONS REQUESTED VIA THE TRACE CT COMMAND WERE SUCCESSFULLY EXECUTED.

Refer to [z/OS MVS System Commands](#) for more information about the TRACE command.

**External writer:** If the trace is active, it is always written to an internal buffer (whose size is set to BUFSIZE during TCP/IP initialization). The internal buffer is available only in a dump of TCP/IP and its dataspace (TCPIPDS1). Optionally, the trace can also be written to an external data set using the MVS CTRACE external writer. If you use an external writer, the trace records are copied to a data set.

To use an external writer, you must create a procedure that specifies the job to run (the external writer) and the trace output data sets. Also, refer to [z/OS MVS Diagnosis: Tools and Service Aids](#) for more information about CTRACE, the external writer (including a sample procedure), dispatching priority for the external writer job, and wrapping.

The external writer must be started before the trace can be activated. The trace must be inactivated before the writer can be stopped. The writer must be stopped before the data set can be formatted or transferred. For example, here is a sequence of commands for using an external writer procedure named ctw:

```
TRACE CT,WTRSTART=ctw
TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpiproc)
   R n,JOBNAME=(ezasokjs),OPTIONS=(sockapi),WTR=ctw,end

<run application being traced>

TRACE CT,OFF,COMP=SYSTCPIP,SUB=(tcpiproc)
TRACE CT,WTRSTOP=ctw
```

The external data set (specified in the procedure "ctw") is now available for formatting.

**Filtering options when recording the trace:** Options for filtering include the following:

**Component**
- Required - SYSTCPIP for SOCKAPI.

**Subtrace**
- Required - TCP/IP stack name.

**Trace option**
- Highly recommended to limit the tracing to the SOCKAPI option. You can also filter on this option when formatting the trace.

**Jobname**
- Highly recommended for socket applications to limit the trace to one application. You can also filter on this option when formatting the trace.

**ASID**
- Highly recommended as an alternative to the job name if the application has already started running (otherwise, the ASID is unknown). You can also filter on this option when formatting the trace.

**IP address**
- Recommended only for certain scenarios (see discussion below). The IP address is a filtering option when formatting the trace.
Port  Recommended only for certain scenarios (see discussion below). The port
number is a filtering option when formatting the trace.

If trace data for multiple applications is collected in the same data set or in a
dump, the trace output should be filtered so that application programmers see
only the data for their applications for security reasons.

Use the IP address and Port options to filter the trace, both when collecting the
trace and when formatting the trace. Generally, it is best to collect all the
application records to avoid having to re-create the problem. After the records are
collected, you can filter the records various ways when formatting the trace.

An example scenario in which you would only want to collect records for one IP
address is if there is a problem with a particular remote client, and the local
application has many clients. If you tried to record the trace records for all clients,
there could be a lot of data and the trace could wrap, thus overwriting older
records. Note that if you specify an IP address when collecting the trace, the trace
records with no IP address are also collected. So you get all the records for the
problem client, and some other client records.

An example scenario, in which you would only want to collect records for one port
number, is if there is a problem with a server on one port. If you specify a port
number when collecting the trace, the trace records with no port number are also
collected. You get all the records for the problem server application, and some
other applications’ records.

IP address/port filtering, when specified, has a varying effect depending on the
type of socket call being traced. Table 13 describes the effect of IP address/port
filtering for the different types of socket API calls. The Yes or No specified in
columns 2 and 3 indicates whether local port filtering and remote IP address
filtering can be activated for the socket calls in column 1. Yes means that if a filter
is set, only the calls matching that filter are collected. No means that whether or
not a filter is specified, all the calls are collected (no filtering is done).

**Table 13. IP address and port filtering effect on different types of socket API calls**

<table>
<thead>
<tr>
<th>Socket call</th>
<th>Filtering active?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Local port</td>
</tr>
<tr>
<td>ACCEPT</td>
<td>Yes</td>
</tr>
<tr>
<td>BIND</td>
<td>Yes/No (2)</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Yes/No (3)</td>
</tr>
</tbody>
</table>
Table 13. IP address and port filtering effect on different types of socket API calls (continued)

<table>
<thead>
<tr>
<th>Socket call</th>
<th>Filtering active?</th>
<th>Local port</th>
<th>Remote IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANCEL</td>
<td></td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>FREEADDRINFO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETADDRINFO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETCLIENTID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETHOSTBYADDR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETHOSTBYNAME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETHOSTID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETHOSTNAME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETNAMEINFO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INITAPI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_RECVFROM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_RECVMSG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELECTEX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SENDMSG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SENDTO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOCKET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAKESOCKET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TERMINAPI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LISTEN</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CLOSE</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GETPEERNAME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETSOCKNAME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETSOCKOPT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIVESOCKET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNCTL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOCTL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHUTDOWN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETSOCKOPT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITEV</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where Yes is indicated in Table 13 on page 79, the assumption is made that the information necessary for the filtering option is available. For example, if a SEND is issued on a socket that is not bound or not connected, no filtering takes place. In addition, the following describe some of the special considerations for the different socket calls in the previous table.
1. Even though the remote IP address is available after an ACCEPT call, it is not used for filtering the exit ACCEPT trace record. This is done to avoid confusion where the entry trace record for ACCEPT would not be filtered, but the exit trace record would.

2. Assumess a BIND issued for a nonzero port. If a BIND is issued for port 0 (meaning an ephemeral port is assigned by TCP/IP), no filtering takes place for this BIND call.

3. If the socket is bound at the time of the CONNECT, local port filtering is honored. Otherwise, the CONNECT is not subject to local port filtering.

**Monitoring the trace:** Use the MVS command DISPLAY TRACE to check the trace options currently in effect. The following is an example of a console showing the display command and the resulting output (the line numbers were added for discussion reference).

```
1. 14.27.14 D TRACE,COMP=SYSTCPIP,SUB=(tcpiproc)
2. 14.27.14 IEE8431 14.27.14 TRACE DISPLAY
3. SYSTEM STATUS INFORMATION
4. ST=(ON,0064K,00064K) AS=ON BR=OFF EX=ON MT=(ON,064K)
5. TRACENAME
6. =========
7. SYSTCPIP
8. 
9. MODE BUFFER HEAD SUBS
10. ===============
11. OFF HEAD 1
12. SUBTRACE MODE BUFFER HEAD SUBS
13. ________________________________________________
14. TCPIPROC ON 0008M
15. ASIDS *NONE*
16. JOBNAME EZASOKJS
17. OPTIONS SOCKAPI
18. WRITER CTW
```

For component SYSTCPIP, do not be misled by line 10 in the example. It always says the trace is off because TCP/IP uses the subtrace for all tracing. The subtrace TCPIPROC on line 14 indicates the actual state of the trace. In this example, the trace is active (ON) with an internal buffer size of eight megabytes and only the SOCKAPI option is active. Only one application (EZASOKJS) is being traced and the trace is being written to an external writer.

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The MVS DISPLAY TRACE command. For more information on this command, see <a href="http://www.ibm.com">z/OS MVS System Commands</a>.</td>
</tr>
<tr>
<td>2–4</td>
<td>These are explained in the <a href="http://www.ibm.com">z/OS MVS System Messages, Vol 1 (ABA-AOM)</a> for IEE843I.</td>
</tr>
<tr>
<td>5–7</td>
<td>Show that this is the CTRACE component SYSTCPIP.</td>
</tr>
<tr>
<td>8–11</td>
<td>These are not applicable for TCP/IP because TCP/IP uses only the subtrace facility of the MVS CTRACE service. Instead of activating a global trace, the trace options are specified for each stack individually. Thus, there can be multiple TCP/IP stacks with different CTRACE options. Note however that line 10 is useful — it shows that there is one subtrace (meaning one TCP/IP stack is active).</td>
</tr>
<tr>
<td>14</td>
<td>Shows the &quot;subtrace&quot; name is the TCP/IP procedure name (TCPIPROC in this example), whether the trace is active (MODE=ON), and the buffer size is eight megabytes. The buffer size is the number of bytes in the data space that is used for recording the trace.</td>
</tr>
</tbody>
</table>
Show the ASID and JOBNAME filtering values. If any ASIDs or JOBNAMEs are listed, only those trace entries matching the ASID or JOBNAME are collected. "ASIDS *NONE*" indicates that all address spaces are being traced (there is no filtering).

Shows the specific options that are active, as specified in the TRACE command or in the CTIEZBxx PARMLIB member. If port or IP address filtering were active, they would appear on this line.

Shows the external writer is inactive. If the writer is active, the writer procedure name is shown instead of *NONE*.

Capturing the trace: If you use only the internal buffer, you must obtain a dump with the TCP/IP data space (TCPIPDS1) in order to view the CTRACE records. It is usually a good idea to also capture the application address space. For example, using the MVS DUMP command, type the following commands. Be sure to specify the TCP/IP data space (TCPIPDS1) because that is where the CTRACE data is located.

Tip: The SDATA options specified are appended to other options.

The SDATA options shown here are the generally recommended options.

Tip: The SDATA options specified are appended to other options.

DUMP COMM=(Sample dump for SOCKAPI)
R n, JOBNAME=(tcpiproc,ezasokjs), DSPNAME=('tcpiproc'.TCPPIPDS1), CON T
R n, SDATA=(ALLNUC, CSA, LPA, LSQA, RG N, SWA, SQA, TRT), CON T
R n, END

Notes:
1. You can type the first three commands in advance, and you can then just type the fourth command at the correct moment to capture the events.
2. If you use the external writer, [“External writer” on page 78](#) explains how to capture the trace in a data set.

**Formatting the SOCKAPI trace option**

Use the IPCS CTRACE command to format the trace, both for a dump and for an external writer. Interactively, you can either type the CTRACE command on the IPCS Command panel or you can use the panel interface. IPCS is also available in batch. Whichever interface you choose, for TCP/IP we recommend using the CTRACE QUERY command to find out what subtraces are contained in the data set. For example, the command CTRACE QUERY(SYSTCPIP) SHORT produced the following output:

```
COMPONENT TRACE QUERY SUMMARY

COMPONENT SUB NAME
-------------- 
0001. SYSTCPIP TCPSVT 
0002. SYSTCPIP TCPSVT3 
0003. SYSTCPIP TCPSVT1 
0004. SYSTCPIP TCPSVT2
```

There are several filters available that can help to limit the amount of data formatted. In addition to the CTRACE options (start and stop time, and such) provided by IPCS, there are some options specifically for TCP/IP:

**DUCB** Not applicable for SOCKAPI. (DUCB is an internal TCP/IP token.)

**CID (connection identifier)**

Not applicable for SOCKAPI.
IPADDR
Use for SOCKAPI. Specify the IPv4 addresses in dotted decimal format, with an optional prefix value (1 to 32) or a subnet mask in dotted decimal form. Specify the IPv6 address in colon-hexadecimal notation (or in a combination of colon-hexadecimal and dotted decimal for IPv4-mapped IPv6 addresses), with an optional prefix value (1 to 128). Several socket calls do not use an IP address. To see the trace records without an IP address (or with an IP address of all zeros), specify zero for one of the IPADDR values. For example, IPADDR(0,9.67.113/24) formats all CTRACE records with an IP address of 000.000.000.000 and formats all CTRACE records with an IP address of 009.067.113.*, where * is any number from 0 to 255.

PORT
Use for SOCKAPI. Specify the port number in decimal. Several socket calls do not have an associated port number, such as INITAPI and SOCKET. To see the trace records without a port (or with a port of 0), specify zero for one of the port values. For example, PORT(0,389,1925).

You can save the formatted output to the IPCSPRNT data set.

If the formatted output does not contain the records you expect:
- In a dump, you can check the options specified when recording the trace by using the TCPIPCS TRACE command to display the TCP/IP CTRACE filtering options in effect. This also indicates whether any records were lost. See Chapter 6, “IPCS subcommands for TCP/IP,” on page 191 for more information on the TCPIPCS TRACE command.
- For either a dump or an external writer data set, use the CTRACE QUERY command to see what tracing was in effect (subtrace name, start and stop times). For a dump, this command also shows the buffer size and options. For example, the command CTRACE QUERY(SYSTCP/IP) SUB((TCP/IPROC)) FULL produced the following output for a dump:

```
COMPONENT TRACE QUERY SUMMARY
  COMP(SYSTCP/IP) SUBNAME((TCP/IPROC))
  START = 01/10/2000 19:49:21.234490 GMT
  STOP = 01/10/2000 19:51:51.360653
  Buffer size: 0256K
  OPTIONS: ACCESS,OPCMDS,OPMSGS,QUEUE,ROUTE,INIT,SOCKAPI,SOCKET
  OPTIONS: MINIMUM

For TCP/IP, the first line of "options" (showing ACCESS) is the applicable one. This shows the options as specified on the command line or in the CTIEZBxx PARMLIB member.
```

Refer to the z/OS MVS IPCS User's Guide for more information about CTRACE formatting. Refer to z/OS MVS IPCS Commands for more information about the CTRACE command.

**Reading and interpreting the SOCKAPI trace option**
The SOCKAPI trace records trace the input and output parameters for most of the API calls. The API calls not traced are GETIBMOPT, TASK, GLOBAL, NTOP, PTON, and any API calls that fail before the trace point is reached. (An API call fails if module EZBSOH03 cannot be located, if EZBSOH03 is unable to obtain storage, and so on.) In addition to tracing API calls, trace records are created for a few special situations (Default INITAPI and Unsolicited Event exit being driven).
For API calls, there is an Entry record describing the input parameters, and an Exit record describing the output parameters (with some input parameters repeated for clarification). For asynchronous calls, there is also an Async Complete (Asynchronous Complete) record (see “Examples of SOCKAPI trace records” on page 85).

The following examples include:
- A SOCKAPI trace record
- Trace records for asynchronous applications
- Resolver API calls
- External IOCTL commands
- API Call with an IOV parameter
- Default INITAPI
- Default TERMAPI
- SELECT
- SELECTEX
- Token error
- Unsolicited event exit

A SOCKAPI trace record: A typical SOCKAPI record is shown below. This example is a READ Entry.

The lines are numbered for discussion reference only. The description for each line is for the example shown. Lines 1-5 are the separator and header lines that exist for all SOCKAPI trace records. Lines 6-7 are optional header lines.

The parameters for the specific call follow the header lines. For Entry records, the input parameters are shown. For Exit and Asynchronous Complete records, the output parameters are shown and some input parameters might also be shown for reference. Parameters are only formatted if they were specified in the call (optional parameters not supplied are not formatted). The parameters are listed in a specific order for consistency. The parameter names are the same as the names in the z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference with a few exceptions; for example, S is formatted as SOCKET. The parameter name, value, and address are shown on one line if the value fits. Numeric parameter values are in decimal unless followed by a lowercase x indicating hexadecimal. Whenever possible, the values are interpreted (such as ERRNO) for reference.

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>This separator line shows the previous SYSTCPIP component trace record number in hexadecimal.</td>
</tr>
</tbody>
</table>
The first data line has the host name (MVS026), trace option (SOCKAPI), trace code (60050042), time, and trace record name.

The home, primary, and secondary ASIDs are always the same value (application’s ASID) for the SOCKAPI trace option. The job name is also shown.

The MVS TCB address is shown. TIE (Task Interface Element) is the value of the TASK parameter on the EZASMI macro. The TIE is described in the z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference. The parameter list address and DUCB are shown. Multiple concurrent calls can use the TIE; if so, they must have a different PLIST. The key is the 4-bit storage key from the PSW.

The ADSNAME (from the INITAPI call) is formatted in EBCDIC. The subtask name (from the INITAPI call) is formatted in EBCDIC if possible; otherwise, it is formatted in hexadecimal. The token is an eight-byte value, which identifies the INITAPI call instance.

If applicable, the ports and IP addresses are shown. The ports are formatted in decimal; the IP addresses are in dotted decimal.

The REQAREA parameter is shown because it was specified by the application. This is the 4-byte token presented to the application’s exit when the response to the function request is complete. At the far right, the address in the application program of the REQAREA parameter is shown.

The SOCKET parameter is formatted in decimal. Its address is also shown.

The NBYTE parameter (number of bytes to be read) is formatted in decimal, followed by its address.

The ALET parameter is formatted in hexadecimal, followed by its address.

The BUF parameter currently has no data (because no data has been read) but its address is shown. In the READ Exit (or READ Async Complete) record, if the call was successful, the first 96 bytes of the data are also shown.

Examples of SOCKAPI trace records: This section includes descriptions and examples of the SOCKAPI trace records.

- “Successful API Call”
- “API call fails synchronously” on page 86
- “API call fails synchronously with parameter not addressable” on page 86
- “API call fails synchronously with diagnostic reason code” on page 87
- “Resolver API calls” on page 87
- “External IOCTL commands” on page 89
- “API call with an IOV parameter” on page 90
- “Default INITAPI” on page 90
- “Default TERMAPI” on page 90
- “SELECT” on page 90
- “SELECTEX” on page 91
- “Token Error” on page 92
- “Unsolicited event exit” on page 92

Successful API Call: For asynchronous APIs, the Exit record merely indicates whether or not the call was acceptable. The contents of general purpose register 15
are displayed to indicate this. The Asynchronous Complete record shows the actual results of the call. In addition to the output parameters, several interesting values are traced, including the contents of general purpose register 0, the pointer to the asynchronous exit routine, the token passed to the asynchronous exit, the key in which the asynchronous exit was invoked, and the authorization state in which the exit is invoked. These values are not parameters on the GETHOSTID call, so their addresses are not shown. In this example, note also that the return code is formatted in dotted decimal and the meaning of the return code is provided.

**Note:** The API call might actually complete synchronously, in which case the Async Complete trace record might appear in the trace prior to the Exit record.

---

**API call fails synchronously:** An asynchronous API call might fail synchronously or asynchronously. In this example, the WRITE call error was detected in the synchronous processing, so general purpose register 15 has a nonzero value. The ERRNO value is interpreted (in this case, the NBYTE parameter on the WRITE call had a value of zero, which is not acceptable).

**Note:** The ERRNO value is the TCP/IP Sockets Extended Return Code. Refer to [z/OS Communications Server: IP and SNA Codes](https://www.ibm.com/support/knowledgecenter/en/SSP5T7/COMM/doc/comm_suapi.htm) for information about TCP/IP Sockets Extended Return Codes.

---

**API call fails synchronously with parameter not addressable:** If a parameter specified in the API call is not addressable by TCP/IP when creating the SOCKAPI record, the string (** PARAMETER NOT ADDRESSABLE **) is shown instead of the parameter value. The parameter address is shown at the far right, as usual.
API call fails synchronously with diagnostic reason code: If the API call does not complete successfully, the return code, ERRNO value (in decimal and interpreted), and possibly a diagnostic reason code are shown. The first two bytes of the diagnostic reason code are a qualifier (IBM internal use only). The last two bytes of the diagnostic reason codes are the UNIX ERRNOJR values described in the z/OS Communications Server: IP and SNA Codes

Resolver API calls: The GETHOSTBYADDR and GETHOSTBYNAME IPv4 Resolver API calls use the HOSTENT structure described in the calls in the z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference. As shown in the following GETHOSTBYADDR Exit trace example, the HOSTENT address is shown on one line, and the contents of the HOSTENT structure are described on separate lines. There can be multiple aliases and host addresses; each one is listed separately. In this example, there are two aliases.

The GETADDRINFO for IPv4 or IPv6 Resolver API shows the call is requesting the IP address for the host (node) name MVS150. No service name is provided. GETADDRINFO exit shows the hostname was resolved to the IPv4 address 9.67.113.117. These fields are described in the Macro and CALL section in the z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference.
The **FREEADDRINFO** for IPv4 or IPv6 Resolver API call displays the **RES** (ADDRINFO) structure that is freed. This field is in the Macro and CALL section in the [z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference](https://www.redbooks.ibm.com/redbooks/p59565/).
GETNAMEINFO Exit shows the IP address was resolved to the name loop6int.resdns.ibm.com and no service name was found for port 1031 (hence the service name is the input port number). These fields are in the Macro and CALL section in the IBM Communications Server: IP Sockets Application Programming Interface Guide and Reference.

External IOCTL commands: For external IOCTL commands, the command name is interpreted. For IBM internal-use-only commands, the hexadecimal value of the command is shown. The input and output for each command can differ. In this example, the SIOCGIFCONF command requests the network interface configuration. The exit record shows the call was successful (the return code is zero) and the network interface configuration is shown.
PORT... 0 IPADDR.... 9.67.113.58
FAMILY.. 2 (AF_INET) RESERVED.. 0000000000000000x
RETCODE..: 0 Addr..00068EB4

API call with an IOV parameter: The IOV parameter is an array of structures used on the READV, RECVMSG, SENDMSG, and WRITEV API calls. Each structure contains three words: the buffer address, the ALET, and the buffer length. Each IOV entry is shown on one line. When there is data available (READV Exit, RECVMSG Exit, SENDMSG Entry, and WRITEV Entry), some of the buffer data is also displayed. A maximum of 96 bytes of data are displayed.

In the READV Exit example, three IOV entries were specified, but only two were used. All the data is displayed because the total is less than 96 bytes.

Default INITAPI: An explicit INITAPI call is not required prior to some API calls, so TCP/IP creates a default INITAPI. (Refer to the z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference for the complete list.) The default INITAPI record is traced after the Entry record for the API call that caused the default INITAPI to occur. There is just one record for this event (no Exit record).

Default TERMAPI: Usually, an application ends the connection between itself and TCP/IP by issuing the TERMAPI call. But sometimes, the connection ends for another reason, such as the application being cancelled. In this case, TCP/IP issues a default TERMAPI. The default TERMAPI is traced in a SOCKAPI trace record. There is just one record for this event (no Exit record).

SELECT: For SELECT and SELECTEX, the socket masks are formatted in both binary and decimal. The socket list is displayed first in binary. The socket numbers...
are indicated by the bit position in the mask, starting with bit position 0 (for socket 0), which is the rightmost bit. The bit positions (socket numbers) are shown at left.

For example, the lowest numbered sockets are on the last line; they are sockets 0 to 31. In this line, only sockets 0, 1, 2, and 3 are selected. As shown in the following example, the binary mask, the decimal socket numbers are listed in numerical order. This is a convenient way to check if the mask is coded as intended.

<table>
<thead>
<tr>
<th>SOCKET NO.</th>
<th>READ SOCKET MASK (INPUT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Decimal)</td>
<td>(Binary)</td>
</tr>
<tr>
<td>31</td>
<td>0  00000000 00000000 00000000 00001111</td>
</tr>
<tr>
<td>63</td>
<td>32 00110111 11111111 11111111 11111101</td>
</tr>
<tr>
<td>95</td>
<td>64 11111111 11111111 11111101 11111111</td>
</tr>
<tr>
<td>127</td>
<td>96 00000000 00000000 00000000 11111111</td>
</tr>
</tbody>
</table>

SELECTED SOCKETS:
0, 1, 2, 3, 32, 34, 35, 36, 37, 38
39, 40, 41, 42, 43, 44, 45, 46, 47, 48
49, 50, 51, 52, 53, 54, 55, 56, 57, 59
60, 61, 64, 65, 66, 67, 68, 69, 70, 71
72, 73, 74, 75, 76, 77, 79, 80, 81, 82
83, 84, 85, 86, 87, 88, 89, 90, 91, 92
93, 94, 95, 96, 97, 98, 100, 101, 102, 103

If the MAXSOC value is so large that all the SELECT or SELECTEX parameters cannot be traced within a single 14K buffer, multiple trace entries are written (one trace entry for each mask). When multiple trace entries are written for the same SELECT or SELECTEX call entry or exit, all the trace data except the masks themselves are duplicated across the trace entries. For example, the time stamp is the same, the MAXSOC value is the same, the TIMEOUT value is the same, and so on. The trace description indicates to which mask the trace entries pertain. For example, if the MAXSOC value in the above trace example were 65535, then each mask would be traced individually.
Token Error: When an API call fails very early in processing, before the SOCKAPI Entry record is created, the Token Error SOCKAPI record is written. In the example, the BIND call failed due to the token being overwritten (the token at offset eight has X'FFFF'). There is no BIND Entry or Exit record.

Unsolicited event exit: If the unsolicited event exit is driven, a SOCKAPI trace record is created (if the SOCKAPI trace option is active).

Note: The key in the header is 0. This means the UEE trace record was created when TCP/IP was in key zero. The UEEXIT has key 8, which means the UE exit is invoked in key eight.

Correlating the data trace and packet trace with the SOCKAPI trace
The SOCKAPI option only records the first 96 bytes of data. To see all the data that was sent or received, you must also activate the data trace or packet trace. The data trace can be correlated easily with the SOCKAPI trace option because both traces are recording data between the application and the TCP/IP stack. The traces can be merged with the IPCS MERGE subcommand. The data trace header contains fields that allow the full data to be correlated.

Figure 16 on page 93 shows the data trace record corresponding to the READ Exit SOCKAPI trace entry in Figure 17 on page 94. The server issues READ and waits...
for a message. The data trace record shows the entire 120 bytes of data because the FULL option was used when starting the data trace. In the READ Exit record, only the first 96 bytes of data are shown.

The records in the two traces can be correlated by the following:

**Time**  The data trace time must be prior to the READ Exit record time. The data trace time is 20:08:09.181239. The READ Exit record time is 20:08:09.181354.

**Jobname**  The job name is EZASOKAS in both records.

**ASID**  The ASID is the server’s 0024 (hexadecimal) in both records.

**TCB**  The TCB is 006E6A68 in both records.

**Data length**  In the data trace, the length is 78 hexadecimal, which is 120 decimal. The SOCKAPI trace record shows that the return code is 120 (decimal) bytes.

**Port**  The source port number in the data trace record (11007 decimal) matches the local port number in the SOCKAPI trace record. The destination and remote ports also match (1040 decimal).

**IP Address**  The IP addresses are handled in the same way as the port numbers. In this example, both the client and server were on the same TCP/IP stack, so the IP addresses are the same.

Figure 16. Data trace record.
Packet trace (SYSTCPDA) for TCP/IP stacks

Packet trace is a diagnostic method for obtaining traces of IP packets flowing to and from a TCP/IP stack on a z/OS Communications Server host. You can use the PKTTRACE statement to copy IP packets as they enter or leave TCP/IP, and then examine the contents of the copied packets. To be traced, an IP packet must meet all the conditions specified on the PKTTRACE statement. The dataspace area for SYSTCPDA traces starts at two times the size of the SYSTCPIP in use.

The trace process

Trace data is collected as IP packets enter or leave TCP/IP. The actual collection occurs within the device drivers of TCP/IP, which capture the data that has just been received from or sent to the network.

Packets that are captured have extra information added to them before they are stored. This extra information is used during the formatting of the packets. The captured data reflects exactly what the network sees. For example, the trace contains the constituent packets of a fragmented packet exactly as they are received or sent.

The selection criteria for choosing packets to trace are specified through the PKTTRACE statement for the TCP/IP address space. Refer to z/OS Communications Server: IP System Administrator’s Commands for more information about the PKTTRACE statement and subcommand.

The PKTTRACE statement and subcommand are applied to device links that are defined in the TCP/IP address space through the LINK statement. Figure 18 on page 95 illustrates the overall control and data flow in the IP packet tracing facility.
**Supported devices**

IP packet tracing is supported for all network interfaces supported by TCP/IP (including loopback).

When using the MULTIPATH option of the IPCONFIG statement, packets can be sent over multiple interfaces. All of the interfaces must be traced. In this case specify an IP address to select the required packet. This statement also applies to the case where packets can be received over multiple interfaces (even if MULTIPATH is not used by this TCP/IP).

For information about the format of the packet trace command (VARY PKTTRACE) see the [z/OS Communications Server: IP System Administrator’s Commands](https://www.ibm.com/support/docview.wss?uid=swg27047759).

**Starting packet trace**

To start packet trace, use the following command:

```
V TCPIP,tcpiprocnm,PKT
```

**Security Note:** To use any VARY command, the user must be authorized in RACF.

The RACF profile for each user must have access for a resource of the form MVS.VARY.TCPIP.xxx, where xxx is the first eight characters of the command name. For packet trace, this would be MVS.VARY.TCPIP.PKTTRACE.

Traces are placed in an internal buffer, which can then be written out using an external writer. The MVS TRACE command must also be issued for component SYSTCPDA to activate the packet trace.

After starting packet trace, you can display the status using the Netstat command, as shown in the following example:
NETSTAT -p TCPCS -d
MVS TCP/IP NETSTAT CS V1R10 TCPIP Name: TCPCS 18:03:31
DevName: LOOPBACK DevType: LOOPBACK
DevStatus: Ready
LnkName: LOOPBACK LnkType: LOOPBACK LnkStatus: Ready
NetNum: 0 QueSize: 0
BytesIn: 192537 BytesOut: 192537
ActMtu: 65535
BSD Routing Parameters:
   MTU Size: 00000  Metric: 00
   DestAddr: 0.0.0.0  SubnetMask: 0.0.0.0
Packet Trace Setting:
   Protocol: *  TrRecCnt: 00000000  PckLength: FULL
   Discard: 0
   SrcPort: *  DestPort: *  PortNum: *
   IpAddr: 9.67.113.1  SubNet: *
Multicast Specific:
   Multicast Capability: No

In this example, the packet length (PckLength) is FULL and TrRecCnt is the number of packets written for this device.

Note: If you are a TSO user, use the corresponding NETSTAT DEV command.

Modifying options with VARY

After starting a packet trace, you can change the trace using the VARY command. For example, if you want to change the packet trace to abbreviate the data being traced, use the following command:
V TCPIP,tcpproc,PKT,ABBREV

You can display the results of the VARY command using the Netstat command:

NETSTAT -p TCPCS -d
MVS TCP/IP NETSTAT CS V1R10 TCPIP Name: TCPCS 18:03:31
DevName: LOOPBACK DevType: LOOPBACK
DevStatus: Ready
LnkName: LOOPBACK LnkType: LOOPBACK LnkStatus: Ready
NetNum: 0 QueSize: 0
BytesIn: 813 BytesOut: 813
ActMtu: 65535
BSD Routing Parameters:
   MTU Size: 00000  Metric: 00
   DestAddr: 0.0.0.0  SubnetMask: 0.0.0.0
Packet Trace Setting:
   Protocol: *  TrRecCnt: 00000000  PckLength: 00200
   Discard: 0
   SrcPort: *  DestPort: *  PortNum: *
   IpAddr: *  SubNet: *
Multicast Specific:
   Multicast Capability: No

Tip: If you are a TSO user, use the corresponding NETSTAT option.

By issuing multiple VARY commands, you can OR filters together. For example, issuing the following VARY commands records all packets whose destination port is xxxx or whose source port is xxxx.
V TCPIP,tcpproc,PKTTRACE,DEST=xxxx
V TCPIP,tcpproc,PKTTRACE,SRCP=xxxx

The result is a trace that contains only packets with a source port of xxxx or packets with a destination port of xxxx.
Tip: An alternative command to use is the PKTTRACE command with PORTNUM.
`V TCPIP,PKTTRACE,PORTNUM=xxxx`

If both DEST and SRCP are specified in the same command, you can AND the parameters together. For example, issuing the following VARY command records only the packets with both a destination port of `xxxx` and a source port of `yyyy`.
`V TCPIP,tcpproc,PKTTRACE,DEST=xxxx,SRCP=yyyy`

You can use the VARY TCPIP,tcpproc,OBEYFILE command to make temporary dynamic changes to system operation and network configuration without stopping and restarting the TCP/IP address space. For example, if you started the address space TCPIPA and created a sequential data set USER99.TCPIP.OBEYFIL1 containing packet trace statements, issue the following command:
`VARY TCPIP,TCPIPA,CMD=OBEYFILE,DSN=USER99.TCPIP.OBEYFIL1`

The VARY TCPIP,PKTTRACE command is cumulative. You can trace all packets for specified IP addresses by entering multiple PKTTRACE commands. In the following example, the two commands trace all the packets received and all the packets sent for the specified IP addresses.
`VARY TCPIP,,PKT,ON,IPADDR=10.27.142.44`
`VARY TCPIP,,PKT,ON,IPADDR=10.27.142.45`

Formatting packet traces using IPCS

The IPCS CTRACE command parameters are described in "Formatting component traces" on page 55. The following notes apply to the IPCS CTRACE parameters with regard to the packet trace formatter:

**JOBLIST, JOBNAME**
- The LINKNAME and JOBNAME keywords in the OPTIONS string can also be used to select records.

**TALLY**
- Equivalent to the STATISTICS(DETAIL) option.

**START and STOP**
- Packets are numbered after the START keyword has filtered records.

**LIMIT**
- See the RECORDS keyword in the OPTIONS string.

**USEREXIT**
- The packet trace formatter calls the CTRACE USEREXIT before testing the records with the filtering criteria. If it returns a nonzero return code, then the record is skipped. The USEREXIT can also be used in the OPTIONS string. It is called after the record has met all the filtering criteria in the OPTIONS string.

**COMP**
- Must be SYSTCPDA.

**SUB**
- The SUB must name the TCP/IP procedure that created the CTRACE records when the input is a dump data set.

**CONSOLE**
- Since there are no EXCEPTION records for packet trace, the EXCEPTION keyword must not be specified.

**ENTIDLIST**
- The following are the valid values for packet trace:
1. IPv4 packet trace records
2. X25 trace records
3. IPv4 Enterprise Extender data trace records
   Tip: Type 1, Type 2, and Type 3 records are no longer written by TCP/IP.
4. IPv4 and IPv6 packet trace records
5. IPv4 and IPv6 data trace records
6. Enterprise Extender trace records

The CTRACE OPTIONS string provides a means of entering additional keywords for record selection and formatting packet traces (COMP=SSTCPDA). See "Syntax" on page 57 for the complete syntax of CTRACE.

**OPTIONS syntax**

**OPTIONS component**

```
  OPTIONS---(( Data Selection | Report Generation ))---GMT---LOCAL
```

**Data Selection:**

```
  Device Type | IP Identifier | IP Address | Name
  Port Number | Protocol     | Record Number | Record Type | SNA Address
```

**Device Type:**

```
  DEVTYPE---(devtype)
```

```
  ETHTYPE---(type) DEVICEID---(device_id) MACADDR---(macaddr) VLANID---(vlanid)
```

**IP Identifier:**

```
  ADDR---(IP Address)---BROADCAST---CLASSA---CLASSB---CLASSC---CLASSD
```

```
  CLASSE---HOST---IPADDR---(IP Address) | IPID---(ip_id_number)
```

```
  IPv4---IPv6---LINKLOCAL---LOOPBACK---LOOPBACK6---MULTICAST
```
Chapter 5. TCP/IP services traces and IPCS support
Chapter 5. TCP/IP services traces and IPCS support
OPTIONS keywords
The following are keywords used for the OPTIONS component routine parameters.

AH
Select packets with an AH extension header.

ASCII
Packet trace data dumped is shown in hexadecimal and interpreted in ASCII translation only. The default is BOTH.

BASIC ((DETAIL|SUMMARY))
For specific packet types, format each element of the packet data. This parameter applies to DNS, RIP, and SNMP packet data.

DETAIL
Format the IP header, protocol header and protocol data in as few lines as possible. DETAIL is the default.

SUMMARY
Format the IP and protocol headers in as few lines as possible.

BOOTP[(port_number|67 port_number|68)]
Select BOOTP and DHCP protocol packets. The port_number defines the BOOTP and DHCP port numbers to select packets for formatting. Equivalent to PORT(67 68).

BOTH
Packet trace data dumped is shown in hexadecimal and interpreted with both ASCII and EBCDIC translations. The default is BOTH.

BROADCAST
Select packets with a broadcast IPv4 address. Equivalent to IPADDR(255.255.255.255).

CHECKSUM [(DETAIL|SUMMARY)]
The selected packets have their checksum values validated.
DETAIL
If there is a checksum error, then the packet is formatted and dumped.

SUMMARY
A message is issued for each packet that encounters a checksum error.
SUMMARY is the default.

CID
Selects data trace records that contain the specific connection ID value. The
connection ID value can be determined from the NETSTAT CONN display. Up
to 16 values or ranges can be specified.

CLASSA
Select packets with a class A IPv4 address. Equivalent to IPADDR(0.0.0.0/
128.0.0.0).

CLASSB
Select packets with a class B IPv4 address. Equivalent to IPADDR(128.0.0.0/
192.0.0.0).

CLASSC
Select packets with a class C IPv4 address. Equivalent to IPADDR(192.0.0.0/
224.0.0.0).

CLASSD
Select packets with a class D IPv4 address. Equivalent to IPADDR(224.0.0.0/
240.0.0.0).

CLASSE
Select packets with a class E IPv4 address. Equivalent to IPADDR(240.0.0.0/
248.0.0.0).

CLEANUP(nnnnn|500)
Defines a record interval where saved packet information in storage is released.
The minimum value is 500 records; the maximum value is 1 048 576 records;
the default is 500 records. If you set the record interval to 0, cleanup does not
occur.

DATASIZE (data_size|0)
Selects packets that contain more protocol data than the data_size value. The
minimum value is 0. The maximum value is 65535. The data size is determined
from the amount of packet data available minus the size of any protocol
headers. Equivalent to FLAGS(DATA).

DATTRACE
Select packets written from the VARY TCPIP,,DATTRACE command.

DEBUG(debug_level_list)
Provides documentation about SYSTCPDA format processing. debug_level_list
is a list of numbers from 1 to 64. Use only under the direction of an IBM
Service representative.

DELAYACK(threshold|200)
The delay acknowledgment threshold in milliseconds used in the calculation of
round trip time in the TCP session report. The minimum value is 10
milliseconds. The maximum value is 1000 milliseconds. The default value is
200 milliseconds.

DEVICEID(device_id)
Selects packets written to or received from an OSAENTA trace with one of the
specified device identifiers. One to sixteen device IDs can be specified. This filter applies only to type 7 trace records. The device_id value is a hexadecimal number in the form X'csnflua':

- \texttt{cs} \quad \text{The channel subsystem ID for this datapath device.}
- \texttt{mf} \quad \text{The LPAR Multiple Image Facility ID for the LPAR using this datapath device.}
- \texttt{cl} \quad \text{The control unit logical identifier for this datapath device.}
- \texttt{ua} \quad \text{The unit address for this datapath device.}

Each identifier is a 2-digit hexadecimal value in the range 00 - FF.

\textbf{Tip:} You can obtain the device_id values for any active user of the OSA by using the Hardware Management Console (HMC). For a data device that is active on a z/OS stack, you can obtain the device_id value for that data device from message IST2190I of the output from the D NET,TRLE command.

\textbf{DEVTYPE(device_type_list)}

Select packets written to or received from an interface with one of the specified device types. From 1 to 16 types can be specified. This does not apply to data trace records. The following types can be specified:

- ATM
- CDLC
- CLAW
- CTC
- ETHER8023
- ETHERNET
- ETHEROR8023
- FDDI
- HCH
- IBMTR
- IPAQENET
- IPAQENET6
- IPAQIDIO
- IPAQIDIO6
- IPAQTR
- LOOPBACK
- LOOPBACK6
- MPCPTP
- MPCPTP6
- OSAFDDI
- OSAENET
- SNALINK
- SNALU62
- VIRTUAL
- VIRTUAL6
- X25NPSI

\textbf{DISCARD(reason_code_list)}

Select packets with one of the specified discard reason codes. Up to 16 discard
reason codes can be specified in the range 0 - 65535. Each entry in the list can be a range: low_number:high_number. Values can be decimal or hexadecimal.

0 Packet was not discarded
1:4087 A packet was discarded by OSA-Express
1:1023 Select packets discarded by OSA-Express for DISCARD=EXCEPTION reasons
4096:8191 IP packet was discarded by TCPIP
8192:12287 TCP packet was discarded by TCPIP

See z/OS Communications Server: IP and SNA Codes for the TCP/IP discard reason codes.

**DNS**[(port_number|53)]
Select Domain Namer Service protocol packets. The port_number defines the DNS port number to select packets for formatting. Equivalent to PORT(53).

**DOMAIN**[(port_number|53)]
Select Domain Namer Service protocol packets. The port_number defines the DNS port number to select packets for formatting. Equivalent to PORT(53).

**DUMP**[(nnnnn|65535)]
Dump the selected packets in hexadecimal with EBCDIC and ASCII translations. The IP and protocol headers are dumped separately from the packet data. The value nnnnn represents the maximum amount of packet data that is to be dumped from each packet. The default value is 65535 bytes. The minimum value is 0. The maximum value is 65535. The IP and protocol headers are not subject to this maximum.

The default report options are DUMP and FORMAT.

The BOTH, ASCII, EBCDIC and HEX keywords describe how the dumped packets are translated. The default is BOTH. The display can be changed using these keywords. The default ASCII translation table is used. This table might not match the table being used by the application. When formatting the CTRACE, it is helpful to have the correct line length. Use the IPCS PROFILE LINESIZE command to set the line length. For example,

IPCS PROFILE LINESIZE(80)

sets the maximum line length to 80 characters so that all formatted data is viewable within 80 characters.

If the STREAM report is chosen, then the dump of the packets is deferred until the stream of data has been collected.

**EBCDIC**
Packet trace data dumped is shown in hexadecimal and interpreted with EBCDIC translation only. The default is BOTH.

**EE** Select Enterprise Extender (EE) protocol packets. The port number defines the first EE port number to select packets for formatting. The EE port number and the next four port numbers are used. Equivalent to PORT(12000:12004).

**ELEMENT(element_number_list)**
Select SNA protocol packets with a matching origin or destination element address in the TH2 or TH4 transmission header. Valid values are in the range from 0 - 65535. Up to 16 element numbers can be specified.

**ESP**
Select packets with a protocol number of 50. Equivalent to PROTOCOL(50).

**ETHTYPE**(type)
Selects packets written to or received from an OSAENTA trace with one of the
specified frame types. From 1 to 16 types can be specified. This filter only applies to type 7 trace records. The following types can be specified:

- x'0800' for IP
- x'86DD' for IPV6
- x'0806' for ARP
- x'80d5' for SNA

**EXPORT[(DETAIL|SUMMARY)]**

The selected packets are written to the EXPORT data set in .CSV (Comma Separated Value) format. In .CSV format, each character field is surrounded by double quotation marks and successive fields are separated by commas. The file's first line defines the fields. Each subsequent line is a record containing the values for each field.

**DETAIL**

Format the IP header, protocol header and protocol data as separate lines of data.

**SUMMARY**

Format the IP header and protocol header in one line of data. SUMMARY is the default.

Allocate a file with DDNAME of EXPORT before invoking the CTRACE command with EXPORT in the OPTIONS string.

```
ALLOC FILE(EXPORT) DA(PACKET.CSV) SPACE(15 15) TRACK
```

The record format is variable block with logical record length of 512 bytes.

**FINGER[port_number|79]**

Select FINGER protocol packets. The port_number defines the FINGER port number to select packets for formatting. Equivalent to PORT(79).

**FIRST|LAST**

Selects which packet in a set of encapsulated packets is used for selection. An example is the ICMP error report packet that contains the IP header that is in error. FIRST indicates that the ICMP packet is used for selection. LAST indicates that the last encapsulated IP header is used for selection. FIRST is the default.

**FLAGS(flags list)**

Select packets that have the matching characteristics. Flags that can be specified are:

- **ALL** When more than one flag is specified, the packet must meet all the criteria of the flags requested. ALL is the default.
- **ANY** When more than one flag is specified, the packet need only meet one of the criteria of the flags requested.

**ABBREV**

Select packets that are abbreviated.

- **ACK** Select packets that have a TCP header with the ACK flag set.
- **BAD** Select packets that may be too short to contain all the required headers
- **BBI** The SNA packet contains a begin bracket indicator.
- **BCI** The SNA packet contains a begin chain indicator.
- **CDI** The SNA packet contains a change direction indicator.
- **CEBI** The SNA packet contains a conditional end bracket indicator.
CSI  The SNA packet contains a code selection indicator.

CKSUM  Select packets that have a check sum error.

DATA  Selects packets that contain data.

DF  Select packets that have a non-zero discard code. These packets were discarded by TCP/IP.

DFC  The SNA packet is a data flow control packet.

DISCARD  Select packets that have a non-zero discard code. These packets were discarded by OSA-Express or by TCP/IP.

DR1  The SNA packet is requesting a DR1 response.

DR2  The SNA packet is requesting a DR2 response.

EBI  The SNA packet contains an end bracket indicator.

ECI  The SNA packet contains a end chain indicator.

EDI  The SNA packet contains an enciphered data indicator.

ERI  The SNA packet is an error response.

FI  The SNA packet contains formatted data.

FIB  The SNA packet is the first packet of a bracket (or of a conditional begin bracket). The RH BBI flag is set and the EBI flag is not.

FIC  Select packets that are the first in chain SNA RU.

FIN  Select packets that have a TCP header with the FIN flag set.

FIS  Select packets that are in the first fragment of an IPv4 or IPv6 packet or the first segment of a SNA PDU.

FMD  The SNA packet is a function management data packet.

FMH  The SNA packet is a function management data header.

FRAME  Selects OSAENTA packets that have a frame header.

FULL  Select packets that are complete.

HOME  Select packets that have an IP destination address equal to the IP source address.

IN  Select packets that are inbound.

IPEXT  Select packets that have an extension header.

IPO  Select packets that have an IPv4 header options field.

IPv4  Select IPv4 packets. IPv4 cannot be used in combination with other data selectors that are IPv6-specific, such as LINKLOCAL.

IPv6  Select IPv6 packets. IPv6 cannot be used in combination with other data selectors that are IPv4-specific, such as BROADCAST.

IPv6Ext  Select packets that have an extension header. This is equivalent to IPEXT.
LIB The SNA packet is a last packet of a bracket. The RH BBI flag is not set and the EBI flag is set.

LIC Select packets that are the last in a chain of SNA RUs.

LIS Select packets that are the last fragment of an IPv4 or IPv6 packet or the last segment of a SNA PDU.

LPAR Select NTA packets that were transmitted between LPARs shared by an OSA-Express device.

L2 The OSAENTA packet is from a layer 2 OSA application.

L3 The OSAENTA packet is from a layer 3 OSA application (like TCP/IP).

MIB The SNA packet is in the middle of a bracket. The RH BBI flag is not set and the EBI flag is not set.

MIC Select packets that are the middle fragment of an IPv4 or IPv6 packet.

MIS Select packets that are the middle fragment of an IPv4 or IPv6 packet or the middle segment of a SNA PDU.

NC The SNA packet is a Network Control packet.

NTA Select OSAENTA packets.

OFFLOAD Select outbound packets for which segmentation has been offloaded.

OIB The SNA packet is the only packet of a bracket. The RH BBI flag is set and the EBI flag is set.

OIC Select packets that are only in a chain SNA RH request.

OIS Select packets that are IPv4 or IPv6 packets that are not fragmented or that are the only segment of a SNA PDU.

OUT Select packets that are outbound.

PDI Select SNA packets with the padded data indicator.

PDU The IP packets that were packed by TCP/IP into a single PDU buffer.

PI The SNA packet contains a pacing indicator.

PING Select packets that are ICMP/ICMPv6 echo request and echo reply.

PSH Select packets that have a TCP header with the PSH flag set.

QRI The SNA packets with a queued response indicator

REQ The SNA packet is a request.

RESP The SNA packet is a response.

RLWS Select SNA packets with the request large window size indicator.

RSM Select packets that have been reassembled.

RST Select packets that have a TCP header with the RST flag set.

SC The SNA packet is a session-control packet.

SDI The SNA packet contains sense data.

SEG Select packets that have been segmented.

SNA Select SNA packets.

SYN Select packets that have a TCP header with the SYN flag set.
TCPO  Select packets that have a TCP header options field.
TOS   Select IPv4 packets that have a nonzero value in the ip_tos field.
TUNNEL  Select packets with protocol number 47 GRE or 41 (IPv6 over IPv4).
  z/OS Communications Server currently does not support IPv6 over IPv4 (protocol number 41).
URG   Select packets that have a TCP header with the URG flag set.
VLAN  Select packets that have a VLAN 802.1q tag
ZWIN  Select packets that have a TCP header with a zero window value.

Notes:
1. The use of the FIC, MIC and LIC flags require the use of the
   NOREASSEMBLY option.
2. When a packet is reassembled, then it becomes an OIC packet with the
   RSM flag set.
3. Do not intermix SNA and IP flags.

Table 14. Flags that apply to IP or SNA packets

<table>
<thead>
<tr>
<th>Flag</th>
<th>Applies to IP</th>
<th>Applies to SNA</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABBREV</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>BAD</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>BBI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>CDI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>CEBI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>CI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>CKSUM</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>DF</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>DFC</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>DISCARD</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>DR1</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>DR2</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>EBI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>ECI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>EDI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>ERI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>FI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>FIB</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>FIC</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>FIN</td>
<td>Y</td>
<td>N</td>
<td>TCP only</td>
</tr>
<tr>
<td>FIS</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>FM</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Flag</td>
<td>Applies to IP</td>
<td>Applies to SNA</td>
<td>Comments</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>----------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>FMD</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>FMH</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>FRAME</td>
<td>N</td>
<td>Y</td>
<td>OSAENTA only</td>
</tr>
<tr>
<td>FULL</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>HOME</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>IPEXT</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>IPO</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>IPV4</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>IPV6</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>IPV6EXT</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>LIB</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>LIC</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>LIS</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>LPAR</td>
<td>Y</td>
<td>Y</td>
<td>OSAENTA only</td>
</tr>
<tr>
<td>L2</td>
<td>Y</td>
<td>Y</td>
<td>OSAENTA only</td>
</tr>
<tr>
<td>L3</td>
<td>Y</td>
<td>Y</td>
<td>OSAENTA only</td>
</tr>
<tr>
<td>MIB</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>MIC</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>MIS</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>NTA</td>
<td>Y</td>
<td>Y</td>
<td>OSAENTA only</td>
</tr>
<tr>
<td>OFFLOAD</td>
<td>Y</td>
<td>N</td>
<td>TCP only</td>
</tr>
<tr>
<td>OIB</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>OIC</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>OIS</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>PDI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>PDU</td>
<td>Y</td>
<td>N</td>
<td>SYSTCPDA only</td>
</tr>
<tr>
<td>PI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>PING</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>PSH</td>
<td>Y</td>
<td>N</td>
<td>TCP only</td>
</tr>
<tr>
<td>QRI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>REQ</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>RESP</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>RLWS</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>RSM</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>RST</td>
<td>Y</td>
<td>N</td>
<td>TCP only</td>
</tr>
<tr>
<td>SC</td>
<td>N</td>
<td>Y</td>
<td>TCP only</td>
</tr>
<tr>
<td>SDI</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>
Table 14. Flags that apply to IP or SNA packets (continued)

<table>
<thead>
<tr>
<th>Flag</th>
<th>Applies to IP</th>
<th>Applies to SNA</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEG</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>SYN</td>
<td>Y</td>
<td>N</td>
<td>TCP only</td>
</tr>
<tr>
<td>TCPO</td>
<td>Y</td>
<td>N</td>
<td>TCP only</td>
</tr>
<tr>
<td>TOS</td>
<td>Y</td>
<td>N</td>
<td>SNA TPF field</td>
</tr>
<tr>
<td>TUNNEL</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>URG</td>
<td>Y</td>
<td>N</td>
<td>TCP only</td>
</tr>
<tr>
<td>VLAN</td>
<td>Y</td>
<td>Y</td>
<td>OSAENTA only</td>
</tr>
<tr>
<td>ZWIN</td>
<td>Y</td>
<td>N</td>
<td>TCP only</td>
</tr>
</tbody>
</table>

**FMT**
Equivalent to FORMAT.

**FORMAT[(DETAIL | SUMMARY | ALL | FIRST | LAST)]**
The selected packets with defined packet data are to be formatted. The SHORT keyword on the CTRACE command selects this option if no other report options are specified. The default report options are DUMP and FORMAT.

**DETAIL**
Format the IP header, protocol header, and the protocol data.

**SUMMARY**
Format the IP header and protocol header. DETAIL is the default.

**ALL**
Format all encapsulated packets. ALL is the default.

**FIRST**
Format the first encapsulated packet.

**LAST**
Format the last encapsulated packet

An example of an encapsulated packet is an ICMP error report.

**FTP[(data_port_number | 20 control_port_number | 21)]**
Select FTP protocol packets. The port_number defines the FTP port numbers to select packets for formatting. Equivalent to PORT(20,21).

**FULL**
Equivalent to DUMP and FORMAT. The FULL keyword on the CTRACE command selects this option if no other report options are specified.

**GAIN(rtgain | 125,vargain | 250)**
Values of the round trip gain (rtgain) and the variance gain (vargain), in milliseconds, used in the calculation of round trip time in the TCP session report. Valid values are in the range 0–1000. The default values for rtgain is 125. The default value for vargain is 250.

**GOPHER[(port_number | 70)]**
Select GOPHER protocol packets. The port_number defines the GOPHER port numbers to select packets for formatting. Equivalent to PORT(70).

**GRE**
Select packets with a protocol number of 47. Equivalent to PROTOCOL(47).
GMT
Format the time stamps in GMT time. The default is the value specified on the
CTRACE subcommand.

HEX
Packet trace data dumped is shown in hexadecimal only with no translation.
The default is BOTH.

HOST
Select packets with a host IP address. Equivalent to IPADDR(0.0.0.0/
255.255.0.0)

HTTP([port_number|80])
Select HTTP protocol packets. The port_number defines the HTTP port
numbers to select packets for formatting. Equivalent to PORT(80). See 120.

ICMP
Select packets with a protocol number of 1. Equivalent to PROTOCOL(1).

ICMP6 or ICMPV6
Select packets with a protocol number of 58. Equivalent to PROTOCOL(58).

IGMP
Select packets with a protocol number of 2. Equivalent to PROTOCOL(2).

INTERFACE(interface_name_list) or LINKNAME(interface_name_list)
Select packet trace records with the specified interface name. Up to 16 interface
names can be specified. Each interface name can be up to 16 characters. Use an
asterisk (*) as a wild card to replace characters at the end of the interface
name.

IPADDR(ipaddr[/mask_or_prefixlength]|X'hhhhhhhh'[-nnnn])
Select packets with a matching IP address, optional IPv4 address mask or IPv6
prefix length and optional port number. Up to 16 IP addresses can be specified.
The IPADDR is specified in three parts:
1. An IPv4 or IPv6 address
   The IPv4 address can be in dotted decimal notation, a keyword, or a hex
   value.
   • IPv4 dotted decimal notation
     127.0.0.1
   • IPv4 keyword
     A A class A IPv4 address, 0.0.0.0/128.0.0.0
     B A class B IPv4 address, 128.0.0.0/192.0.0.0
     C A class C IPv4 address, 192.0.0.0/224.0.0.0
     D A class D IPv4 address, 224.0.0.0/240.0.0.0
     E A class E IPv4 address, 240.0.0.0/248.0.0.0
     H A local host address, 0.0.0.0/0.0.255.255
     L An IPv4 or IPv6 loopback address, 127.0.0.0/255.0.0.0 or ::1
     M The broadcast IPv4 address, 255.255.255.255/255.255.255.255
     * Any address, 0.0.0.0/0.0.0.0
     0 An IPv4 or IPv6 address of zero, 0.0.0.0/255.255.255.255 or ::/128
   • IPv4 or IPv6 address as a hexadecimal number up to 32 (IPv4) or 128
     (IPv6) digits
• IPv6 address
  1080::8:800:200c:417a

2. An IPv4 address mask or IPv6 prefix length
   The IPv4 address mask (1–32) or IPv6 prefix length (1–128) is preceded by a
   slash (/). Specify an IPv4 address mask only when the IPv4 address is in
dotted decimal notation. The IPv4 address mask can be in dotted decimal
notation, for example: 9.37/255.0.0.0 or 9.37/255.255.0.0

3. A port number
   The port number is preceded by a dash (-). It is a decimal number in the
   range 0–65535.

Notes:
1. There should be no spaces between the IP addresses and the subnet masks.
2. The BROADCAST, CLASSA, CLASSB, CLASSC, CLASSD, CLASSE, HOST,
   LINKLOCAL, LOOPBACK, MULTICAST, and SITELOCAL keywords add
to the total of 16 IP addresses.
3. The port number when used adds to the total of 16 port numbers in the
   PORT keyword.
4. IPv4 addresses and IPv4-mapped IPv6 addresses are treated as equivalent
   addresses.

IPID(pid_number_list)
   Select packets that match the ip_id number in the IPv4 packet header. Up to 16
   ID numbers can be specified in the range 0–2147483647 or 0–X'FFFFFFF'. Each
   entry in the list can be a range: low_number:high_number. Values can be
decimal (nnnnn) or hexadecimal (X'hhhh'). If the packets have been
   fragmented, specify NOREASSEMBLY to select each packet.

IPv4
   Equivalent to FLAGS(IPV4).

IPv6
   Equivalent to FLAGS(IPV6).

IKE
   Select ISAKMP protocol packets. Equivalent to PORT(500). See the ISAKMP
   keyword.

ISAKMP
   Select ISAKMP protocol packets. Equivalent to PORT(500). See the IKE
   keyword.

JOBLIST | JOBNAME(job_name_list)
   Select data trace records with the specified JOBNAME. Up to 16 job names can
   be specified. Each job name can be up to 8 characters. If the last character of a
   job name is an asterisk (*) then only the characters up to the asterisk are
   compared.

   The CTRACE JOBLIST/JOBNAME parameter provides the same function,
   except that wildcards are not supported.

LIMIT(record_count)
   record_count
   The maximum number of records that are formatted. The default value
   999 999 999 records.
Guideline: This keyword is also accepted if specified on the CTRACE subcommand.

LINKLOCAL
Select packets with an IPv6 link-local unicast prefix. Equivalent to IPADDR(FE80::/10).

LINKNAME(link_name_list)
Select packet trace records with the specified LINKNAME. Up to 16 link names can be specified. Each link name can be up to 16 characters. If the last character of a link name is an asterisk (*) then only the characters up to the asterisk are compared.

The CTRACE JOBLIST/JOBNAME parameter provides the same function, except that wildcards are not supported and only the first 8 characters of the link name are compared.

LOCAL
Format the time stamps in local time. The default is the value specified on the CTRACE subcommand.

LOOPBACK
Select packets with either an IPv4 or IPv6 loop back address. Equivalent to IPADDR(127.0.0.0/255.0.0.0::1). If other addresses are defined as loopback, they can be selected explicitly using IPADDR().

LOOPBACK6
Select packets with an IPv6 loop back address. Equivalent to IPADDR(:1). If other addresses are defined as loopback, they can be selected explicitly using IPADDR().

MACADDR(macaddr)
Selects packets written to or received from an OSAENTA trace with one of the specified MAC addresses. From 1 to 16 addresses can be specified. This filter only applies to type 7 trace records. A MACADDR is twelve hexadecimal digits.

MULTICAST
Select packets with either an IPv4 or IPv6 multicast address. Equivalent to CLASSD IPADDR(FF00::/8).

NAT
Select NAT protocol packets. Equivalent to PORT(4500).

NOCHECKSUM
The selected packets do not have their checksum values validated. CHECKSUM is the default.

NOREASSEMBLY
Do not reassemble fragmented IP packets into a complete packet. REASSEMBLY is the default.

NOSEGMENT
Packet trace records that span multiple Ctrace records are not recombined. Only the first segment record of packet is used. The rest of the segment records are discarded. SEGMENT is the default.

NOT
If the NOT option is selected then any selection criteria is reversed. If a record matches the selection criteria, it is not processed. If a record does not match the selection criteria, it is processed.
NTP[(port_number|123)]
Select NTP protocol packets. The port number defines the NTP port number to
select packets for formatting. Equivalent to PORT(123).

OPTION
The selected options with defaults are listed.

OSPFI
Select packets with a protocol number of 89. Equivalent to PROTOCOL(89).

PACKETTRACE
Select packets written from the VARY TCPIP,,PKTTRACE command.

IPEXT
Select packets with an extension header.

PORT(port_number_list)
Select packets with one of the specified port numbers. Up to 16 port numbers
can be specified in the range 0–65535. Each entry in the list can be a range:
low_number:high_number. Values can be decimal (nnnnn) or hexadecimal
(X'hhhh'). The following keywords add to the list of 16 port numbers:
• BOOTP
• DHCP
• DNS
• DOMAIN
• EE
• FINGER
• GOPHER
• HTTP
• NAT
• IKE
• RIP
• NTP
• ROUTER
• RPC
• SASP
• SMTP
• SNMP
• TELNET
• TFTP
• TIME
• WWW

PROTOCOL(protocol number list)
Select packets with one of the specified protocol numbers. Up to 16 protocol
numbers can be specified in the range 0–255. Each entry in the list can be a range:
low_number:high_number. Values can be decimal (nnn) or hexadecimal
(X'hh').

Protocol filters on only the upper-layer header of an IPv6 packet. It does not
filter for IPv6 extension headers (Hop-by-Hop Options, Routing, Fragment).
Instead, IPv6 extension headers are included in the display of the basic IPv6
header. The following keywords add to the list of 16 protocol numbers:
• AH
QOS(quality_of_service_list)
Select the records with the matching Quality of Service from the IPv4 Type of Service field. Up to 16 QoS values can be specified in the range 0-7. Each entry in the list can be a range: low_number:high_number. Values can be decimal (n) or hexadecimal (X'h').

REASSEMBLY(packet_size|65535,DETAIL|SUMMARY)
Reassemble IP fragments into a complete packet.

packet_size
The maximum size of a reassembled packet that is allowed. The smallest value allowed is 576 bytes, the largest is 65535 bytes. The default value is 65535 bytes.

DETAIL
List each of the reassembly statistics for each packet when a packet completes reassembly.

SUMMARY
Show only the reassembly statistics and information about packets that did not complete reassembly.

REASSEMBLY(65535,SUMMARY) is the default.

RECORDS(record_number_list)
Select the records with matching record numbers in the trace data. Up to sixteen (16) record numbers can be specified. Record numbers are assigned after any IPCS CTRACE selection criteria have been met. Each entry in the list can be a range: low_number:high_number. Values can be decimal (nnnnnnnnnn) or hexadecimal (X'hhhhhhhh').

RIP(port_number|520)
Select RIP protocol packets. The port_number defines the RIP port number to select packets for formatting. Equivalent to PORT(520).

ROUTER(port_number|520)
Select RIP protocol packets. The port_number defines the RIP port number to select packets for formatting. Equivalent to PORT(520).

RIPNG
Select packets with a port number of PORT(521). Equivalent to PORT(521).

RPC(port_number|111)
Select RPC protocol packets. The port_number defines the RPC port number to select packets for formatting. Equivalent to PORT(111).
**SASP (port_number|3860)**
Select z/OS Load Balancing Advisor port numbers. The port_number defines the SASP port number to select packets for formatting. Equivalent to PORT(3860).

**SEGMENT**
Packet trace records that span multiple Ctrace records are recombined. Data from segment records is saved until all the Ctrace records have been read to re-create the original packet. SEGMENT is the default.

**SESSION[(DETAIL|PIPE|STATE|SUMMARY)]**
Generate a report showing TCP or UDP session traffic.

- **DETAIL**
  List each of the packets for a session, as well as the summary statistics.
  DETAIL is the default.

- **PIPE**
  List the amount of data left unacknowledged.

- **STATE**
  List the beginning and ending state of each session.

- **SUMMARY**
  Show only the summary statistics.

**Tip:** The UDP session analysis is also used for other protocols.

**SITELOCAL**
Select packets with an IPv6 site-local unicast address prefix. Equivalent to IPADDR(FEC0::/10).

**SMTP[(port_number|25)]**
Select SMTP protocol packets. The port_number defines the SMTP port number to select packets for formatting. Equivalent to PORT(25).

**SNIFFER[(nnnnn|200, ETHERNET|TCPDUMP)]**
Writes the trace records in a format acceptable for downloading to other trace analysis programs, such as programs from [http://www.wireshark.org/](http://www.wireshark.org/)

- **nnnnn**
  The maximum size of trace data. Packets with more data than this value are truncated. The default is 200 bytes. The largest value is derived from the LRECL of the SNIFFER data set.

- **ETHERNET**
  If this keyword is specified, the output is formatted for the Ethernet analysis application of the analyzer. This keyword specifies the file format only and does not imply that only packets traced on an Ethernet are collected. Packets from all devices can be collected using this option.

  The default for the SNIFFER option is ETHERNET.

- **TCPDUMP**
  The format is compatible with the [www.tcpdump.org](http://www.tcpdump.org) files with an Ethernet header.

**Note:** The TOKENRING keyword on the CTRACE OPTIONS(( SNIFFER(TOKENRING) )) on the IPCS Ctrace subcommand is ignored. The ETHERNET format of the sniffer data set will be selected.

The trace records are written to the file with a DD name of SNIFFER. After the file is generated, it can be downloaded as a binary file to the analyzer and
loaded using the standard features of the analyzer. Use NOREASSEMBLY to prevent the formatter from reassembling packets. Then, each packet is passed as the packets are collected. The logical record length of the SNIFFER data set determines the largest amount of packet data written to the data set.

Allocate a file with DDNAME of SNIFFER before invoking the CTRACE command with SNIFFER in the OPTIONS string as follows:

```
ALLOC FILE(SNIFFER) DA(PACKET.TRC) SPACE(15 15) TRACK +
   LRECL(8000) BLKSIZE(32000)
```

The data set has a record format of variable blocked with a logical record length of 8000 bytes. The maximum IP packet size is 7962 (8000 - 38) for SNIFFER(ETHERNET).

The minimum logical record length of the data set is 256 bytes.

**SNMP**(port_number|161 port_number|162)

Select SNMP protocol packets. The port_number defines the SNMP port number to select packets for formatting. Equivalent to PORT(161 162).

**SPEED**(local | 10, remote | 10)

The link speed, in megabits per second, for the local and remote link. These values are used in throughput calculations in the TCP session report. Valid values are in the range 0-17171. The default value is 10. Specify the slowest speed of the link in the route.

**STATISTICS**(DETAIL | SUMMARY)

After all the records have been processed, generate statistical reports.

**DETAIL**
Reports are produced showing the number of records selected by record type, device type, jobname, linkname, protocol number, IP address and port numbers. The session summary report is a listing of the IP address and port number pairs showing the number of records, the first and last record numbers, and the first and last record times.

**SUMMARY**
Only the session summary report is produced. SUMMARY is the default.

TALLY on the CTRACE command selects this option if no other report options are specified.

**STATS**
Equivalent to the STATISTICS option.

**STREAMS**(stream_size | 128 DETAIL | SUMMARY)

Collect the packet data for dumping or formatting after the trace file is processed. The value *nnn* represents the maximum amount of storage used to capture each stream. The value *stream_size* represents the maximum amount of storage used to capture each stream. The smallest value is 16KB. The largest value is 512KB. The default value is 128KB. The value is in 1024 bytes (1K) units.

**SUMMARY**
List about each packet in the stream. SUMMARY is the default.

**DETAIL**
Issue messages about the status of the stream.

**Requirement**: The DUMP keyword is required to dump the packet data.
SUBAREA(subarea_number_list)
Select SNA protocol packets with a matching subarea address in the TH4 transmission header. Valid values are in the range 1 - 65535. You can specify up to sixteen subarea numbers.

SUMMARY
Format a single line for each trace record. SUMMARY on the CTRACE command selects this option if no other report options are specified. If no other report option specified on the CTRACE command, then SUMMARY is selected as the report.

TALLY
Equivalent to the STATISTICS(DETAIL) option.

TCID(transport_connection_id_list)
Select SNA protocol packets with a matching transport connection identifier in the RTP transport header. Valid values range from 1 - 16 hexadecimal digits. Up to sixteen transport connection identifiers can be specified.

TCP
Select packets with a protocol number of 6. Equivalent to PROTOCOL(6).

TELNET[(port_number|23 [screen_width|80] [SUMMARY|DETAIL] ) ]
Select TELNET protocol packets. The port_number defines the TELNET port number to select packets for formatting. Equivalent to PORT(23).

The screen_width parameter defines the value used for converting buffer offsets into row and column values for the 3270 data stream formatting. If the screen_width parameter is provided, then the port_number parameter must also be used. The minimum value is 80. The maximum value is 255. The default value is 80.

SUMMARY formats the 3270 data stream into a representation of the screen. DETAIL formats each 3270 command and order.

There is no default for DETAIL or SUMMARY.

TFTP[(port_number|69)]
Select TFTP protocol packets. The port_number defines the TFTP port number to select packets for formatting. Equivalent to PORT(69).

TH5ADDR(session_address_list)
Select SNA protocol packets with a matching session address in the TH5 transmission header. Valid values range from 1 - 16 hexadecimal digits. You can specify up to sixteen session addresses.

TIME[(port_number|37)]
Select TIME protocol packets. The port_number defines the TIME port number to select packets for formatting.

TOD
Use the time the trace data was captured for the reports. Normally the time the trace data was moved to the trace buffer is shown. The CTRACE command uses the time stamp when the trace data was moved to the buffers for START and STOP time selection.

TRAFFICCLASS(traffic_class)
Select the records with the matching IPv6 traffic class field. Up to 16 traffic class values can be specified in the range from 0 to 255. Each entry in the list can be a range: low_number:high_number. Values can be decimal (nn) or Hexadecimal (Xhh').
UDP
Select packets with a protocol number of 17. Equivalent to PROTOCOL(17).

USEREXIT(exitname)
Names the user exit to be called for each selected record. The USEREXIT keyword on the CTRACE command names a user exit that is called before the SYSTCPDA packet trace filtering is done. If this exit routine returns a nonzero return code, then the record is skipped by the SYSTCPDA formatter.

VLANID(vlanid)
Select packets written to or received from an OSAENTA trace with one of the specified VLAN identifiers. From 1 to 16 identifiers can be specified. This filter only applies to type 7 trace records. A VLAN identifier has a value in the range 0 - 4094.

Tip: The DEVICEID, MACADDR, ETHTYPE, and VLANID filter keywords apply to SYSTCPOT data. If these keywords are specified with SYSTCPDA data, then these filters will be ignored.

WWW[(port_number|80)]
Select HTTP protocol packets. The port_number defines the HTTP port number to select packets for formatting. Equivalent to PORT(80).

X25
Select packet trace records created by the X25 processor.

Tip: This option is obsolete, but it is still accepted.

Report Examples
The CTRACE packet trace (SYSTCPDA) report generation outputs are described in the following examples.

Because IPv6 increases the IP address size, formatted IPv6 packet/data traces might be much wider than 80 columns.
OPTION:

Purpose: List the selected options and default keyword values.

Format: CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((STAT(DETAIL)
OPTION TCP))

Examples:

COMPONENT TRACE SHORT FORMAT
COMP(SYSTCPDA)SUBNAME((TCPCS))
OPTIONS((STAT(DETAIL) OPTION TCP))

1 DSNAME('IPCS.R744334.DUMPA')

2 OPTIONS((Both Bootp(67,68) Checksum(Summary) Cleanup(500) Datasize(0) DelayAck(200,200)
Domain(53) EE(12000:12004) Finger(79) Flags() Ftp(20,21) Gain(125,250) Gopher(78) Limit(999999999)
Gmt Ntp(123) Option Reassembly(65535,Summary) Router(520) Rpc(111) Sasp(3860) Segment Smtp(25)
Snmp(161,162) Speed(10,10) Statistics(Detail) Telnet(23,80,) Tftp(69) Time(37) Userexit() Www(80)
3 Protocol( /* 1 */ 6 /* TCP */ , ) ))

The following fields are on the OPTION report.

1 DSNAME - The name of the source data.

2 OPTIONS((...) - A listing of the active options with default values.

3 When a filter is specified, the list of filters with the number of filter values
and filter values.
SUMMARY:

**Purpose:** Show one or two lines of information about each record in the trace.

**Format:** `CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SUMMARY`

**Examples:** The following fields are on the SUMMARY report.

- **D** Direction of the packet:
  - I Inbound
  - O Outbound

- **P** The packet protocol
  - T TCP
  - U UDP
  - I ICMP
  - G IGMP

- **G** Data Trace

- **P** Neither TCP, UDP, ICMP, nor IGMP

- **Nr** The Ctrace record number

- **hh:mm:ss.mmmmmm** The time stamp of the record

- **Source** The source IP address and port number

- **Destination** The destination IP address and port number

- **lpld** The packet ID number in hexadecimal
  - For TCP

**** 2004/01/26
I - Inbound packet
O - Outbound packet

<table>
<thead>
<tr>
<th>dp</th>
<th>nr</th>
<th>hh:mm:ss.mmmmmm</th>
<th>lpid</th>
<th>seq num</th>
<th>ack num</th>
<th>wndw</th>
<th>flags</th>
<th>datln</th>
<th>source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>OT</td>
<td>1</td>
<td>14:18:00.447462</td>
<td>19E6</td>
<td>1452693653</td>
<td>0</td>
<td>32768</td>
<td>SYN</td>
<td>0</td>
<td>10.7.1.61-3470</td>
</tr>
<tr>
<td>IT</td>
<td>2</td>
<td>14:18:00.601784</td>
<td>4E3C</td>
<td>3454024895</td>
<td>1452693654</td>
<td>32768</td>
<td>ACK</td>
<td>0</td>
<td>192.168.248.44-5000</td>
</tr>
<tr>
<td>OT</td>
<td>3</td>
<td>14:18:00.601917</td>
<td>1A00</td>
<td>1452693654</td>
<td>3454024896</td>
<td>32768</td>
<td>ACK</td>
<td>0</td>
<td>10.7.1.61-3470</td>
</tr>
<tr>
<td>IT</td>
<td>4</td>
<td>14:18:01.111074</td>
<td>4E3D</td>
<td>3454024896</td>
<td>1452693654</td>
<td>32768</td>
<td>ACK</td>
<td>0</td>
<td>192.168.248.44-5000</td>
</tr>
<tr>
<td>IT</td>
<td>5</td>
<td>14:18:01.126148</td>
<td>4E3E</td>
<td>3454024943</td>
<td>1452693654</td>
<td>32768</td>
<td>ACK</td>
<td>0</td>
<td>10.7.1.61-3470</td>
</tr>
<tr>
<td>OT</td>
<td>6</td>
<td>14:18:01.126248</td>
<td>1A46</td>
<td>1452693654</td>
<td>3454025008</td>
<td>32703</td>
<td>ACK</td>
<td>0</td>
<td>192.168.248.44-5000</td>
</tr>
<tr>
<td>OT</td>
<td>7</td>
<td>14:18:03.290611</td>
<td>1B7F</td>
<td>1452693654</td>
<td>3454025008</td>
<td>32703</td>
<td>ACK</td>
<td>0</td>
<td>10.7.1.61-3470</td>
</tr>
<tr>
<td>IT</td>
<td>8</td>
<td>14:18:03.373175</td>
<td>4E3F</td>
<td>3454025008</td>
<td>1452693664</td>
<td>32768</td>
<td>ACK</td>
<td>0</td>
<td>192.168.248.44-5000</td>
</tr>
</tbody>
</table>

Figure 19. Example of a SUMMARY report
seq_num
The sequence number

ack_num
The acknowledgment sequence number

wndw
The window size

flags
The TCP header flags

DatL
The length of data in the datagram

EBCDIC
The first eight bytes with EBCDIC translation

ASCII
The first eight bytes with ASCII translation

• For UDP

nnnnn
The length of the UDP datagram

DatL
The length of the UDP packet data

EBCDIC
The first eight bytes with EBCDIC translation

ASCII
The first eight bytes with ASCII translation

• For ICMP

ccccccccccc
The type of ICMP message

xxxxxxxxxxx
The first eight bytes of the user data in hexadecimal

• For IGMP

nnnnn
The maximum response time

ccccccccccc
The type of IGMP message

nnn.nnn.nnn.nnn
The IGMP group address

• Other protocols

ccccccccccc
The protocol name

nnnnn
The length of the protocol data
EXPORT:

Purpose: Reformat the information about the IP header, protocol header, and packet data into a file with CSV format.

Format:

ALLOC FILE(EXPORT) DA(EXPORT.CSV) SPACE(15 15) TRACK
CTRACE COMP(SYSTCPDA) SUB((TCPSC)) SHORT OPTIONS((EXPORT))

Examples: The following describe the EXPORT, EXPORT(SUMMARY), and EXPORT(DETAIL) report outputs.

- **EXPORT**
  
  Export Report

  1. 124 records written to USER2.EXPORT.CSV
  2. 20,168 bytes written

  The following fields are on the EXPORT report.

  1. The number of data records written to the export data set.

  2. The size of the export data set.

- **EXPORT (SUMMARY)**

  "Flags ","Packet","Absolute Time ","Rel Time","Delta Time",
  "Device ","Source ","Destination ",
  "IpId","IpLen","Protocol ","Summary"

  "I O ",1,19:49:42.788207", 0.000000, 0.000000,
  "OSAQDIO1046 ","9.67.115.17 ","9.67.115.63 ",
  17158, 78,"UDP","S=137 D=137 LEN=58"

  "I O ", 29,19:52:21.240160",158.451952, 0.016739,
  "OSAQDIO1046 ","9.67.115.69 ","9.67.115.5 ",
  5971, 56,"ICMP","? LEN=28"

  "I O ",37,19:52:27.789344",164.995736, 0.000134,
  "LOOPBACK ","9.67.115.5 ","9.67.115.5 ",
  129, 56,"ICMP","? LEN=28"

  "O O ",40,19:52:39.28402",176.496595, 5.500260,
  "OSAQDIO1046 ","9.67.115.8 ","9.67.115.7 ",
  20, 60,"UDP","S=32810 D=33435 LEN=20"

  "O O ",41,19:52:39.28470",176.496662, 0.000067,
  "OSAQDIO1046 ","9.67.115.8 ","9.67.115.7 ",
  32, 72,"ICMPv6","ICMPv6"

  "I O ",42,19:52:39.28555",176.497748, 0.001085,
  "OSAQDIO1046 ","9.67.115.8 ","9.67.115.7 ",
  32, 72,"ICMPv6","ICMPv6"

  "O O ",49,19:52:58.28634",195.498140, 13.972912,
  "LOOPBACK6 ","9.67.115.8 ","9.67.115.7 ",
  20, 60,"UDP","S=32810 D=33435 LEN=20"

  "I O ",50,19:52:58.28653",195.498323, 0.000182,
  "LOOPBACK6 ","9.67.115.8 ","9.67.115.7 ",

  The following describes fields found on the EXPORT (SUMMARY) report:

  **Control flags**

  **Direction**

  - I — Input
  - O — Output

  A  The packet was abbreviated (used with the following fragment flags).
R  Reassembled packet.
O  The Only fragment of a packet (it is complete).
F  First fragment of a packet.
M  Middle fragment of a packet.
L  Last fragment of a packet.
T  The packet was in a tunnel.

Packet
The packet number

Absolute Time
The time stamp on the packet

Rel Time
The time from the first packet in seconds

Delta Time
The time from the previous packet in seconds

Device
The device the packet was received on or sent from

Source
The source IP address

Destination
The destination IP address

IpId
The ID number from the IP packet header

IpLen
The length of the IP packet

Protocol
The protocol from the IP packet

Summary
Additional information from the protocol header.

- EXPORT (DETAIL)

The following describes fields found on the EXPORT (DETAIL) report:

Control flags

Direction
– I — Input
– O — Output

A  The packet was abbreviated (used with the following fragment flags).
   R  Reassembled packet.
   O  Only fragment of a packet (it is complete).
   F  First fragment of a packet.
   M  Middle fragment of a packet.
   L  Last fragment of a packet.
T  The packet was in a tunnel.

Packet
   The packet number.

Delta Time
   The time from the previous packet in seconds.

Source
   The source IP address.

Destination
   The destination IP address.

Protocol
   There are multiple lines about a single packet. The first line contains "IP" to
   identify the data in the summary field. The second line identifies
   information about the protocol used by the packet. The possible third line
   identifies the application data in the packet.

Summary
   Additional information from the protocol headers or packet data.
FORMAT:

Purpose: Format the Ctrace record header, the IP packet header, the protocol header, and the packet data. If one of the ports is a well-known port number and the SYSTCPDA supports data for the port number, the packet data is shown.

Format:
CTRACE COMP(SYSCTCPDA) SUB((TCPICS)) SHORT OPTIONS((FORMAT))

Examples:
1 3 MVSJ PACKET 0000001 23:39:11.873541 Packet Trace
2 To Interface : TR1 Device: LCS Token Ring Full=56
   Tod Clock : 2002/02/12 23:39:11.873539
   Sequence #: 0 Flags: Pkt Ver2 Out
   Source Port : 1025 Dest Port: 53 Asid: 001E TCB: 007F62C0
3 IpHeader: Version : 4 Header Length: 20
   Tos : 00 QOS: Routine Normal Service
   Packet Length : 56 ID Number: 000E
   Fragment : Offset: 0
4   TTL : 64 Protocol: UDP CheckSum: A6FB FFFF
   Source : 9.67.113.65 Destination : 9.37.80.3

5 UDP
   Source Port : 1025 () Destination Port: 53 (domain)
   Datagram Length : 36 CheckSum: AD0B FFFF
6 DNS: 28

===============================================================================
7 ;; -->DNS HEADER<<- opcode: QUERY, status: NOERROR, id: 40266
    ;; flags: rd; Ques: 1, Ans: 0, Auth: 0, Addit: 0

    ;; QUESTIONS: 1
    ;; w3.ibm.com IN AAAA

1 A summary line indicating the source of the trace record showing:
   • The record number.
   • The system name.
   • The type of the trace record.
   • The time the record was moved to the trace buffer, or with the TOD option the time the trace data was captured.
   • The description of the trace record, Packet Trace, X25, or Data Trace.

2 The trace header with these fields:
   • The direction of the trace record: From or To.
   • The network interface name (or job name for Data Trace).
   • The device type.
   • Full or Abbrev with amount of trace data available.
   • The time the trace record was captured.
   • The number of records lost.
   • The packet trace header flags.

3 The IP header showing fields from the IPv4 header. The header length is the number of bytes for the header. The offset field is the number of bytes from
the end of the IP header where the fragment appears. With the REASSEMBLY option active, this field always contains zeros.

4 The check sum value. If possible, the check sum of the packet is calculated. If the calculated value is X'FFFF', the check sum is correct. If X'0000', the check sum could not be calculated because the packet was incomplete or fragmented. Other values indicate a check sum error.

5 The UDP protocol header. The fields of the header are shown.

6 The length of the DNS packet data following is shown.

7 The DNS header and resource records are formatted. Using the protocol numbers and the well known port numbers, format routines are invoked to format standard packet data records. The port number for the PORT keywords define the port numbers to be used to invoke a format routine.

<table>
<thead>
<tr>
<th>Port</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>67, 68</td>
<td>BOOTP</td>
</tr>
<tr>
<td>67, 68</td>
<td>DHCP</td>
</tr>
<tr>
<td>53</td>
<td>Domain</td>
</tr>
<tr>
<td>520</td>
<td>RIP</td>
</tr>
<tr>
<td>520</td>
<td>Router</td>
</tr>
<tr>
<td>161,162</td>
<td>SNMP</td>
</tr>
<tr>
<td>23</td>
<td>TELNET</td>
</tr>
<tr>
<td>69</td>
<td>TFTP</td>
</tr>
</tbody>
</table>


1 17 MVSN PACKET 00000004 19:43:02.541728 Packet Trace
To Interface: LOGETH5 Device: QDIO Ethernet Full=6300
Tod Clock: 2004/10/18 19:43:02.541728 Intfx: 5
Sequence #: 0 Flags: Pkt Out Offl

3 IpHeader: Version: 4 Header Length: 20
Tos: 00 QoS: Routine Normal Service
Offload Length: 6300 ID Numbers: 0012-0016
Fragment: Offset: 0

4 TTL: 64 Protocol: TCP CheckSum: 0000 971D
Source: 8.1.1.1 Destination: 8.1.1.2

5 TCP Source Port: 1026 () Destination Port: 1026 ()
Sequence Number: 3823117120 Ack Number: 3823533758
Header Length: 32 Flags: Ack Psh
Window Size: 32768 CheckSum: 1200 0000 Urgent Data Pointer: 0000
Offload Segments: 4 Length: 1448 Last: 456
Option: NOP Option: NOP Option: Timestamp Len: 10 Value: F3913448 Echo: F3913446

Figure 20. Format report example

A summary line indicating the source of the trace record showing:
- The record number.
- The system name.
- The type of the trace record.
- The time the record was moved to the trace buffer, or with the TOD option the time the trace data was captured.
- The description of the trace record, Packet Trace or Data Trace.

The trace header with these fields:
- The direction of the trace record: From or To.
- The network interface name (or job name for Data Trace).
- The device type.
- Full or Abbrev with amount of trace data available.
- The time the trace record was captured.
- The number of records lost.
- The packet trace header flags.

The IP header showing fields from the IPv4 header. The header length is the number of bytes for the header. The offset field is the number of bytes from the end of the IP header where the fragment appears. With the REASSEMBLY option active, this field always contains zeros. If segmentation is offloaded, the ID number field shows the range of IP identifiers represented by this send and the Offload Length field shows the total length of the send (total data length plus one set of headers).

The check sum value. If possible, the check sum of the packet is calculated. If the calculated value is X’FFFF’, the check sum is correct. If X’0000’, the check sum could not be calculated because the packet was incomplete or fragmented. Other values indicate a check sum error.
The TCP protocol header. The fields of the header are shown. If segmentation is offloaded, the Offload Segments field shows the number of TCP segments represented by this send and the length of each segment. The length of each segment is the data length (not including headers). If all the segments are the same size, then the Last field does not appear. If the remainder of data length is nonzero, then Last field contains the remainder.
DUMP:

Purpose:  Format the IP header, protocol header, and packet data in hexadecimal. The data can also be translated into EBCDIC, ASCII, or both.

Format:
CTRACE COMP(SYSCTPDA) SUB((TCP)) SHORT OPTIONS((DUMP))

Examples:
1 MV5073 PACKET 00000001 19:49:42.788207 Packet Trace
   From Interface : OSAQDOLINK   Device: QDIO Ethernet   Full=78
   Tod Clock : 2002/02/12 19:49:42.788204
   Sequence # : 0   Flags: Pkt Ver2
   Source Port : 137   Dest Port: 137   Asid: 002B TCB: 00000000
   IP Header : 20
   000000 4500004E 43060000 8011FEC2 09437311 0943733F
   Protocol Header : 8
   000000 00890089 003AD7D7
   Data : 50 Data Length: 50
   000000 AD3D0110 00010000 00000000 20464845 |................ .=.......... FHE|
   000010 50464345 46454846 46464646 46644143 |&...<.....&..... PFCELEFCEPFFFFAC|
   000020 41434143 41434143 41434142 4LC00020 |............<... ACACACACACABL.. |
   000030 0001  ..  ..

1 The IP header is dumped with no translation.
2 The protocol header is dumped with no translation.
3 The packet data is dumped with the translation specified by the ASCII, BOTH, EBCDIC or HEX keyword. The default is BOTH. The amount of data dumped can be limited by the value specified with the DUMP keyword. The default is 65535 bytes.
REASSEMBLY:

Purpose: This report shows the packets that were reassembled. Use the REASSEMBLY(DETAIL) option to see all the packets that were reassembled. If the reassembled packets are larger than 32K then use REASSEMBLY(????????), where ????? is the maximum size of a reassembled packet.

Format:
CTRACE COMP(SYSTCPDA) SUB((TCPSC)) SHORT OPTIONS((REASSEMBLY(DETAIL) STAT))

Examples:

<table>
<thead>
<tr>
<th>Rcd Nr</th>
<th>Time</th>
<th>Delta</th>
<th>Offset</th>
<th>Length</th>
<th>Next</th>
<th>Gap</th>
<th>Data</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1638</td>
<td>15:28:49.975479 00:00:00.000000</td>
<td>0</td>
<td>3976</td>
<td>3976</td>
<td>0</td>
<td>3976</td>
<td>Fis</td>
<td></td>
</tr>
<tr>
<td>1639</td>
<td>15:28:49.975501 00:00:00.000022</td>
<td>3976</td>
<td>3976</td>
<td>7952</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1640</td>
<td>15:28:49.975524 00:00:00.000044</td>
<td>7952</td>
<td>3976</td>
<td>11928</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1641</td>
<td>15:28:49.975545 00:00:00.000066</td>
<td>11928</td>
<td>3976</td>
<td>15904</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1642</td>
<td>15:28:49.975567 00:00:00.000088</td>
<td>15904</td>
<td>3976</td>
<td>19880</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1643</td>
<td>15:28:49.975594 00:00:00.000115</td>
<td>19880</td>
<td>3976</td>
<td>23856</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1644</td>
<td>15:28:49.975620 00:00:00.000141</td>
<td>23856</td>
<td>3976</td>
<td>27832</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1645</td>
<td>15:28:49.975649 00:00:00.000171</td>
<td>27832</td>
<td>3976</td>
<td>31808</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1646</td>
<td>15:28:49.975737 00:00:00.000258</td>
<td>31808</td>
<td>3976</td>
<td>35784</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1647</td>
<td>15:28:49.975771 00:00:00.000292</td>
<td>35784</td>
<td>3976</td>
<td>39760</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1648</td>
<td>15:28:49.975804 00:00:00.000325</td>
<td>39760</td>
<td>3976</td>
<td>43736</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1649</td>
<td>15:28:49.975835 00:00:00.000356</td>
<td>43736</td>
<td>3976</td>
<td>47712</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1650</td>
<td>15:28:49.975865 00:00:00.000386</td>
<td>47712</td>
<td>3976</td>
<td>51688</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1651</td>
<td>15:28:49.975898 00:00:00.000419</td>
<td>51688</td>
<td>3976</td>
<td>55664</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1652</td>
<td>15:28:49.975926 00:00:00.000447</td>
<td>55664</td>
<td>3976</td>
<td>59640</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1653</td>
<td>15:28:49.975962 00:00:00.000483</td>
<td>59640</td>
<td>3976</td>
<td>63616</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>1654</td>
<td>15:28:49.975986 00:00:00.000507</td>
<td>63616</td>
<td>392</td>
<td>64008</td>
<td>0</td>
<td>392</td>
<td>Lis</td>
<td></td>
</tr>
</tbody>
</table>

17 packets were used for reassembly
64,008 bytes is the final length of the IP packet

Packet Reassembly Report
Maximum reassembly buffer size is 65535

Reassembly of: 9.67.113.65-0 9.27.11.173-0 Id: 0043 status: +Fis +Lis

<table>
<thead>
<tr>
<th>Rcd Nr</th>
<th>Time</th>
<th>Delta</th>
<th>Offset</th>
<th>Length</th>
<th>Next</th>
<th>Gap</th>
<th>Data</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>15:28:49.975479 00:00:00.000000</td>
<td>0</td>
<td>3976</td>
<td>3976</td>
<td>0</td>
<td>3976</td>
<td>Fis</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>15:28:49.975501 00:00:00.000022</td>
<td>3976</td>
<td>3976</td>
<td>7952</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>15:28:49.975524 00:00:00.000044</td>
<td>7952</td>
<td>3976</td>
<td>11928</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>15:28:49.975545 00:00:00.000066</td>
<td>11928</td>
<td>3976</td>
<td>15904</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>15:28:49.975567 00:00:00.000088</td>
<td>15904</td>
<td>3976</td>
<td>19880</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>15:28:49.975594 00:00:00.000115</td>
<td>19880</td>
<td>3976</td>
<td>23856</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>15:28:49.975620 00:00:00.000141</td>
<td>23856</td>
<td>3976</td>
<td>27832</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>15:28:49.975649 00:00:00.000171</td>
<td>27832</td>
<td>3976</td>
<td>31808</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>15:28:49.975737 00:00:00.000258</td>
<td>31808</td>
<td>3976</td>
<td>35784</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>15:28:49.975771 00:00:00.000292</td>
<td>35784</td>
<td>3976</td>
<td>39760</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>15:28:49.975804 00:00:00.000325</td>
<td>39760</td>
<td>3976</td>
<td>43736</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>15:28:49.975835 00:00:00.000356</td>
<td>43736</td>
<td>3976</td>
<td>47712</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>15:28:49.975865 00:00:00.000386</td>
<td>47712</td>
<td>3976</td>
<td>51688</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>15:28:49.975898 00:00:00.000419</td>
<td>51688</td>
<td>3976</td>
<td>55664</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>15:28:49.975926 00:00:00.000447</td>
<td>55664</td>
<td>3976</td>
<td>59640</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>15:28:49.975962 00:00:00.000483</td>
<td>59640</td>
<td>3976</td>
<td>63616</td>
<td>0</td>
<td>3976</td>
<td>Mis</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>15:28:49.975986 00:00:00.000507</td>
<td>63616</td>
<td>392</td>
<td>64008</td>
<td>0</td>
<td>392</td>
<td>Lis</td>
<td></td>
</tr>
</tbody>
</table>

64,008 bytes is the final length of the IP packet
17 packets were used for reassembly
64,008 bytes were accumulated for reassembly
64,008 bytes is the final length of the IP packet
41 packets were used for reassembly
59,568 bytes were accumulated for reassembly

1,641 packets required reassembly
54 IP packet reassemblies were done
52 IP packets were completely reassembled
2 IP packets were incomplete
0 packets failed reassembly
1,627 storage requests for buffers were made
64,080 bytes of buffer space are still in use
191,872 bytes of buffer space was the maximum in use
114,688 bytes of control storage were used

Reassembly is always done (except with the NOREASSEMBLY option). However, the REASSEMBLY(DETAIL) option is needed for the report on completed reassemblies.

1. The current packet that was reassembled is identified with source and destination IP address and port numbers. The IP identification number is shown. The status of the reassembly is shown. Completed packets are shown when the final packet is received. Incomplete packets are shown during the final processing.

2. Each packet that was reassembled is shown. The flag shows the type of packet:
   - *Fis* First in the segment. The offset was 0.
   - *Mis* Middle in the segment. The offset was a nonzero value and the more fragment flag was set.
   - *Lis* Last in the segment. The offset was nonzero and the more fragment flag was not set.
   - *Ooo* The packet arrived out of order.

The Gap field is the number of bytes between the end of one packet and the start of the next. This should have a value of zero for normal processing. Nonzero values indicate duplicate data being sent.

3. When all the trace records have been processed the final report on reassembly is formatted. The maximum reassembly buffer size is shown. Packets that would exceed this size are rejected. This simulates the Ping of Death processing.

4. Incomplete packets that did not complete reassembly are shown.
The total number of trace records that were reassembled is shown with other statistics.

200 packets required reassembly
The number of packets that required reassembly (that had a fragment offset or the more fragment flag set).

57 IP packet reassemblies were done
The number of reassembled packets.

54 IP packets were completely reassembled
The number of reassembled packets where all the fragments were found.

3 IP packets were incomplete
The number of reassembled packets where all the fragments were not found.

0 packets failed reassembly
The number of packets that would have caused the completed packet to exceed the reassembly size.

170 storage requests for buffers were made
The number of times a request for reassembly buffer was made.

128,747 bytes of buffer space is still in use
The amount of storage still in use for incomplete packets.

284,158 bytes of buffer space was the maximum in use
The maximum amount of storage in use while reassembling packets.

Guideline: For reassembled packets, the calculated check sum fields are not X'FFFF', because the packets were modified by the reassembly process.
SESSION:

Purpose: This report shows traffic for a TCP session.

Format:
CTRACE COMP(SYSTCPDA) SUB((TCPICS)) SHORT OPTIONS((SESSION TCP))

Examples:
1 2 packets summarized
Local IP Address: FEC9:C204::6:2900:EDC:217C
Remote IP Address: FEC9:C204::9:67:115:17
Host: Local, Remote
Client or Server: CLIENT, SERVER
Port: 1027, 21
Application: ftp
Link speed (parm): 10, 10 Megabits/s
2 Connection:
First timestamp: 19:55:46.934032
Last timestamp: 19:55:46.934989
Duration: 00:00:00.000957
Average Round-Trip-Time: 0.000 sec
Final Round-Trip-Time: 0.000 sec
Final state: CLOSED (PASSIVE RESET)
Out-of-order timestamps: 0
3 Data Quantity & Throughput: Inbound, Outbound
Application data bytes: 0, 0
Sequence number delta: 0, 1
Total bytes Sent: 0, 0
Bytes retransmitted: 0, 0
Throughput: 0, 0 Kilobytes/s
Bandwidth utilization: 0.00%, 0.00%
Delay ACK Threshold: 200, 200 ms
Minimum Ack Time: 0.000957, 0.000000
Average Ack Time: 0.000957, 0.000000
Maximum Ack Time: 0.000957, 0.000000
4 Data Segment Stats: Inbound, Outbound
Number of data segments: 0, 0
Maximum segment size: 1432, 0
Largest segment size: 0, 0
Average segment size: 0, 0
Smallest segment size: 0, 0
Segments/window: 0.0, 0.0
Average bytes/window: 0, 0
Most bytes/window: 0, 0
Offload Sends: 3 (50%)
Offload Segments: 6
Offload Bytes: 43616 (72.69%)
Total Packets(normal + offload): 18 (83.33%)
5 Window Stats: Inbound, Outbound
Number of windows: 0, 0
Maximum window size: 0, 0
Largest window advertised: 0, 32768
Average window advertised: 0, 32768
Smallest window advertised: 0, 32768
Window scale factor: 0, 0
Window frequency: 0, 0 Windows/s
Time Stamp updates: 0, 0
Total Round Trip Time: 0.000000, 0.000000 (0%), (0%)
Average Round Trip Time: 0.000000, 0.000000
6 Number of: Inbound, Outbound
Packets:
(a) Untraced Packets: 0, 0
(b) In-order data: 0, 0 (0.00%), (0.00%)
(c) Acknowledgments: 0, 0 (100.00%), (0.00%)
(d) Data and ACK: 0, 0 (0.00%), (0.00%)
(e) Duplicate ACKs: 0, 0 (0.00%), (0.00%)
(f) Window size updates: 0, 0 (0.00%), (0.00%)
(g) Zero window sizes: 0, 0 (0.00%), (0.00%)
(h) Window probes: 0, 0 (0.00%), (0.00%)
(i) Keepalive segments: 0, 0 (0.00%), (0.00%)
(j) Retransmissions: 0, 0 (0.00%), (0.00%)
(k) Out-of-order: 0, 0 (0.00%), (0.00%)
(l) Delayed ACKs: 0, 0 (0.00%), (0.00%)
(m) Fragments: 0, 0 (0.00%), (0.00%)
7 Time Spent on: Inbound, Outbound
(a) In-order data: 00:00:00.000000, 00:00:00.000000 (0.00%), (0.00%)
(b) Acknowledgments: 00:00:00.000957, 00:00:00.000000 (106.33%), (0.00%)

Chapter 5. TCP/IP services traces and IPCS support 135
1 Host

The number of packets records for this session; the IP addresses and port of the session.

2 Connection

The first and last time of the session, the length of the session, the final value of RRT, and the final state of the session.

3 Data Quality & Throughput

These statistics are about the quantity of data transmitted. The number of bytes received inbound and the number of bytes send outbound.

4 Data Segment Stats

These statistics are about the segments, the number of segments, and the sizes of the segments. The maximum segment size is captured from the SYN packet. Offload statistics appear only when there were any offloaded packets. These values reflect the number of offload packets, the number segments in these offloaded packets, the number of bytes in offloaded packets, and the total number of segments sent from the interface.

5 Window Stats

These statistics are about the window changes. The Window scale factor is captured from the SYN packet. The Time Stamp updates are captured from the Tcp header options.

6 Number of Packets

---

The largest inbound window is less than twice the inbound MSS.
2 This may reduce inbound throughput for bulk data transfers.
3 It is usually desirable for the window size to be at twice the MSS.

---

## Host

### Number of Packets

<table>
<thead>
<tr>
<th>Inbound</th>
<th>Outbound</th>
</tr>
</thead>
<tbody>
<tr>
<td>136</td>
<td></td>
</tr>
</tbody>
</table>

## Messages

2 The largest inbound window is less than twice the inbound MSS.
2 This may reduce inbound throughput for bulk data transfers.
3 The outbound side of the connection appears to be a bulk data transfer.
These statistics are about the number of data packets that flow for carrying data. The percentages are based on the number of packets.

Time Spent on:
These statistics are about the delta times of data packets that flow for carrying data. The percentages are based on the duration of the session.

Number of
These statistics are about the number of control packets that flow for starting and ending a session. The percentages are based on the number of packets.

Time Spent on
These statistics are about the delta times of control packets that flow for starting and ending a session. The percentages are based on the duration of the session.

Details TcpHdr
The flags from the TCP header

* This packet has been reassembled.
A This packet is an acknowledgment.
P This packet has the PUSH flag set.
U This packet is urgent.
S This packet is a syn.
F This packet is a fin.
R This packet is a reset.

The type of data packet has one of the following flags:
.
The packet flowed in order with respect to its sequence number.
X There is a gap in the sequence number and there appears to be untraced data.
a The packet is a stand-alone acknowledgment of previously received data.
+ The packet is an acknowledgment of previously received data and also contains data.
U The packet is an acknowledgment of data previously acknowledged.
w The packet updated the window size.
z The packet changed the window size to zero.
p The packet was a window probe.
k The packet was a keepalive packet.
r The packet was retransmission.
o The packet arrived out of order.
d The packet exceeded the delay time threshold.
The packet was a fragment of a complete IP packet.

A dropped packet that had a checksum error, that was a fragment, or that was discarded.
SNIFFER:

Purpose: This report shows information written to the SNIFFER data set.

Format:
ALLOC F(SNIFFER) DATASET(SNIFFER.TRC) LRECL(1600) RECFM(V B) REUSE TRACK SPACE(15 15)
CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((SNIFFER NOREASSEMBLY STATS))

Examples:

Interface Table Report
<table>
<thead>
<tr>
<th>Index</th>
<th>Count</th>
<th>Link</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>OSAQDIO1LINK</td>
<td>9.67.115.63</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>LOOPBACK</td>
<td>127.0.0.1</td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>OSAQDIO1LINK</td>
<td>9.67.115.5</td>
</tr>
<tr>
<td>4</td>
<td>31</td>
<td>OSAQDIO16</td>
<td>FEC9:C2D4::6:2900:EDC:217C</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>OSAQDIO16</td>
<td>FE80::6:2900:EDC:217C</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>LOOPBACK6</td>
<td>::1</td>
</tr>
</tbody>
</table>

Sniffer Report
1 13,963 records written to USER2.SNIFFER.ETH
2 1,730,182 bytes written
3 1184 packets were abbreviated
4 200 is the maximum data size
5 12438 packets were truncated from 1546 bytes

The list of device names found in the selected records. Each device is assigned an interface index.

This record count includes the two header records and one trailer record written to the SNIFFER data set.

The number of data bytes written to the SNIFFER data set. This is the amount of data to be downloaded.

The number of abbreviated records. This number is included in 4.

Maximum size of the truncated records.

The number of truncated records. Records were truncated because the size of the packet exceeded the logical record length of the SNIFFER file. Increase the logical record length to prevent the records from being truncated. The maximum logical record length is 32,763 or the size of physical disk blocks, whichever is smaller.
**STATISTICS:**

*Purpose:* The records are counted by record type, device type, device name, job name, protocol, IP address, TCP port number, and UDP port number.

*Format:*

CTRACE COMP(SYSCTPDA) SUB((TCPCS)) SHORT OPTIONS((STATISTICS(DETAIL)))

*Examples:*

1. No packets required reassembly

---

**SYSTCPDA Trace Statistics**

123 ctrace records processed
0 segmented trace records read
123 trace records read
0 records could not be validated
123 records passed filtering
123 packet trace records processed
0 data trace records processed

---

**Record Type Report**

<table>
<thead>
<tr>
<th>Type</th>
<th>Input</th>
<th>Output</th>
<th>First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
<th>Record Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>55</td>
<td>3814</td>
<td>10</td>
<td>2002/02/12 19:49:42</td>
<td>1(Packet Trace)</td>
</tr>
<tr>
<td>58</td>
<td>19</td>
<td>1828</td>
<td>39</td>
<td>2002/02/12 19:52:39</td>
<td>4(Packet Trace)</td>
</tr>
<tr>
<td>123</td>
<td>74</td>
<td>5642</td>
<td>49</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 Record Type(s) found

---

**Ip Version Report**

<table>
<thead>
<tr>
<th>Version</th>
<th>Input</th>
<th>Output</th>
<th>First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
<th>Ip Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>55</td>
<td>3814</td>
<td>10</td>
<td>2002/02/12 19:49:42</td>
<td>4</td>
</tr>
<tr>
<td>58</td>
<td>19</td>
<td>1828</td>
<td>39</td>
<td>2002/02/12 19:52:39</td>
<td>6</td>
</tr>
<tr>
<td>123</td>
<td>74</td>
<td>5642</td>
<td>49</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 Ip Version(s) found

---

**Device Type Report**

<table>
<thead>
<tr>
<th>Type</th>
<th>Input</th>
<th>Output</th>
<th>First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
<th>Device Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>42</td>
<td>42</td>
<td>2667</td>
<td>0</td>
<td>2002/02/12 19:49:48</td>
<td>Loopback</td>
</tr>
<tr>
<td>23</td>
<td>13</td>
<td>1147</td>
<td>10</td>
<td>2002/02/12 19:49:42</td>
<td>Loopback</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>324</td>
<td>3</td>
<td>2002/02/12 19:52:39</td>
<td>Loopback</td>
</tr>
<tr>
<td>52</td>
<td>16</td>
<td>1504</td>
<td>36</td>
<td>2660</td>
<td>Loopback</td>
</tr>
<tr>
<td>123</td>
<td>74</td>
<td>5642</td>
<td>49</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 Device Type(s) found

---

**Interface Report**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Input</th>
<th>Output</th>
<th>First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOOPBACK</td>
<td>42</td>
<td>2667</td>
<td>0</td>
<td>2002/02/12 19:49:48</td>
<td>Loopback</td>
</tr>
<tr>
<td>OSAQDIO4</td>
<td>6</td>
<td>324</td>
<td>3</td>
<td>2002/02/12 19:52:39</td>
<td>Loopback</td>
</tr>
<tr>
<td>OSAQDIO6</td>
<td>52</td>
<td>1504</td>
<td>36</td>
<td>2660</td>
<td>Loopback</td>
</tr>
<tr>
<td>123</td>
<td>74</td>
<td>5642</td>
<td>49</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 Interface(s) found

---

**Interface Address Report**

<table>
<thead>
<tr>
<th>Address</th>
<th>Input</th>
<th>Output</th>
<th>First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
<th>Interface Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.67.115.63</td>
<td>5</td>
<td>699</td>
<td>0</td>
<td>2002/02/12 19:49:42</td>
<td>OSAQDIO4</td>
</tr>
<tr>
<td>9.67.115.5</td>
<td>42</td>
<td>2667</td>
<td>0</td>
<td>2002/02/12 19:49:48</td>
<td>OSAQDIO4</td>
</tr>
<tr>
<td>9.67.115.5</td>
<td>18</td>
<td>448</td>
<td>10</td>
<td>2002/02/12 19:51:17</td>
<td>OSAQDIO4</td>
</tr>
<tr>
<td>9.67.115.5</td>
<td>31</td>
<td>1360</td>
<td>17</td>
<td>2002/02/12 19:52:39</td>
<td>OSAQDIO4</td>
</tr>
<tr>
<td>9.67.115.5</td>
<td>21</td>
<td>144</td>
<td>19</td>
<td>2002/02/12 19:52:44</td>
<td>OSAQDIO4</td>
</tr>
<tr>
<td>9.67.115.5</td>
<td>6</td>
<td>324</td>
<td>3</td>
<td>2002/02/12 19:52:58</td>
<td>OSAQDIO4</td>
</tr>
<tr>
<td>9.67.115.5</td>
<td>123</td>
<td>74</td>
<td>5642</td>
<td>49</td>
<td>OSAQDIO4</td>
</tr>
</tbody>
</table>

6 Interface Address(s) found

---

**Asid Report**

<table>
<thead>
<tr>
<th>Asid</th>
<th>Input</th>
<th>Output</th>
<th>First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
<th>Asid</th>
</tr>
</thead>
<tbody>
<tr>
<td>002A</td>
<td>34</td>
<td>0</td>
<td>04</td>
<td>2002/02/12 19:55:48</td>
<td>002A</td>
</tr>
<tr>
<td>002B</td>
<td>89</td>
<td>74</td>
<td>5642</td>
<td>1</td>
<td>002B</td>
</tr>
<tr>
<td>002B</td>
<td>123</td>
<td>74</td>
<td>5642</td>
<td>49</td>
<td>002B</td>
</tr>
</tbody>
</table>

2 Asid(s) found

---

**Protocol Report**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Input</th>
<th>Output</th>
<th>First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMP</td>
<td>30</td>
<td>29</td>
<td>1</td>
<td>2002/02/12 19:49:48</td>
<td>ICMP</td>
</tr>
<tr>
<td>TCP</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2002/02/12 19:55:20</td>
<td>TCP</td>
</tr>
<tr>
<td>UDP</td>
<td>53</td>
<td>26</td>
<td>27</td>
<td>2002/02/12 19:49:42</td>
<td>UDP</td>
</tr>
<tr>
<td>ICM Puerto</td>
<td>36</td>
<td>17</td>
<td>19</td>
<td>2002/02/12 19:52:39</td>
<td>ICM Puerto</td>
</tr>
</tbody>
</table>

2 Asid(s) found
### Protocol(s) found

**IP Address Report**

<table>
<thead>
<tr>
<th>Protocol(s)</th>
<th>First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Protocol(s) found</td>
<td>198002/02/12 19:51:17</td>
<td>19802/02/12 19:52:21</td>
</tr>
<tr>
<td>123 74 5642 49</td>
<td>Addr: 9.67.115.1</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>Addr: 9.67.115.5</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>Addr: 9.67.115.17</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>Addr: 9.67.115.63</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Addr: 9.67.115.69</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Addr: 9.67.115.69</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Addr: 9.67.115.17</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Addr: 9.67.115.63</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Addr: 9.67.115.69</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Addr: 9.67.115.69</td>
<td></td>
</tr>
</tbody>
</table>

### Qos Report

<table>
<thead>
<tr>
<th>Qos(s) found</th>
<th>First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 IP Address(s) found</td>
<td>2002/02/12 19:51:54</td>
<td>2002/02/12 19:55:09</td>
</tr>
<tr>
<td>3 0 0 3 160 28</td>
<td>Addr: 9.67.115.17</td>
<td></td>
</tr>
<tr>
<td>2 0 0 2 100 30</td>
<td>Addr: 9.67.115.63</td>
<td></td>
</tr>
<tr>
<td>1 0 0 1 40 32</td>
<td>Addr: 9.67.115.69</td>
<td></td>
</tr>
</tbody>
</table>

### Tcp Port Report

| Tcp Port(s) found | Total Input Data Output Data First yyyy/mm/dd hh:mm:ss Last yyyy/mm/dd hh:mm:ss |
|-------------------|------------------------------------------------------------------|-----------------------------|
| 3 0 0 3 28 | Addr: 9.67.115.17 |
| 2 1 60 | Addr: 9.67.115.63 |
| 1 0 0 1 40 | Addr: 9.67.115.69 |

### Udp Port Report

| Udp Port(s) found | Total Input Data Output Data First yyyy/mm/dd hh:mm:ss Last yyyy/mm/dd hh:mm:ss |
|-------------------|------------------------------------------------------------------|-----------------------------|
| 3 0 0 3 28 | Addr: 9.67.115.17 |
| 2 1 60 | Addr: 9.67.115.63 |
| 1 0 0 1 40 | Addr: 9.67.115.69 |

### Protocol Summary Report

Chapter 5. TCP/IP services traces and IPCS support

141
<table>
<thead>
<tr>
<th>Protocol</th>
<th>Packets</th>
<th>Bytes</th>
<th>Packets</th>
<th>Bytes</th>
<th>Packets</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tcp</td>
<td>2</td>
<td>120</td>
<td>2</td>
<td>160</td>
<td>4</td>
<td>280</td>
</tr>
<tr>
<td>Udp</td>
<td>26</td>
<td>2190</td>
<td>27</td>
<td>1440</td>
<td>53</td>
<td>3630</td>
</tr>
<tr>
<td>Icmp</td>
<td>46</td>
<td>3332</td>
<td>20</td>
<td>1884</td>
<td>66</td>
<td>5216</td>
</tr>
<tr>
<td>Other</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Session Summary Report**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Packets</th>
<th>Bytes</th>
<th>Packets</th>
<th>Bytes</th>
<th>Packets</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tcp</td>
<td>2</td>
<td>120</td>
<td>2</td>
<td>160</td>
<td>4</td>
<td>280</td>
</tr>
<tr>
<td>Udp</td>
<td>26</td>
<td>2190</td>
<td>27</td>
<td>1440</td>
<td>53</td>
<td>3630</td>
</tr>
<tr>
<td>Icmp</td>
<td>46</td>
<td>3332</td>
<td>20</td>
<td>1884</td>
<td>66</td>
<td>5216</td>
</tr>
<tr>
<td>Other</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The standard statistics shown with all executions of the SYSTCPDA packet trace formatter.

**Ctrace records processed**
The total number of Ctrace records given to the SYSTCPDA packet trace formatter by IPCS.

**segmented trace records read**
The total number of packets that spanned multiple Ctrace records.

**segmented trace records were lost**
The total number of segmented packets records that could not be put back together.

**trace records read**
The total number of complete trace records.

**records could not be validated**
The number of incomplete Ctrace records that could not be used.

**records passed filtering**
The number of records that were successfully formatted.

**packet trace records processed**
The number of records that were packet trace records.

**data trace records processed**
The number of records that were data trace records.
The totals by record type (Packet trace, X25, and data trace).

The totals by device type for packet trace records.

The totals by Interface or Link Name for packet trace records.

The totals by Protocol number for packet trace records.

The totals by IP Address. Both the destination and source IP addresses are counted except when they are the same within a record.

The totals by TCP Port number. Both the destination and source port numbers are counted except when they are the same within a record.

The totals by UDP port number. Both the destination and source port numbers are counted except when they are the same within a record.

Restriction: Reports 2 through 8 are shown only when STATISTICS(DETAIL) is specified in the OPTIONS string.

The totals by session partner pairs (IP addresses, protocol number, and port numbers).

The number of records processed for the statistics report.

The time stamp of the first record in the input file, the time stamp of the last record in the input, and the duration from the first to last record.

Tip: Records that have been abbreviated are not shown in this example. The number of records that were abbreviated and the maximum abbreviated size are shown. Also, the number and maximum size of the records that were not abbreviated are shown.

The size of the largest packet found in the input file.

The number of records processed for the statistics report, the number of 1KB blocks of storage required for this report, the number of storage requests, and the total amount of storage required for the requests.

The report by Jobname for data trace records is not shown. Each category of totals is broken down by:

- The total number of records
- The total number of inbound records
- The total amount of inbound data
- The total number of outbound records
- The total amount of outbound data
- The record number of the first record
- The time stamp of the first record
- The record number of the last record
• The time stamp of the last record
STREAM:

Purpose: Sometimes messages span multiple packets. TELNET and DNS are examples. The STREAM report (with the DUMP or FORMAT keywords) capture the entire stream of data.

Format:
CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((STREAM DUMP ASCII))

Examples:

1 Streams Report
60 Streams found
23600 bytes of storage for the session report was allocated
1146880 bytes of storage for buffers was allocated

3 packets found
Stream buffer at 0A818000 for 20480 bytes. 144 bytes were used
3 packets moved for 144 bytes
I - Inbound packet
O - Outbound packet

3 Rcd # Time Delta Seq # Position Length End_Pos
1 90 19:55:04.615118 00:00:00.000000 0 0 24 24
000000 FEC9C2D4 00000000 02062AFF FE714400 02010006 2A714400 |..........*..qD.....*qD. |
1 91 19:55:04.631206 00:00:00.016087 24 24 60 84
000000 60000000 00141101 | `.......| 000020 FEC9C2D4 00000000 00062900 0EDC217C FEC9C2D4 00000000 02062AFF FE714400 |..........}!|..........*..qD. |
000000 0802A829B 0014A3B6 01010000 3C697318 00095CAB |.*........<is..... |

After all the records have processed, the number of streams and the amount of storage required for the report and buffers are shown.

Each session is identified by the IP addresses, port number, and protocol. The time stamps of the first and last packet are shown along with the number of packets, the address, and size of the stream buffer.

When a stream is dumped, each packet and the data from the packet is shown. If there are gaps in the stream, the number of bytes skipped is displayed. The data about each packet formatted are:

D The direction of the packet: I for inbound and O for outbound.

Rcd #
The record number.

Time
The time stamp of the record.

Delta
The time from the first record of the stream.

Seq #
The sequence number of the TCP packet. For other packets it is the relative offset of the packet from the first packet.

Position
The relative offset of the packet.
Length
The number of bytes in the packet.

End_Pos
The ending sequence number.
Formatting packet trace using a batch job

A Packet Trace can also be formatted through the use of a batch job. The following is an example of JCL for a batch job:

```
//jobname DD (accounting),pgmname,CLASS=A,MSGCLASS=A
//DUMP EXEC PGM=IKJEFT01
//STEPLIB DD DISP=SHR,DSN=hlq.MIGLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//PRINTER DD SYSOUT=*
//SYSPROC DD DISP=SHR,DSN=SYS1.CLIST
// DD DISP=SHR,DSN=SYS1.SBLSCLI0
// IPCSPARM DD DISP=SHR,DSN=SYS1.PARMLIB
// DD DISP=SHR,DSN=CPAC.PARMLIB
// DD DISP=SHR,DSN=SYS1.IBM.PARMLIB
// IPCSPRINT DD SYSOUT=*
// IPCSTOC DD SYSOUT=*
// IPCSSDIR DD DISP=SHR,DSN=userid.IPCS.DMPDIR
// SYSTSIN DD *
IPCS NOPARM
SETDEF DA('ctrace.dataset')
CTRACE COMP(SYSTCPDA) OPTIONS((systcpda_options_string))
END
/*
```

Data trace (SYSTCPDA) for TCP/IP stacks

Use the DATTRACE command to trace socket data (transforms) into and out of the physical file structure (PFS). DATTRACE operates with the following APIs:

- REXX
- C-sockets
- IMS
- CICS
- Native z/OS UNIX
- Macro
- CALL Instruction

Refer to the [z/OS Communications Server: IP System Administrator’s Commands](https://www.ibm.com) for information about the format of the data trace command (VARY DATTRACE).

Starting data trace

You can start data trace for all job names using the VARY command:

```
V TCPIP,tcpprocname,DAT
```

Tips:

- To use any VARY command, the user must be authorized in RACF. This replaces the old OBEY list authorization.
- Each user’s RACF profile must have access for a resource of the form MVS.VARY.TCPIP.xxx, where xxx is the first eight characters of the command name. For data trace, this would be MVS.VARY.TCPIP.DATTRACE.
- Traces are placed in an internal buffer, which can then be written out using an external writer. The MVS TRACe command must also be issued for component SYSTCPDA to activate the data trace.
Displaying data traces

You can use the Netstat command to display data traces. Figure 21 shows a data trace for a single entry.

```
netstat -p TCPCS -f
...
Data Trace Setting:
  JobName: *  TrRecCnt: 00000006 Length: FULL
  IpAddr: *  SubNet: *
  PortNum: *
```

Figure 21. Data trace: Single entry

```
netstat -p TCPCS -f
...
Data Trace Setting:
  JobName: MEGA4  TrRecCnt: 00000000 Length: FULL
  IpAddr: 127.0.0.3  SubNet: *
  PortNum: *
  JobName: *  TrRecCnt: 00000000 Length: FULL
  IpAddr: 127.0.0.9  SubNet: *
  PortNum: *
```

Figure 22. Data trace with multiple entries

Formatting data traces using IPCS

Data trace records are written to the same CTRACE component as packet trace records (SYSTCPDA). Thus, all the IPCS formatting features for packet trace are also available for data trace. You can use the ENTRIDLIST parameter to isolate data trace records and packet trace records from each other. See "Formatting packet traces using IPCS" on page 97 for more information.

Intrusion Detection Services trace (SYSTCPIS)

When starting the TCP stack, the stack reads the CTIIDS00 parmlib member to determine the size to reserve for the SYSTCPIS Ctrace. You can override this default by starting TCP/IP with the PARM option and the keyword IDS=xx, where xx is the suffix of the CTIIDSxx PARMLIB member. In the following example, the trace searches for PARMLIB member CTIIDS3A3.

```
% tcpiproc,PARM='IDS=A3'
```

If the parmlib member is not found or the member contains data that is not valid, the following message is displayed.

```
EZZ4210I CTRACE DEFINE FAILED FOR CTIIDS00
```

If the EZZ4210I message indicates the parmlib member name CTIIDS00, the IDS CTrace space is set up using the default BUFSIZE of 32M.

The CTIIDS00 member is used to specify the IDS CTrace parameters. To eliminate this message, ensure that a CTIIDS00 member exists within Parmlib and that the options are correctly specified. A sample CTIIDS00 member is shipped with z/OS Communications Server.

Packets are traced based on IDS policy defined in LDAP. Refer to Intrusion Detection Services in the z/OS Communications Server: IP Configuration Guide for information on defining policy.
Restrictions
For IDS trace records the COMP keyword must be SYSTCPIS. Because there are no EXCEPTION records for IDS trace, the EXCEPTION keyword must not be specified.

CTRACE keywords on SYSTCPIS
The following describes those CTRACE keywords that affect SYSTCPIS processing.

ENTIDLIST
Use the ENTIDLIST keyword to select trace records with a specific ProbeId.

JOBLIST, JOBNAME
Use the JOBLIST and JOBNAME keywords to select trace records with a matching job name. Also, use the JOBNAME keyword in the OPTIONS list to select records.
ASIDLIST
Use the ASIDLIST to select trace records with a matching Asid.

GMT
The time stamps are converted to GMT time.

LOCAL
The time stamps are converted to LOCAL time.

SHORT
If the OPTIONS keyword does not specify any reports, format the trace records. Equivalent to the FORMAT option.

FULL
If the OPTIONS keyword does not specify any reports, format and dump the trace records. Equivalent to the FORMAT and DUMP options.

SUMMARY
If the OPTIONS keyword does not specify any reports, create a one line summary for each trace record. Equivalent to the SUMMARY option.

TALLY
If the OPTIONS keyword does not specify any reports, then count the trace records. Equivalent to the STATISTICS option.

START and STOP
These keywords limit the trace records seen by the packet trace formatter. The START keyword determines the time when records are seen by the packet trace report formatter. The STOP keyword determines the time when records are no longer seen by the packet trace report formatter.

Rule: CTRACE always uses the time the trace record was moved to the buffer for START and STOP times.

LIMIT
Determines the number of records that the packet trace formatter is allowed to process. See the RECORDS keyword value in OPTIONS.

USEREXIT
The CTRACE USEREXIT is not called because the packet trace formatter tells CTRACE to skip all the records. Therefore, the packet trace formatter calls the CTRACE USEREXIT before testing the records with the filtering criteria. If it returns a nonzero return code, the record is skipped. The USEREXIT can also be used in the OPTIONS keyword. It is called after the record has met all the filtering criteria in the OPTIONS keyword. The OPTIONS keyword provides a means of entering additional keywords for record selection and formatting.

SYSTCPI5 OPTIONS
The syntax for the OPTIONS component routine parameters is:

OPTIONS component:

Data Selection:

Device Type   IP Identifier   Name   Port Number
Protocol | Record Number | Record Identifiers | Record Type
Device Type:

DEVTYPE (devtype)

IP Identifier:

ADDR (IP Address)

CLASS HOST IPADDR (IP Address) IPID (ip_id_number)

LOOPBACK QOS (quality_of_service)

IP Address:

BROADCAST CLASSA CLASSB CLASSC CLASSD

dd.d.ddd.ddd

A

X'hbbbbbb'

Name:

LINKNAME (linkname) JOBLIST (jobname)

Port Number:

BOOTP DHCPOP (port_number port_number)
Protocol:
OPTIONS Keywords

The following are keywords used for the OPTIONS component routine parameters.

ASCII

Dump trace record data with ASCII translation only. The default is BOTH.

BASIC

For specific packet types dump each element of the packet. Applies to DNS, RIP, and SNMP packet data.

BOOTP[(port_number|67; port_number|68)]

Select BOOTP protocol packets. The port_number defines the BOOTP port numbers to select trace records for formatting. Equivalent to PORT(67 68).

BOTH

Dump trace record data with both ASCII and EBCDIC translations. This is the default.

BROADCAST

Select trace records with a broadcast IP address. Equivalent to IPADDR(255.255.255.255/255.255.255.255).

CID

A connection identifier. Up to 16 identifiers can be specified. The CID values can be entered in either decimal (such as CID(182)) or hexadecimal (such as CID(X’0006CE7E’)), but are displayed in hexadecimal. This is the same value that appears in the NETSTAT connections display.
CLASSA
Select trace records with a class A IP address. Equivalent to IPADDR(0.0.0.0/128.0.0.0).

CLASSB
Select trace records with a class B IP address. Equivalent to IPADDR(128.0.0.0/192.0.0.0).

CLASSC
Select trace records with a class C IP address. Equivalent to IPADDR(192.0.0.0/224.0.0.0).

CLASSD
Select trace records with a class D IP address. Equivalent to IPADDR(224.0.0.0/240.0.0.0).

CLASSE
Select trace records with a class E IP address. Equivalent to IPADDR(240.0.0.0/248.0.0.0).

CLEANUP(nnnnn | 500)
Defines a record interval where saved packet information in storage is released. The minimum value is 500 records. The maximum value is 1 048 576 records; the default is 500 records. If you set the record interval to 0, cleanup does not occur.

DATASIZE(data_size | 0)
Selects packets that contain more protocol data than the data_size value. The minimum value is 0. The maximum value is 65535. The data size is determined from amount of packet data available minus the size of any protocol headers. Equivalent to FLAGS(DATA).

CORRELATOR
Select trace records with one of the matching correlator identifiers. Up to 16 identifiers can be specified. Each identifier in the list can be a range: low_number:high_number. Values can be decimal (nnnnnnnnnnn) or hexadecimal (X‘hhhhhhhh’). This filter associates packets in the trace with an IDS event message in syslogd or the system console.

DEBUG(debug_level_list)
Provide documentation about SYSTCPIS format processing. debug_level_list is a list of numbers from 1 to 64. Use only under the direction of an IBM Service representative.

DELAYACK(threshold | 200)
The delay acknowledgement threshold in milliseconds used in the calculation of round-trip time in the TCP session report. The minimum value is 10 milliseconds; the maximum value is 1000 milliseconds; the default value is 200 milliseconds.

DEVTYPE(device_type_list)
Select packets written to or received from an interface with one of the specified device types. From 1 to 16 types can be specified. This does not apply to data trace records. The following types can be specified:
• ATM
• CDLC
• CLAW
• CTC
• ETHER8023
• ETHERNET
• ETHEROR8023
• FDDI
• HCH
• IBMTR
• IPAQENET
• IPAQENET6
• IPAQIDIO
• IPAQIDIO6
• IPAQTR
• LOOPBACK
• LOOPBACK6
• MPCPTP
• MPCPTP6
• OSAFDDI
• OSAENET
• SNALINK
• SNALU62
• VIRTUAL
• VIRTUAL6
• X25NPSI

DHCP[(port_number|67; port_number|68)]
Select DHCP protocol packets. The port_number defines the DHCP port numbers to select trace records for formatting. Equivalent to PORT(67 68).

DISCARD(reason_code_list)
Select packets with one of the specified discard reason codes. Up to 16 discard reason codes can be specified in the range 0 - 65535. Each entry in the list can be a range: low_number:high_number. Values can be decimal or hexadecimal.

0 Packet was not discarded
1:4087 A packet was discarded by OSA-Express
1:1023 Select packets discarded by OSA-Express for DISCARD=EXCEPTION reasons
4096:8191 IP packet was discarded by TCPIP
8192:12287 TCP packet was discarded by TCPIP

See [z/OS Communications Server: IP and SNA Codes](https://www.ibm.com/support/docview.ws/docview/105830) for the TCP/IP discard reason codes.

DNS[(port_number|53)]
Select Domain Namer Service protocol packets. The port_number defines the DNS port number to select trace records for formatting. Equivalent to PORT(53).

DOMAIN[(port_number|53)]
Select Domain Namer Service protocol packets. The port_number defines the DNS port number to select trace records for formatting. Equivalent to PORT(53).

DUMP(nnnnnnnn|65535)]
Dump the selected packets in hexadecimal with EBCDIC and ASCII translations. The IP and protocol headers are dumped separately from the packet data. The value nnnnnnnn is the maximum amount of packet data to be dumped from each packet. The default value is 65535 bytes; the minimum
value is 0; the maximum value is 65535. The IP and protocol headers are not subject to this maximum. The default report options are DUMP and FORMAT. The BOTH, ASCII, EBCDIC, and HEX keywords describe how the dumped packets are translated. The default is BOTH.

**EBCDIC**
Dump trace record data with EBCDIC translation only. The default is BOTH.

**EXPORT[DETAIL|SUMMARY]**
The selected packets are written to the EXPORT data set in .CSV (Comma Separated Value) format. In .CSV format, each character field is surrounded by double quotation marks and successive fields are separated by commas. The file's first line defines the fields. Each subsequent line is a record containing the values for each field.

**DETAIL**
Format the IP header, protocol header, and protocol data as separate lines of data.

**SUMMARY**
Format the IP header and protocol header in one line of data. SUMMARY is the default.

Allocate a file with DDNAME of EXPORT before invoking the CTRACE command with EXPORT in the OPTIONS string.

ALLOC FILE(EXPORT) DA(PACKET.CSV) SPACE(15 15) TRACK

The record format is variable block with logical record length of 512 bytes.

**FINGER[port_number|79]**
Select FINGER protocol packets. The port_number defines the FINGER port number to select trace records for formatting. Equivalent to PORT(79).

**FLAGS(flags list)**
Select packets that have the matching characteristics. Flags that can be specified are:

**ABBREV**
Select packets that are abbreviated.

**ACK**
Select packets that have a TCP header with the ACK flag set.

**DATA**
Selects packets that contain data.

**DF**
Select IPv4 packets that have the do not fragment (ip_df) flag set.

**FIC**
Select packets that are the first fragment of an IPv4 or IPv6 packet.

**FIN**
Select packets that have a TCP header with the FIN flag set.

**FULL**
Select packets that are complete.

**HOME**
Select packets that have an IP destination address equal to the IP source address.

**IN**
Select packets that are inbound.

**IPO**
Select packets that have an IPv4 header options field.

**IPV4**
Select IPv4 packets. IPv4 cannot be used in combination with other data selectors that are IPv6-specific, such as LINKLOCAL.

**IPV6**
Select IPv6 packets. IPv6 cannot be used in combination with other data selectors that are IPv4-specific, such as BROADCAST.
IPV6EXT
Select packets that have an IPv6 extension header.
LIC Select packets that are the last fragment of an IPv4 or IPv6 packet.
MIC Select packets that are the middle fragment of an IPv4 or IPv6 packet.
OIC Select IPv4 or IPv6 packets that are not fragmented.
OUT Select packets that are outbound.
PING Select packets that are ICMP/ICMPv6 echo request and echo reply.
PSH Select packets that have a TCP header with the PSH flag set.
RSM Select packets that have been reassembled.
RST Select packets that have a TCP header with the RST flag set.
SEG Select packets that have been segmented.
SYN Select packets that have a TCP header with the SYN flag set.
TCPO Select packets that have a TCP header options field.
TOS Select IPv4 packets that have a nonzero value in the ip_tos field.
TUNNEL
Select packets with protocol number 47 GRE or 41 (IPv6 over IPv4).
z/OS Communications Server currently does not support IPv6 over IPv4 (protocol number 41).
URG Select packets that have a TCP header with the URG flag set.
ZWIN Select packets that have a TCP header with a zero window value.

Notes:
1. The use of the FIC, MIC and LIC flags require the use of the NOreassemble option.
2. When a packet is reassembled, then it becomes an OIC packet with the RSM flag set.

FMT
Equivalent to FORMAT.

FORMAT[(DETAIL | SUMMARY)]
The selected packets with defined packet data are to be formatted. The SHORT keyword on the CTRACE command selects this option if no other report options are specified. The default report options are DUMP and FORMAT.

DETAIL
Format the IP header, protocol header, and the protocol data. This is the default.

SUMMARY
Format the IP header and protocol header. DETAIL is the default.

FTP[(port_number|20; port_number|21)]
Select FTP protocol packets. The port_number defines the FTP port numbers to select trace records for formatting. Equivalent to PORT(20,21).

FULL
Equivalent to DUMP and FORMAT. The FULL keyword on the CTRACE command selects this option if no other report options are specified.
GAIN(rtgain | 125, vargain | 250)
Used in the calculation of round-trip time in the TCP session report. The time is expressed in milliseconds. The minimum value is 0 milliseconds; the maximum value is 1000 milliseconds.

rtgain  The round trip gain value. The default value is 125 milliseconds.

vargain  The variance gain value. The default value is 250 milliseconds.

GOPHER((port_number | 70)]
Select GOPHER protocol packets. The port_number defines the GOPHER port numbers to select trace records for formatting. Equivalent to PORT(70).

GRE
Select trace records with a protocol number of 47. Equivalent to PROTOCOL(47).

GROUP
Select trace records with one of the matching group identifiers. The following group identifiers can be specified:
  • TCPTP
  • UDPTR
  • SCAN
  • ATTACK

HEX
Trace record data is not dumped with ASCII or EBCDIC translation. The default is BOTH.

HOST
Select trace records with a host IP address. Equivalent to IPADDR(0.0.0.0/255.255.0.0).

HTTP((port_number | 80)]
Select HTTP protocol packets. The port_number defines the HTTP port numbers to select trace records for formatting. Equivalent to PORT(80). See www on 120.

ICMP
Select trace records with a protocol number of 1. Equivalent to PROTOCOL(1).

IGMP
Select trace records with a protocol number of 2. Equivalent to PROTOCOL(2).

INSTANCE
Select trace records with one of the matching instance identifiers. The identifiers can be in decimal (nnnnn) or hexadecimal (x'hhhhhhhh'). The instance identifier is the lower 2 bytes of the PROBEID. Up to 16 identifiers can be specified.

IPADDR(ipaddr[/subnet_mask] | X'hhhhhhhh'[\-nnnnn.])
Select packets with a matching IP address, optional subnet mask and optional port number. Up to 16 IP addresses can be specified. The IPADDR is specified in three parts:
  1. An IP address
     The address can be in dotted decimal notation, a keyword, or a hex value.
     • Dotted decimal notation
       127.0.0.1
     • A keyword
A class A address, 0.0.0.0/128.0.0.0
B class B address, 128.0.0.0/192.0.0.0
C class C address, 192.0.0.0/224.0.0.0
D class D address, 224.0.0.0/240.0.0.0
E class E address, 240.0.0.0/248.0.0.0
H local host address, 0.0.0.0/0.255.255
L loopback address, 127.0.0.0/255.0.0.0
M broadcast address, 255.255.255.255/255.255.255.255
* Any address, 0.0.0.0/0.0.0.0
0 An address of zero, 0.0.0.0/255.255.255.255

Hexadecimal notation
X'7f000001'

2. A submask
   The submask is preceded by a slash (/). Specify a submask only when the
   IP address is in dotted decimal notation. The mask can be in dotted
decimal notation or as a shift value. The subnet shift value is a number less
than or equal to 32. Example: 9\8 or 9.37\16

3. A port number
   The port number is preceded by a dash (—). It is a decimal number in the
range 0–65535.

Notes:
1. There should be no spaces between the IP addresses and the subnet masks.
2. The BROADCAST, CLASSA, CLASSB, CLASSC, CLASSD, CLASSE, HOST,
   and LOOPBACK keywords add to the total of 16 IP addresses.
3. The port number, when used, adds to the total of 16 port numbers in the
   PORT keyword.

IKE
   Select ISAKMP protocol packets. Equivalent to PORT(500 4500).

IPID(ipid_number_list)
   Select packets that match the ip_id number in the IP packet header. Up to 16
   ID numbers can be specified in the range 0–65535. Each entry in the list can be
   a range: low_number:high_number. Values can be decimal (nnnnn) or
   hexadecimal (x'hhhh'). If the packets have been fragmented, specify
   NOREASSEMBLY to select each packet.

JOBLIST|JOBNAME(job_name_list)
   Select data trace records with the specified JOBNAME. Up to 16 job names can
be specified. Each job name can be up to 8 characters in length. If the last
character of a job name is an asterisk (*) then only the characters up to the
asterisk are compared.

   The CTRACE JOBLIST/JOBNAME parameter provides the same function,
except that wildcards are not supported.

LINKNAME(link_name_list)
   Select packet trace records with the specified LINKNAME. Up to 16 link names
   can be specified. Each link name can be up to 16 characters in length. If the
   last character of a link name is an asterisk (*) then only the characters up to the
   asterisk are compared.
LOOPBACK
Select trace records with a loop back address. Equivalent to
IPADDR(127.0.0.0/255.0.0.0).

NOREASSEMBLY
Do not reassemble fragmented IP packets into a complete packet. This is the
default.

NOSEGMENT
Packet trace records that span multiple Ctrace records are not recombed.
Only the first segment record of a packet is used. The rest of the segment
records are discarded. SEGMENT is the default.

NOT
If the NOT option is selected then any selection criteria is reversed. If a record
matches the selection criteria, it is not processed. If a record does not match the
selection criteria, it is processed. If no selection criteria were found in the
OPTIONS(( )) keyword then the NOT option has no effect.

NTP(port_number|123)]
Select NTP protocol packets. The port number defines the NTP port number to
select packets for formatting. Equivalent to PORT(123).

OPTION
The selected options with defaults are listed.

OSPF1
Select packets with a protocol number of 89. Equivalent to PROTOCOL(89).

PORT(port_number_list)
Select trace records with one of the specified port numbers. Up to 16 port
numbers can be specified in the range 0–65535. The following keywords add to
the total number of ports:
• BOOTP
• DHCP
• DNS
• DOMAIN
• FINGER
• GOPHER
• HTTP
• RIP
• ROUTER
• RPC
• SMTP
• SNMP
• TELNET
• TFTP
• TIME
• WWW

PROBEID
Select trace records with one of the matching probe identifiers. The identifiers
can be expressed in decimal (nnnnn) or hexadecimal (x'hhhhhhhhh'). Up to 16
identifiers can be specified. You can also specify the probe identifiers on the
ENTIDLIST keyword of the CTRACE subcommand. Refer to the **z/OS Communications Server: IP and SNA Codes** for additional information about probe identifiers.

**PROTOCOL(protocol number list)**
Select trace records with one of the specified protocol numbers. Up to 16 protocol numbers can be specified in the range 0–255. Each entry in the list can be a range: low_number:high_number. Values can be decimal (nnn) or hexadecimal (X’hh’). The following keywords add to the total number of protocols:
- ICMP
- IGMP
- OSPFI
- SESSION
- TRANSIT
- TCP
- UDP

**QOS(quality_of_service_list)**
Select the records with the matching quality of service from the ip_tos field. Up to 16 QOS values can be specified in the range 0–7. Each entry in the list can be a range: low_number:high_number. Values can be decimal (n) or hexadecimal (X’h’).

**REASSEMBLY[(packet_size|65535)]**
Reassemble IP fragments into a complete packet. packet_size is the maximum size of a reassembled packet that is allowed. The smallest value allowed is 576 bytes, the largest is 65535 bytes. The default value is 65535 bytes. NOREASSEMBLY is the default.

**RECORDS(record_number_list)**
Select the records with the matching record numbers in the trace data. Up to sixteen (16) record numbers can be specified. Record numbers are assigned after any IPCS CTRACE selection criteria have been met. Each entry in the list can be a range: low_number:high_number. Values can be decimal (nnnnnnnnnn) or hexadecimal (X’hhhhhhhh’).

**RIP[(port_number|520)]**
Select RIP protocol packets. The port_number defines the RIP port number to select trace records for formatting. Equivalent to PORT(520).

**ROUTER[(port_number|520)]**
Select RIP protocol packets. The port_number defines the RIP port number to select trace records for formatting. Equivalent to PORT(520).

**RPC[(port_number|111)]**
Select RPC protocol packets. The port_number defines the RPC port number to select trace records for formatting. Equivalent to PORT(111).

**SEGMENT**
Packet trace records that span multiple Ctrace records are recombined. Data from segment records is saved until all the Ctrace records have been read to re-create the original packet. SEGMENT is the default.

**SESSION[(DETAIL|STATE|SUMMARY)]**
Generate a report showing TCP or UDP session traffic.
DETAIL
List each of the packets for a session, and the summary statistics.
DETAIL is the default.

STATE
List the beginning and ending state of each session.

SUMMARY
Show only the summary statistics.

Guideline: The UDP session analysis is also used for other protocols.

SMTP[(port_number|25)]
Select SMTP protocol packets. The port_number defines the SMTP port number to select trace records for formatting. Equivalent to PORT(25).

SNIFFER[nnnn 200, ETHERNET|TCPDUMP]
Writes the trace records in a format acceptable for downloading to other trace analysis programs, such as programs from http://www.wireshark.org/.

nnnn
The maximum size of trace data. Packets with more data than this value are truncated. The default is 200 bytes. The largest value is derived from the LRECL of the SNIFFER data set.

ETHERNET
If this keyword is specified, the output is formatted for the Ethernet analysis application of the analyzer. This keyword specifies the file format only and does not imply that only packets traced on an Ethernet are collected. Packets from all devices can be collected using this option.

The default for the SNIFFER option is ETHERNET.

TCPDUMP
The format is compatible with the www.tcpdump.org files with an Ethernet header.

Note: The TOKENRING keyword used on the IPCS Ctrace subcommand is ignored.

CTRACE OPTIONS(( SNIFFER(TOKENRING) ))

The ETHERNET format of the sniffer data set will be selected.

The trace records are written to the file with a DD name of SNIFFER. After the file is generated, it can be downloaded as a binary file to the analyzer and loaded using the standard features of the analyzer. Use NOREASSEMBLY to prevent the formatter from reassembling packets. Then, each packet is passed as the packets as they are collected. The logical record length of the SNIFFER data set determines the largest amount of packet data written to the data set.

Allocate a file with DDNAME of SNIFFER before invoking the CTRACE command with SNIFFER in the OPTIONS string as follows:

ALLOC FILE(SNIFFER) DA(PACKET.TRC) SPACE(15 15) TRACK +
LRECL(8000) BLKSIZE(32000)

The data set has a record format of variable blocked with a logical record length of 8000 bytes. The maximum IP packet size is 7962 (8000 - 38) for SNIFFER(ETHERNET).

The minimum logical record length of the data set is 256 bytes.
SNMP[(port_number|161 port_number|162)]
Select SNMP protocol packets. The port_number defines the SNMP port number to select trace records for formatting. Equivalent to PORT(161 162).

SPEED(local | 10,remote | 10)
The link speed, in megabits per second, for the local and remote link. These values are used in throughput calculations in the TCP session report. Valid values are in the range 0-17171. The default value is 10. Specify the slowest speed of the link in the route.

STATISTICS[(DETAIL | SUMMARY)]
After all the records have been processed, generate a report showing the number of records selected by record type, Device type, Jobname, Linkname, Protocol number, IP address, and port numbers. TALLY on the CTRACE command selects this option if no other report options are specified.

STATS
Equivalent to the STATISTICS option.

STREAMS[(nnn | 128)]
Collect the packet data for dumping or formatting after the trace file has been processed. nnn is the maximum amount of storage used to capture each stream. The smallest value is 16KB. The largest value is 512KB. The default value is 128KB. The value is in 1024 bytes (1K) units.

SUMMARY
Format a single line for each trace record. SUMMARY on the CTRACE command selects this option if no other report options are specified.

TALLY
Equivalent to the STATISTICS option.

TCP
Select trace records with a protocol number of 6. Equivalent to PROTOCOL(6).

TELNET[(port_number|23 [screen_width|80] [SUMMARY|DETAIL] ) ]
Select TELNET protocol packets. The port_number defines the TELNET port number to select packets for formatting. Equivalent to PORT(23).

The screen_width parameter defines the value used for converting buffer offsets into row and column values for the 3270 data stream formatting. If the screen_width parameter is provided, then the port_number parameter must also be used. The minimum value is 80. The maximum value is 255. The default value is 80.

SUMMARY formats the 3270 data stream into a representation of the screen.
DETAIL formats each 3270 command and order.

There is no default for DETAIL or SUMMARY.

TFTP[(port_number|69)]
Select TFTP protocol packets. The port_number defines the TFTP port number to select trace records for formatting. Equivalent to PORT(69).

TIME[(port_number|37)]
Select TIME protocol packets. The port_number defines the TIME port number to select trace records for formatting. Equivalent to PORT(37).

TOD
Use the time the trace data was captured for the reports. Normally the time the
trace data was moved to the trace buffer is shown. The CTRACE command uses the time stamp when the trace data was moved to the buffers for START and STOP time selection.

**TYPE(probe type identifier)**
Select trace records with one of the matching probe type identifiers. The **probe type identifier** is the second byte of the probe identifier. Up to 16 identifiers can be specified. You can use the following probe types:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>TCPTR</td>
</tr>
<tr>
<td>0200</td>
<td>UD PTR</td>
</tr>
<tr>
<td>0301</td>
<td>VSSCAN</td>
</tr>
<tr>
<td>0303</td>
<td>NORMSCAN</td>
</tr>
<tr>
<td>0401</td>
<td>MALFORMED</td>
</tr>
<tr>
<td>0402</td>
<td>RAW</td>
</tr>
<tr>
<td>0403</td>
<td>IPFRAGMENT</td>
</tr>
<tr>
<td>0404</td>
<td>ICMP</td>
</tr>
<tr>
<td>0405</td>
<td>IPOPT</td>
</tr>
<tr>
<td>0406</td>
<td>IPPROTO</td>
</tr>
<tr>
<td>0407</td>
<td>FLOOD</td>
</tr>
<tr>
<td>0408</td>
<td>PREPECHO</td>
</tr>
</tbody>
</table>

**UDP**
Select trace records with a protocol number of 17. Equivalent to PROTOCOL(17).

**USEREXIT(exitname)**
Names the user exit to be called for each selected record. The USEREXIT is called after the record passes the other filter options. It is passed the same parameter list as the CTRACE user exit. A nonzero return code indicates the record is not selected for formatting. The USEREXIT keyword on the CTRACE command names a user exit that is called before the SYSTCPI5 trace record filtering is done. If this exit routine returns a nonzero return code, then the record is skipped by the SYSTCPI5 formatter.

**WWW(port_number|80)**
Select HTTP protocol packets. The **port_number** defines the HTTP port number to select trace records for formatting. Equivalent to PORT(80).
IDS reports

The SYSTCPIS Ctrace formatter is based on the SYSTCPDA formatter (and in fact shares many of the data structures and format routines) and includes the reports for the SYSTCPDA formatter. However, the REASSEMBLY, SESSION, and STREAMS reports might prove of little value for the SYSTCPIS, because they depend on having a more complete set of packets.

- The STATISTICS report (both SUMMARY and DETAIL) provide an overview of the data collected.
- The SUMMARY report provides one line per IDS event.
- The FORMAT, and DUMP reports format individual packets.
- The EXPORT and SNIFFER options write the packet to an external file for later analysis.

The following sections describe the various reports available.
OPTION

Purpose: List the selected options and default keyword values.

Format: The following command was used to obtain the example of this report.

CTRACE COMP(SYSTCPIS) SUB((TCPCS)) DSN('IBMUSER.CTRACE1')
OPTION((OPT SESS FORM))
REPORT

Examples:

COMPONENT TRACE SHORT FORMAT
SYSNAME(MVS118)
COMP(SYSTCPIS)SUBNAME((TCPCS))
OPTIONS((OPT SESS FORM))
1 DSNAME('IBMUSER.CTRACE1')
2 OPTIONS((Both Bootp(67,68) Cleanup(500) DelayAck(200,200) Domain(53)
Finger(79) Flags() Format(Detail) Ftp(20,21) Gain(125,250) Gopher(70)
Limit(999999999) Ntp(123) Option Noreassembly Router(520) Rpc(111) Segment
Session(Detail) Smtp(25) Snmp(161,162) Speed(10,10) Telnet(23) Tftp(69)
Time(37) Userexit() Www(80))
)

The following describes numbered areas of the example.

1 DSNAME is the name of the source data.

2 OPTIONS((...)) is a listing of the active options with default values.
SUMMARY

Purpose: Show one line of information about each record in the trace.

Format: The following command was used to obtain the example of this report.
CTRACE COMP(SYSCTPIS) SUB((TCPCS)) SUMMARY DSN('IBMUSER.CTRACE1')

Examples:

COMPONENT TRACE SUMMARY FORMAT
SYSNAME(MVS118)
COMP(SYSCTPIS)SUBNAME((TCPCS))
DSNAME('IBMUSER.CTRACE1')

**** 2002/11/20
I - Inbound packet
O - Outbound packet

<table>
<thead>
<tr>
<th>DP</th>
<th>Nr</th>
<th>hh:mm:ss.mmmm</th>
<th>IpId Group</th>
<th>Probe Id</th>
<th>Corelatr JobName</th>
<th>Cid</th>
<th>DatLn</th>
<th>Data</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>II</td>
<td>4521</td>
<td>17:38:32.175560</td>
<td>0000 SCAN</td>
<td>03030000</td>
<td>10 TCP</td>
<td>00000000</td>
<td>12 ICMP</td>
<td>9.42.105.71</td>
<td>9.42.104.38</td>
</tr>
<tr>
<td>IT</td>
<td>4522</td>
<td>17:38:45.130339</td>
<td>163F SCAN</td>
<td>03030026</td>
<td>11 FTPD1</td>
<td>00000002</td>
<td>0 TCP</td>
<td>9.2.197.34-4691</td>
<td>9.42.104.38-21</td>
</tr>
<tr>
<td>IT</td>
<td>4523</td>
<td>17:38:45.153474</td>
<td>173F SCAN</td>
<td>03030026</td>
<td>12 FTPD1</td>
<td>00000002</td>
<td>0 TCP</td>
<td>9.224.157.220-47167</td>
<td>9.42.104.38-21</td>
</tr>
<tr>
<td>IT</td>
<td>4524</td>
<td>17:38:45.170441</td>
<td>183F SCAN</td>
<td>03030026</td>
<td>13 FTPD1</td>
<td>00000002</td>
<td>0 TCP</td>
<td>9.74.208.131-47423</td>
<td>9.42.104.38-21</td>
</tr>
<tr>
<td>IT</td>
<td>4525</td>
<td>17:38:45.190606</td>
<td>193F SCAN</td>
<td>03030026</td>
<td>14 FTPD1</td>
<td>00000002</td>
<td>0 TCP</td>
<td>9.79.235.253-47679</td>
<td>9.42.104.38-21</td>
</tr>
<tr>
<td>IT</td>
<td>4526</td>
<td>17:38:45.213117</td>
<td>1A3F SCAN</td>
<td>03030026</td>
<td>15 FTPD1</td>
<td>00000002</td>
<td>0 TCP</td>
<td>9.40.107.43-47935</td>
<td>9.42.104.38-21</td>
</tr>
<tr>
<td>IT</td>
<td>5671</td>
<td>17:59:32.787165</td>
<td>0B3B ATTACK</td>
<td>04070002</td>
<td>277 FTPD1</td>
<td>00000002</td>
<td>0 TCP</td>
<td>9.42.104.38-21</td>
<td>9.84.160.95-47938</td>
</tr>
<tr>
<td>IT</td>
<td>5672</td>
<td>17:59:32.806700</td>
<td>0B1A ATTACK</td>
<td>04070002</td>
<td>277 FTPD1</td>
<td>00000002</td>
<td>0 TCP</td>
<td>9.42.104.38-21</td>
<td>9.156.214.250-44610</td>
</tr>
<tr>
<td>IT</td>
<td>5673</td>
<td>17:59:32.827193</td>
<td>0B1B ATTACK</td>
<td>04070002</td>
<td>277 FTPD1</td>
<td>00000002</td>
<td>0 TCP</td>
<td>9.42.104.38-21</td>
<td>9.150.148.96-44866</td>
</tr>
<tr>
<td>IT</td>
<td>5674</td>
<td>17:59:32.847730</td>
<td>0B1C ATTACK</td>
<td>04070002</td>
<td>277 FTPD1</td>
<td>00000002</td>
<td>0 TCP</td>
<td>9.42.104.38-21</td>
<td>9.48.42.177-45122</td>
</tr>
</tbody>
</table>

================================================================================

SYSTCPI Trace Statistics
2,583 ctrace records processed
0 segmented trace records read
0 segmented trace records were lost
2,583 trace records read
0 records could not be validated
2,583 records passed filtering
2,583 packet trace records processed
0 data trace records processed

The following describe areas of the example.

D  Direction of the packet:
   I  Inbound packet
   O  Outbound packet

P  The packet protocol:
T  TCP
U  UDP
I  ICMP
G  IGMP
P  Other

Nr  The Ctrace record number.

hh:mm:ss.mmmmmmm
The time stamp of the record.

IpId
The packet ID number in hexadecimal.

Group
The group assigned to the trace record. The value can be ATTACK, SCAN,
UDPTR or TC PTR.

Probe Id
The probe identifier assigned to the trace record.

Corelatr
The correlator assigned to the trace record. Use this to correlate the trace data
with console or syslog messages.

JobName
The job name assigned to the trace record.

Cid
The connection identifier assigned to the trace record.

DatLn
The length of the data.

Data
The protocol in the IP header.

Source/Destination
The source and destination IP address and port number.
FORMAT

Purpose: Format the Ctrace record header, the IP packet header, the protocol header, and the packet data. If one of the ports is a well-known port number and the SYSTCPIS supports data for the port number, the packet data is shown.

Format: The following command was used to obtain the example of this report.
CTRACE COMP(SYSTCPIS) SUB((TCPCS)) SHORT DSN('IBMUSER.CTRACE1') OPTIONS((OPT FORMAT))

Examples:

COMPONENT TRACE SHORT FORMAT
SYSNAME(MVS118)
COMP(SYSTCPIS)SUBNAME((TCPCS))
OPTIONS((OPT FORMAT))
DSNAME('IBMUSER.CTRACE1')

OPTIONS((Both Bootp(67,68) Cleanup(500) DelayAck(200,200) Domain(53)
Finger(79) Flags() Format(Detail) Ftp(20,21) Gain(125,250) Gopher(70)
Limit(999999999) Gmt Ntp(123) Option Noreassembly Router(520) Rpc(111)
Segment Smtp(25) Snmp(161,162) Speed(10,10) Telnet(23) Tftp(69) Time(37)
Userexit() Www(80)))

RcdNr Sysname Mnemonic Entry Id Time Stamp Description
----- -------- -------- -------- --------------- --------------------------------
-------------------------------------------------------------------------------
/ SF5800001/SF590000
4521 MVS118 SCAN 03030000 17:38:32.175560 Scan-Normal packet
From Link : ETH1 Device: LCS Ethernet Full=40
Tod Clock : 2002/11/20 17:38:32.175559 Module: EZBIPCIM
Job Name : TCPCS Asid: 01F7 Tcb: 00000000
Cid : 00000000 Correlator: 10
Policy : ScanEventIcmp-rule
IpHeader: Version : 4 Header Length: 20
Tos : 00 QOS: Routine Normal Service
Packet Length : 40 ID Number: 0000
Fragment : DontFragment Offset: 0
TTL : 62 Protocol: ICMP CheckSum: 5914 FFFF
Source : 9.42.105.71 Destination : 9.42.104.38
ICMP
Type/Code : ECHO CheckSum: 5592 FFFF
Id : 0B3F Seq: 0
Echo Data : 12
000000 AEBDB3D 03340A00 00000000 |...=4......|
-------------------------------------------------------------------------------
/ SF5800002/SF590000
4522 MVS118 SCAN 03030026 17:38:45.130339 Scan Normal-TCP SYN dropped
From Link : UNKNOWN Device: Unknown:0 Full=40
Tod Clock : 2002/11/20 17:38:45.130338 Module: EZBTCPCN
Job Name : FTPD1 Asid: 01F7 Tcb: 00000000
Cid : 00000020 Correlator: 11
Policy : ScanEventHigh-rule
IpHeader: Version : 4 Header Length: 20
Tos : 00 QOS: Routine Normal Service
Packet Length : 40 ID Number: 163F
Fragment : Offset: 0
TTL : 253 Protocol: TCP CheckSum: 681C FFFF
Source : 9.2.197.34 Destination : 9.42.104.38

TCP
The following describes numbered areas of the example.

1. The date of the trace records.

2. A summary line indicating the source of the trace record showing:
   - The record number.
   - The system name.
   - The group name.
   - The probe ID value (in hexadecimal).
   - The time the record was moved to the trace buffer, or with the TOD option
     the time the trace data was captured.
   - The description of the IDS event associated with the probe.

3. The trace header with these fields:
   - The direction of the trace record: From or To.
   - The link name.
   - The device type.
   - Full or Abbrev with amount of trace data available.
   - The time the trace record was captured.
   - The module that triggered the probe.
   - The job name associated when the probe was triggered.
   - The ASID of the address space when the probe was triggered.
   - The system tcb pointer when the probe was triggered (or zero if in SRB
     mode).
   - The CID (communications ID) of the session.
   - The Event identifier, the upper 2 bytes of the PROBEID.
   - The Correlator identifier.
   - The name of the current policy. This might be the policy that triggered the
     probe or the name of the policy the session was using at the time the probe
     was triggered.

4. The IP header showing fields from the IPv4 4 header. The header length is the
   number of bytes for the header. The offset field is the number of bytes from
the end of the IP header where the fragment appears. With the REASSEMBLY option active, this field always displays zeros.

5 The protocol header. In this example, it is an ICMP header.

6 Depending on the port number, the trace data might be formatted.

Guideline: If possible, the check sum of the packet is calculated. If the calculated value is X'FFFF', then the check sum is correct. If the calculated value is X'0000', then the check sum could not be calculated. The packet was incomplete or fragmented. Other values indicate a check sum error.

Using the protocol numbers and the well known port numbers, format routines are invoked to format standard packet data records. The port number for the PORT keywords define the port numbers to be used to invoke a format routine.

<table>
<thead>
<tr>
<th>Port</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>67, 68</td>
<td>BOOTP</td>
</tr>
<tr>
<td>67, 68</td>
<td>DHCP</td>
</tr>
<tr>
<td>53</td>
<td>Domain</td>
</tr>
<tr>
<td>79</td>
<td>Finger</td>
</tr>
<tr>
<td>70</td>
<td>Gopher</td>
</tr>
<tr>
<td>520</td>
<td>Rip</td>
</tr>
<tr>
<td>520</td>
<td>Router</td>
</tr>
<tr>
<td>111</td>
<td>RFC</td>
</tr>
<tr>
<td>25</td>
<td>SMTP</td>
</tr>
<tr>
<td>23</td>
<td>TELNET</td>
</tr>
<tr>
<td>69</td>
<td>TFTP</td>
</tr>
<tr>
<td>37</td>
<td>TIME</td>
</tr>
</tbody>
</table>
**DUMP**

**Purpose:** Format the IP header, protocol header and packet data in hexadecimal. The data can also be translated into EBCDIC, ASCII or both.

**Format:** The following command was used to obtain the example of this report.

```
CTRACE COMP(SYSTCPIS) SUB((TCPCS)) DSNAME('IBMUSER.CTRACE1') SHORT OPTIONS((OPT DUMP))
```

**Examples:**
```
COMPONENT TRACE SHORT FORMAT
SYSNAME(MVS118)
COMP(SYSTCPIS)SUBNAME((TCPCS))
OPTIONS((OPT DUMP))
DSNAME('IBMUSER.CTRACE1')

OPTIONS((Both Bootp(67,68) Cleanup(500) DelayAck(200,200) Domain(53)
          Dump(65535) Finger(79) Flags() Ftp(20,21) Gain(125,250) Gopher(70)
          Limit(999999999) Gmt Ntp(123) Option Noreassembly Router(520) Rpc(111)
          Segment Smtp(25) Snmp(161,162) Speed(10,10) Telnet(23) Tftp(69) Time(37)
          Userexit() Www(80)))

**** 2002/11/20
```

**RcdNr** **Sysname** **Mnemonic** **Entry Id** **Time Stamp** **Description**
--- **---------** **---------** **--------** **------------** **-------------------------------**

<table>
<thead>
<tr>
<th>RcdNr</th>
<th>Sysname</th>
<th>Mnemonic</th>
<th>Entry Id</th>
<th>Time Stamp</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4521</td>
<td>MVS118</td>
<td>SCAN</td>
<td>03030000</td>
<td>17:38:32.175560</td>
<td>Scan-Normal packet</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>From Link : ETH1 Device: LCS Ethernet Full=40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tod Clock : 2002/11/20 17:38:32.175559 Module: EZBIPICM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Job Name : TCPCS Asid: 01F7 Tcb: 00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cid : 00000000 Correlator: 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Policy : ScanEventIcmp-rule</td>
</tr>
<tr>
<td>4522</td>
<td>MVS118</td>
<td>SCAN</td>
<td>03030026</td>
<td>17:38:45.130339</td>
<td>Scan Normal-TCP SYN dropped</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>From Link : UNKNOWN Device: Unknown:0 Full=40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tod Clock : 2002/11/20 17:38:45.130338 Module: EZBTCPCN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Job Name : FTPD1 Asid: 01F7 Tcb: 00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cid : 00000020 Correlator: 11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Policy : ScanEventHigh-rule</td>
</tr>
</tbody>
</table>

---

```
1 IP Header : 20
000000 45000028 00004000 3E015914 092A6947 092A6826
```

```
2 Protocol Header : 8
000000 0B005592 0B3F0000
```

```
3 Data : 12  Data Length: 12
000000 AEBCDB3D 03340A00 00000000 .......... ...=.4...... |
```

---

```
4523 MVS118 SCAN 03030026 17:38:45.153474 Scan Normal-TCP SYN dropped
```

```
IP Header : 20
000000 45000028 163F0000 FD06681C 0902C522 092A6826
```

```
Protocol Header : 20
000000 B73F0015 BEEC917D 00000000 500200F2 4E530000
```

---
Protocol Header : 20
000000 B83F0015 76399A57 00000000 500200F2 5D2C0000
.
.
.

===============================================================================
SYSTCPIS Trace Statistics
2,623 ctrace records processed
 0 segmented trace records read
 0 segmented trace records were lost
2,623 trace records read
 0 records could not be validated
2,623 records passed filtering
2,623 packet trace records processed
 0 data trace records processed

The following describes numbered areas of the example.

1
   The IP header is dumped with no translation.

2
   The protocol header is dumped with no translation.

3
   The packet data is dumped with the translation specified by the ASCII, BOTH, EBCDIC, or HEX keyword. The default is BOTH. The amount of data dumped can be limited by the value specified with the DUMP keyword. The default is 65535 bytes.
SNIFTER

Purpose:  This report shows information written to the SNIFTER data set.

Format:  The following command was used to obtain the example of this report.

ALLOC F(SNIFFER) DATASET(SNIFFER.TRC) LRECL(1600) RECFM(V B) +
REUSE TRACK SPACE(15 15)

CTRACE COMP(SYSTCPIS) DSN('MWS.PQ33208.PTRACE4')+
OPTION((OPT TALLY SNIFFER NOREASSEMBLY))

Examples:

COMPONENT TRACE SHORT FORMAT
SYSNAME(MVS142)
COMP(SYSTCPIS)
OPTIONS(( OPT TALLY SNIFFER(4000) NOREASSEMBLY))
DSNAME('MWS.PQ33208.PTRACE4')
PTRPT04I SNIFFER(ETHERNET) option selected

OPTIONS((Both Bootp(67,68) Checksum(Summary) Cleanup(500) Datasize(0)
DelayAck(200,200) Domain(53) EE(12000:12004) Finger(79) First Flags(All )
Ftp(20,21) Gain(125,250) Gopher(70) Ike(500) Limit(999999999) Gmt
Nat(4500) Ntp(123) Option Noreassembly Router(520) Rpc(111) Sasp(3860)
Segment Smtp(25) Sniffer(4000, Ethernet) Snmp(161,162) Speed(10,10)
Statistics(Detail) Telnet(23,80,) Tftp(69) Time(37) Tod Userexit() Www(80)
))

Sniffer Report
3,385 records written to MWS.SNIFFER.ETH
639,789 bytes written
2121 packets were abbreviated
2024 is the maximum data size
3375 packets were truncated from 1843 bytes

Following are descriptions for some areas of the example.

108 records written to SNIFTER
This record count includes the two header records and one trailer record that
were written to the SNIFTER data set.

46 000 bytes written to SNIFTER
The number of data bytes written to the SNIFTER data set. This should be
close to the amount of data to be downloaded.

22 records were truncated to 1600 bytes
Because the logical record length was 1,600 bytes, 22 records were truncated.
This can be avoided by increasing the logical record length. The maximum
logical record length is 32,763 or the size of physical disk blocks, whichever is
smaller.

3.385 records written to MWS.SNIFFER.ETH
This record count includes the two header records and one trailer record that
were written to the SNIFFER data set.

639,789 bytes written
The number of data bytes written to the SNIFFER data set. This should be the
amount of data to be downloaded.

2121 packets were abbreviated
The number of packets that were abbreviated when the trace data was
collected.

2024 is the maximum data size
The size of the largest record written to the SNIFFER data set.
3375 packets were truncated from 1843 bytes
Because the logical record length was 2,048 bytes, 3375 records were truncated.
This can be avoided by increasing the logical record length. The maximum logical record length is 32,763 or the size of physical disk blocks, whichever is smaller.
STATISTICS

**Purpose:** The records are counted by probe ID, device type, interface, interface address, job name, Asid, QOS, TCP port number, UDP port number, connection identifier, group identifier, type identifier, correlator, protocol summary, and session summary.

**Format:** The following command was used to obtain the example of this report.

```
CTRACE COMP(SYSTCPIS) SUB((TCPCS)) SHORT OPTIONS((OPT STATISTICS(DETAIL)))
```

**Examples:**

```
COMPONENT TRACE SHORT FORMAT
SYSTYPE(MVS118)
COMP(SYSTCPIS)SUBNAME((TCPCS))
OPTIONS((OPT STATISTICS(DETAIL)))
DSNAME('IBMUSER.CTRACE1')
OPTIONS((Both Bootp(67,68) Cleanup(500) DelayAck(200,200) Domain(53)
  Finger(79) Ftp(20,21) Gain(125,250) Gopher(70) Limit(999999999)
  Get Ntp(123) Option Noassembly Router(520) Roc(111) Segment Smtp(25)
  Snmp(161,162) Speed(10,10) Statistics(Detail) Telnet(23) Tftp(69) Time(37)
  Userexit() Www(80)
  )
**** 2002/11/20
```

---

### SYSTCPIS Trace Statistics

2,623 ctrace records processed
0 segmented trace records read
0 segmented trace records were lost
2,623 trace records read
0 records could not be validated
2,623 packet trace records processed
0 data trace records processed

---

### Probe Report

<table>
<thead>
<tr>
<th>Total</th>
<th>Input</th>
<th>Data Output</th>
<th>Data First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
<th>Probe</th>
</tr>
</thead>
<tbody>
<tr>
<td>1576</td>
<td>1526</td>
<td>67144</td>
<td>0</td>
<td>0</td>
<td>4893</td>
</tr>
<tr>
<td>655</td>
<td>655</td>
<td>5652</td>
<td>0</td>
<td>0</td>
<td>5652</td>
</tr>
<tr>
<td>859</td>
<td>859</td>
<td>34360</td>
<td>0</td>
<td>0</td>
<td>5653</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>724</td>
<td>0</td>
<td>0</td>
<td>5654</td>
</tr>
</tbody>
</table>

9 Probe(s) found

---

### Device Type Report

<table>
<thead>
<tr>
<th>Total</th>
<th>Input</th>
<th>Data Output</th>
<th>Data First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
<th>Device Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>966</td>
<td>966</td>
<td>39300</td>
<td>0</td>
<td>0</td>
<td>4521</td>
</tr>
<tr>
<td>966</td>
<td>966</td>
<td>39300</td>
<td>0</td>
<td>0</td>
<td>5652</td>
</tr>
</tbody>
</table>

1 Device Type(s) found

---

### Interface Report

<table>
<thead>
<tr>
<th>Total</th>
<th>Input</th>
<th>Data Output</th>
<th>Data First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>966</td>
<td>966</td>
<td>39300</td>
<td>0</td>
<td>0</td>
<td>4521</td>
</tr>
<tr>
<td>1657</td>
<td>1657</td>
<td>7284</td>
<td>0</td>
<td>0</td>
<td>6376</td>
</tr>
<tr>
<td>2623</td>
<td>2623</td>
<td>112084</td>
<td>0</td>
<td>0</td>
<td>7143</td>
</tr>
</tbody>
</table>

2 Interface(s) found

---

### Interface Address Report

<table>
<thead>
<tr>
<th>Total</th>
<th>Input</th>
<th>Data Output</th>
<th>Data First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
<th>Interface Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>966</td>
<td>966</td>
<td>39300</td>
<td>0</td>
<td>0</td>
<td>4521</td>
</tr>
<tr>
<td>1557</td>
<td>1557</td>
<td>68384</td>
<td>0</td>
<td>0</td>
<td>5652</td>
</tr>
</tbody>
</table>

---
### JobName Report

<table>
<thead>
<tr>
<th>Total Input</th>
<th>Data Output</th>
<th>Data First yyyy/mm/dd hh.mm.ss</th>
<th>Last yyyy/mm/dd hh.mm.ss JobName</th>
</tr>
</thead>
<tbody>
<tr>
<td>2610</td>
<td>2610</td>
<td>110984</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>144</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>416</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>123</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>377</td>
<td>0</td>
</tr>
<tr>
<td>2623</td>
<td>2623</td>
<td>112084</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>4521</td>
<td>2002/11/20 17:38:32</td>
</tr>
<tr>
<td>7143</td>
<td>7143</td>
<td>2002/11/20 18:09:17</td>
<td>FTPD1</td>
</tr>
<tr>
<td>4591</td>
<td>4591</td>
<td>2002/11/20 17:39:16</td>
<td>INETDCS3</td>
</tr>
<tr>
<td>4623</td>
<td>4623</td>
<td>2002/11/20 17:40:48</td>
<td>TRMD</td>
</tr>
<tr>
<td>5653</td>
<td>5653</td>
<td>2002/11/20 17:57:37</td>
<td>USER17</td>
</tr>
</tbody>
</table>

### Asid Report

<table>
<thead>
<tr>
<th>Total Input</th>
<th>Data Output</th>
<th>Data First yyyy/mm/dd hh.mm.ss</th>
<th>Last yyyy/mm/dd hh.mm.ss Asid</th>
</tr>
</thead>
<tbody>
<tr>
<td>2623</td>
<td>2623</td>
<td>112084</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>4521</td>
<td>2002/11/20 17:38:32</td>
</tr>
<tr>
<td>7143</td>
<td>7143</td>
<td>2002/11/20 18:09:17</td>
<td>01F7</td>
</tr>
</tbody>
</table>

### Protocol Report

<table>
<thead>
<tr>
<th>Total Input</th>
<th>Data Output</th>
<th>Data First yyyy/mm/dd hh.mm:ss</th>
<th>Last yyyy/mm/dd hh.mm:ss Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12</td>
<td>656</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>644</td>
<td>0</td>
</tr>
<tr>
<td>2623</td>
<td>2623</td>
<td>112084</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4521</td>
<td>2002/11/20 17:38:32</td>
</tr>
<tr>
<td>5892</td>
<td>5892</td>
<td>2002/11/20 18:00:07</td>
<td>6(TCP)</td>
</tr>
</tbody>
</table>

### IP Address Report

<table>
<thead>
<tr>
<th>Total Input</th>
<th>Data Output</th>
<th>Data First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>11</td>
<td>484</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>1288</td>
<td>0</td>
</tr>
</tbody>
</table>

### Qos Report

<table>
<thead>
<tr>
<th>Total Input</th>
<th>Data Output</th>
<th>Data First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss Qos</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7</td>
<td>392</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>392</td>
<td>0</td>
</tr>
</tbody>
</table>

### Tcp Port Report

<table>
<thead>
<tr>
<th>Total Input</th>
<th>Data Output</th>
<th>Data First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss Tcp Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>2605</td>
<td>2605</td>
<td>110704</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>40</td>
<td>0</td>
</tr>
</tbody>
</table>

### Udp Port Report

<table>
<thead>
<tr>
<th>Total Input</th>
<th>Data Output</th>
<th>Data First yyyy/mm/dd hh:mm:ss</th>
<th>Last yyyy/mm/dd hh:mm:ss Udp Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>644</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>144</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>123</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>144</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>233</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>1288</td>
<td>0</td>
</tr>
</tbody>
</table>

Chapter 5. TCP/IP services traces and IPCS support
The following describes numbered areas of the example.
The standard statistics shown with all executions of the SYSTCPIS packet trace formatter.

- **2,623 Ctrace records processed**
  The total number of Ctrace records given to the SYSTCPIS packet trace formatted.

- **0 segmented trace records read**
  The total number of packets that spanned multiple Ctrace records.

- **0 segmented trace records were lost**
  The total number of packets records that could not be put back together.

- **2,623 trace records read**
  The total number of complete trace records.

- **0 records could not be validated**
  The number of incomplete Ctrace records that could not be used.

- **2,623 records passed filtering**
  The number of records that were successfully formatted.

- **2,623 packet trace records processed**
  The number of records that were packet trace records.

- **0 data trace records processed**
  The number of records that were data trace records.

Probe report, which is the total by ProbeID.

Device type report, which is the totals by device type.

Interface report, which is the totals by interface.

Interface address report, which is the totals interface address.

Jobname report, which is the totals by jobname.

ASID report, which is the totals address space identifier.

Protocol report, which is the totals by protocol.

IP address report, which is the totals by IP address. Both the destination and source IP addresses are counted, except when they are the same in a record.

QOS report, which is the totals by QOS.

TCP port report, which is the totals by TCP port number. Both the destination and source port numbers are counted, except when they are the same in a record.
UDP port report, which is the totals by UPD port number. Both the destination and source port numbers are counted, except when they are the same in a record.

CID report, which is the totals by connection identifier.

Group report, which is the totals by group, first byte PROBEID.

Type report, which is the totals by type, first two bytes of PROBEID.

Correlator report, which is the totals by correlator.

Protocol summary report, which is the summary based on protocol.

Session summary report, which is the summary based on session.
STREAM

Purpose: There are times when messages span multiple packets. TELNET and DNS are examples. The STREAM report (with the DUMP or FORMAT keywords) capture the entire stream of data.

Format: The following command was used to obtain the example of this report.

CTRACE COMP(SYSTCPIS) SUB((TCPICS)) SHORT DSN('IBMUSER.CTRACE1') OPTIONS((OPT STREAM DUMP ASCII))

Examples:

COMPONENT TRACE SHORT FORMAT
SYSTYPE(MVS110)
COMP(SYSTCPIS) SUBNAME((TCPICS))
OPTIONS((OPT STREAM DUMP ASCII))
DSNAME('IBMUSER.CTRACE1')

OPTIONS((Ascii Bootp(67,68) Cleanup(500) DelayAck(200,200) Domain(53)
Dumpp(65535) Finger(79) Flags() Ftp(20,21) Gain(125,250) Gopher(70)
Limit(99999999) Gnt Htp(123) Option Noreassembly Router(520) Rpc(111)
Segment Smtp(25) Snmp(161,162) Speed(10,10) Streams(131072,Summary)
Telnet(23) Tftp(69) Time(37) Userexit() Www(80))

**** 2002/11/20
RcdNr Sysname Mmhnemonic Entry Id Time Stamp Description
----- -------- --------- -------- --------------- --------------------------------

===============================================================================
1 Streams Report
2618 Streams found
611952 bytes of storage for the session report was allocated
348160 bytes of storage for buffers was allocated

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

2 Session: 9.32.74.253-0 9.42.104.38-0 ICMP
From: 2002/11/20 18:00:06.827658 to: 2002/11/20 18:00:07.149355
2 packets found
Stream buffer at 16743000 for 20480 bytes. 56 bytes were used
2 packets moved for 56 bytes
I - Inbound packet
O - Outbound packet

3 D Rcd # Time Delta Seq # Position Length End_Pos
I 5870 18:00:06.827658 00:00:00.000000 0 0 28 28
000000 45000028 1F9BF000 0106EC56 09A0A826 09A032EF 00155E44 76F6CB58 [E...[......V...h&...2...D..1.x...]
I 5892 18:00:07.149355 00:00:00.321697 28 28 28 56
000000 4500002C | E...|
000020 1FDC0000 0106EC11 09A0A826 09A032EF 00155E44 76F6CB58 [E...[......V...h&...2...D..1.x...]

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

SYSTCPIS Trace Statistics
2,623 ctrace records processed
0 segmented trace records read
0 segmented trace records were lost
2,623 trace records read
0 records could not be validated
2,623 records passed filtering
2,623 packet trace records processed
0 data trace records processed
TCP/IP Services component trace is also available for use with the OSA-Express Network Traffic Analyzer (OSAENTA) trace facility. The OSAENTA trace is a diagnostic method for obtaining frames flowing to and from an OSA adapter. You can use the OSAENTA statement to copy frames as they enter or leave an OSA adapter for an attached host. The host can be an LPAR with z/OS, VM, or Linux. You can then examine the contents of the copied frames. To be traced, the frame must meet all the conditions specified on the OSAENTA statement or the OSAENTA command.

The OSAENTA trace process

Trace data is collected as frames enter or leave an OSA adapter for a connected host. The actual collection occurs within the device drivers of OSA cards, capturing the data at the point where it has just been received from or sent to the network.

Frames that are captured have extra information added to them before they are stored. This extra information, such as timestamps, is used during the packet formatting. The captured data reflects exactly what the network sees. For example, the trace contains the constituent packets of a fragmented packet exactly as they are received or sent.

The selection criteria for choosing packets to trace are specified through the OSAENTA statement or OSAENTA command. Refer to z/OS Communications Server: IP Configuration Reference for more information about the OSAENTA statement and refer to z/OS Communications Server: IP System Administrator’s Commands for more information about the OSAENTA command.

The OSAENTA trace can have performance implications if you do not specify sufficient trace filters before enabling the trace. OSAENTA can reduce the amount of traffic the OSA-Express feature can process and the amount of traffic that can be accelerated through that OSA-Express. Also, host processing to collect the OSAENTA trace records can increase host CPU consumption. Specify sufficient filters to limit the amount of traffic that is traced to only what is necessary for problem diagnosis.

Figure 24 on page 185 illustrates the overall control and data flow in the OSAENTA tracing facility.
Starting OSAENTA trace

You can start an OSAENTA trace in one of the following ways:

- Using the V TCPIP,OSAENTA command
  ```
  V TCPIP,tcpprocname,OSAENTA,ON,PORTNAME=OSA4,IPADDR=9.1.27.2
  ```
- Using the OSAENTA statement in TCPIP.PROFILE
  ```
  OSAENTA ON PORTNAME=OSA4 IPADDR=9.1.27.2
  ```

**Security Rule:** To use any VARY command, the user must be authorized in RACF. The OPERCMDS RACF profile for each user must have access for a resource of the form MVS.VARY.TCPIP.OSTAENTA.

Traces are placed in an internal buffer, which can then be written out using a CTRACE external writer. The MVS TRACE command must also be issued for component SYSTCPOT to activate the OSAENTA trace.

After starting OSAENTA trace, you can display the status using the Netstat command, as shown in the following example:

```
D TCPIP,TCPCS,NETSTAT,DEV
  DEVNAME: OSA4
  DEVTYPE: MPCIPA
  DEVSTATUS: READY
  LNKNAME: LOSAFE
  LNKTYPE: IPAQENET
  LNKSTATUS: READY

OSA-Express Network Traffic Analyzer Information:
  OSA PortName: OSA4
  OSA DevStatus: Ready
```
If you are a TSO user, use the NETSTAT DEVlinks command.

**Modifying options with VARY commands**

After starting an OSAENTA trace, you can change the trace using the VARY command. For example, if you want to change the trace to abbreviate the data being traced, use the following command:

```
V TCPIP,tcipproc,OSAENTA,ON,ABBREV=480
```

You can display the results of the VARY command using Netstat:

```
netstat -p TCPCS -d
```

OSA Express Network Traffic Analyzer Information:

If you are a TSO user, use the NETSTAT DEVlinks command.

**Modifying options with VARY commands**

After starting an OSAENTA trace, you can change the trace using the VARY command. For example, if you want to change the trace to abbreviate the data being traced, use the following command:

```
V TCPIP,tcipproc,OSAENTA,ON,ABBREV=480
```

You can display the results of the VARY command using Netstat:

```
netstat -p TCPCS -d
```

OSA Express Network Traffic Analyzer Information:
If you are a TSO user, use the NETSTAT DEVlinks option.

You can use the VARY TCPIP,TCPPROC,OBEYFILE command to make temporary
dynamic changes to system operation and network configuration without stopping
and restarting the TCP/IP address space. For example, if you started the address
space TCPIPA and created a sequential data set USER99.TCPIP.OBEYFILE2
containing OSAENTA statements, issue the following command:

VARY TCPIP,TCPIPA,CMD=OBEYFILE,DSN=USER99.TCPIP.OBEYFILE2

The VARY TCPIP,OSAENTA command is cumulative. You can trace all packets for
specified IP addresses by entering multiple OSAENTA commands. In the following
example, the five commands disable the current trace, clear any previous trace
filters, trace all the frames received and all the frames sent for the specified IP
addresses, and activate the OSAENTA trace facility.

VARY TCPIP,,OSAENTA,OFF,PORTNAME=OSA4
VARY TCPIP,,OSAENTA,CLEAR,PORTNAME=OSA4,ABBREV=200,FRAMES=8000
VARY TCPIP,,OSAENTA,PORTNAME=OSA4,IPADDR=10.27.142.44
VARY TCPIP,,OSAENTA,PORTNAME=OSA4,IPADDR=10.27.142.45
VARY TCPIP,,OSAENTA,ON,PORTNAME=OSA4

**Formatting OSA traces using IPCS**

The IPCS CTRACE command parameters are described in "Formatting component
traces" on page 55. The following notes apply to the IPCS CTRACE parameters
with regard to the OSAENTA trace formatter:

**JOBLIST, JOBNAME**

The LINKNAME and JOBNAME keywords in the OPTIONS string can also be
used to select records.

**TALLY**

Equivalent to the STATISTICS(DETAIL) option.

**START, STOP**

The time is set when the record was moved to the trace buffer, not when the
OSA card recorded the data.

**LIMIT**

See the RECORDS keyword in the OPTIONS string.

**USEREXIT**

The packet trace formatter calls the CTRACE USEREXIT before testing the
records with the filtering criteria. If it returns a nonzero return code, then the
record is skipped. The USEREXIT can also be used in the OPTIONS string. It is
called after the record has met all the filtering criteria in the OPTIONS string.

**COMP**

Must be SYSTCPOT.

**SUB**

The SUB must name the TCP/IP procedure that created the CTRACE records
when the input is a dump data set.

**EXCEPTION**

Since there are no EXCEPTION records for OSAENTA trace, the EXCEPTION
keyword must not be specified.

**ENTIDLIST**

The following are the valid values for OSAENTA trace:

- 7 Link Frame trace records
The CTRACE OPTIONS string provides a means of entering additional keywords for record selection and formatting OSA traces (COMP=SYSTCPOT). See “Syntax” on page 57 for the complete syntax of CTRACE.

The same program is used to format OSA traces as well as packet traces. See “OPTIONS syntax” on page 98 for the values specified for the OPTIONS keyword.

Network security services (NSS) server trace (SYSTCPNS)

TCP/IP Services component trace is also available for use with the network security services server. See “TCP/IP services component trace for the network security services (NSS) server” on page 368.

Defense Manager daemon (DMD) trace (SYSTCPDM)

TCP/IP Services component trace is also available for use with the Defense Manager daemon (DMD). See “TCP/IP services component trace for the Defense Manager daemon” on page 719.

OMPROUTE trace (SYSTCPRT)

TCP/IP Services component trace is also available for use with the OMPROUTE application. See “TCP/IP services component trace for OMPROUTE” on page 772.

RESOLVER trace (SYSTCPR)

TCP/IP Services component trace is also available for use with the RESOLVER application. See Chapter 39, “Diagnosing resolver problems,” on page 853.

Configuration profile trace

You can use the ITRACE statement in the PROFILE.TCPIP data set to activate TCP/IP run-time tracing for configuration, the TCP/IP SNMP subagent, commands, and the autolog subtask. ITRACE should only be set at the direction of an IBM Support Center representative.

Following are descriptions of the ITRACE parameters:

**ON**

Select ON to establish run-time tracing. ITRACE ON commands are cumulative until an ITRACE OFF is issued.
OFF
Select OFF to terminate run-time tracing.

CONFig
Turn internal trace for configuration ON or OFF.

SUBAgent
Turn internal trace for TCP/IP SNMP subagent ON or OFF.

COMMAND
Turn internal trace for command ON or OFF.

AUTODAEMON
Turn internal trace for the autolog subtask ON or OFF.

level
Indicates the tracing level to be established. Levels are as follows:

Levels for CONFIG
1  ITRACE for all of config
2  General level of tracing for all of config
3  Tracing for configuration set commands
4  Tracing for configuration get commands
5  Tracing for syslog calls issued by config
100 Tracing for the parser
200 Tracing for scanner
300 Tracing for mainloop
400 Tracing for commands

Levels for SUBAGENT
1  General subagent tracing
2  General subagent tracing, plus DPI traces
3  General subagent tracing, plus extended storage dump traces
4  All trace levels

Level for COMMAND
1  ITRACE for all commands

Following is an example illustrating how to use the ITRACE command:
ITRACE ON CONFIG 3
ITRACE OFF SUBAGENT

Trace output is sent to the following locations:
• Subagent trace output is directed to the syslog daemon. This daemon is configured by the /etc/syslog.conf file and must be active.
• AUTOLOG trace output goes to ALGPRINT.
• Trace output for other components goes to SYSPRINT.
Chapter 6. IPCS subcommands for TCP/IP

Use the IPCS subcommands for TCP/IP to format data from IPCS system dumps. This topic describes the subcommands (including description, syntax, parameters, and sample output), installation, entering, and execution, and includes the following sections:

- “TCPIPCS command” on page 194
- “TCPIPCS subcommands” on page 197
- “ERRNO command” on page 293
- “IPHDR” on page 297
- “RESOLVER” on page 299
- “SETPRINT” on page 304
- “SKMSG” on page 304
- “TCPHDR” on page 306
- “TOD” on page 307
- “UDPHDR” on page 308
- “Installing TCP/IP IPCS subcommands by using the panel interface” on page 310
- “Entering TCP/IP IPCS subcommands” on page 310

Types of subcommands

There are two types of subcommands. These are described as follows:

- Many of the TCP/IP subcommands work on a specific stack or Telnet instance. These subcommands are grouped under the TCPIPCS subcommand to share the TCP (to select the stack or Telnet) and TITLE options. A subset of these commands are available for work with an instance of Telnet. If available, “Available for Telnet” appears at the end of the description in Table 15.
- The remaining TCP/IP IPCS subcommands do not require a TCP/IP stack, and they are not under the TCPIPCS subcommand.

Restriction: The TCP/IP IPCS commands are not supported for IPCS “active.”

Table 15 lists all the IPCS subcommands. The TCPIPCS commands are shown first, followed by the general commands.

Table 15. TCP/IP IPCS commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Alias</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCPIPCS ALL</td>
<td>Equivalent to TCPIPCS STATE TSEB TSDB TSDX DUAF CONFIG ROUTE SOCKET STREAM RAW TCB UDP LOCK TIMER STORAGE</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>TCPIPCS API</td>
<td>Display control blocks for Sockets Extended Assembler Macro and Pascal APIs</td>
<td></td>
<td>“TCPIPCS API” on page 197</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Alias</td>
<td>See</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>TCPIPCS CONFIG</td>
<td>Display device configuration information</td>
<td>TCPIPCS CNFG TCPIPCS CONF</td>
<td>“TCPIPCS CONFIG” on page 199</td>
</tr>
<tr>
<td>TCPIPCS CONNECTION</td>
<td>Display active or all connections</td>
<td>TCPIPCS CONN</td>
<td>“TCPIPCS CONNECTION” on page 200</td>
</tr>
<tr>
<td>TCPIPCS COUNTERS</td>
<td>Display information about TCP/IP internal execution statistics</td>
<td></td>
<td>“TCPIPCS COUNTERS” on page 202</td>
</tr>
<tr>
<td>TCPIPCS DETAIL</td>
<td>Equivalent to TCPIPCS TSEB TSDB TSDX DUAF Available for Telnet.</td>
<td>TCPIPCS CBS</td>
<td>NA</td>
</tr>
<tr>
<td>TCPIPCS DU</td>
<td>Equivalent to TCPIPCS DUAF DUCB Available for Telnet.</td>
<td></td>
<td>NA</td>
</tr>
<tr>
<td>TCPIPCS DUAF</td>
<td>Summarize DUCBs Available for Telnet.</td>
<td>TCPIPCS DUCBS</td>
<td>“TCPIPCS DUAF” on page 204</td>
</tr>
<tr>
<td>TCPIPCS DUCB</td>
<td>Find and format DUCBs Available for Telnet.</td>
<td></td>
<td>“TCPIPCS DUCB” on page 206</td>
</tr>
<tr>
<td>TCPIPCS FRCA</td>
<td>Display state information about FRCA connections and objects</td>
<td></td>
<td>“TCPIPCS FRCA” on page 210</td>
</tr>
<tr>
<td>TCPIPCS HASH</td>
<td>Display TCP/IP data stored in hash tables</td>
<td></td>
<td>“TCPIPCS HASH” on page 212</td>
</tr>
<tr>
<td>TCPIPCS HEADER</td>
<td>Display dump Header info</td>
<td>TCPIPCS HDR</td>
<td>“TCPIPCS HEADER” on page 216</td>
</tr>
<tr>
<td>TCPIPCS HELP</td>
<td>Display syntax help for TCPIPCS command</td>
<td>TCPIPCS ?</td>
<td>“TCPIPCS HELP” on page 218</td>
</tr>
<tr>
<td>TCPIPCS IPSEC</td>
<td>Display information about IP security filters and tunnels</td>
<td></td>
<td>“TCPIPCS IPSEC” on page 219</td>
</tr>
<tr>
<td>TCPIPCS LOCK</td>
<td>Display locks Available for Telnet.</td>
<td>TCPIPCS LOCKSUM</td>
<td>“TCPIPCS LOCK” on page 222</td>
</tr>
<tr>
<td>TCPIPCS MAP</td>
<td>Display storage map</td>
<td></td>
<td>“TCPIPCS MAP” on page 223</td>
</tr>
<tr>
<td>TCPIPCS MTABLE</td>
<td>Display module table</td>
<td></td>
<td>“TCPIPCS MTABLE” on page 226</td>
</tr>
<tr>
<td>TCPIPCS POLICY</td>
<td>Display service policy data</td>
<td></td>
<td>“TCPIPCS POLICY” on page 227</td>
</tr>
<tr>
<td>TCPIPCS PROFILE</td>
<td>Display TCP/IP configuration data in the format of a profile dataset</td>
<td>TCPIPCS PROF</td>
<td>“TCPIPCS PROFILE” on page 229</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Alias</td>
<td>See</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------------------</td>
<td>-----------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>TCPIPCS</td>
<td>Equivalents to TCPIPCS</td>
<td></td>
<td><strong>TCPIPCS SUMMARY</strong> on page 242</td>
</tr>
<tr>
<td>PROTOCOL</td>
<td>Displays general stack information</td>
<td>TCPIPCS STATE</td>
<td><strong>TCP/IP CP</strong> on page 242</td>
</tr>
<tr>
<td>TCPIPCS RAW</td>
<td>Displays RAW control blocks</td>
<td>TCPIPCS MRCB</td>
<td><strong>TCPIPCS RAW</strong> on page 235</td>
</tr>
<tr>
<td>TCPIPCS RCB</td>
<td></td>
<td>TCPIPCS RAWSUM</td>
<td><strong>TCPIPCS RAW</strong> on page 235</td>
</tr>
<tr>
<td>TCPIPCS ROUTE</td>
<td>Displays routing information</td>
<td>TCPIPCS RTE</td>
<td><strong>TCP/IP CP</strong> on page 237</td>
</tr>
<tr>
<td>TCPIPCS SOCKET</td>
<td>Displays socket information</td>
<td>TCPIPCS SCB</td>
<td><strong>TCP/IP CP</strong> on page 240</td>
</tr>
<tr>
<td>TCPIPCS SCB</td>
<td></td>
<td>TCPIPCS SOCKSUM</td>
<td><strong>TCP/IP CP</strong> on page 240</td>
</tr>
<tr>
<td>TCPIPCS STATE</td>
<td>Displays TCP/IP storage usage</td>
<td>TCPIPCS STOR</td>
<td><strong>TCP/IP CP</strong> on page 264</td>
</tr>
<tr>
<td>TCPIPCS STREAM</td>
<td>Displays streams information</td>
<td>TCPIPCS SKSH</td>
<td><strong>TCP/IP CP</strong> on page 266</td>
</tr>
<tr>
<td>TCPIPCS STREAM</td>
<td></td>
<td>TCPIPCS STREAMS</td>
<td><strong>TCP/IP CP</strong> on page 266</td>
</tr>
<tr>
<td>TCPIPCS SUMMARY</td>
<td>Equivalent to TCPIPCS DUAF CONFIG SOCKET</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 281</td>
</tr>
<tr>
<td>TCPIPCS TCB</td>
<td>Displays TCP protocol control blocks</td>
<td>TCPIPCS MTCB</td>
<td><strong>TCP/IP CP</strong> on page 268</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TCPIPCS TCBSUM</td>
<td><strong>TCP/IP CP</strong> on page 268</td>
</tr>
<tr>
<td>TCPIPCS TCB</td>
<td>Displays TCP protocol control blocks</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 268</td>
</tr>
<tr>
<td>TCPIPCS TELNET</td>
<td>Displays Telnet information</td>
<td>TCPIPCS TCB</td>
<td><strong>TCP/IP CP</strong> on page 268</td>
</tr>
<tr>
<td></td>
<td>Available for Telnet.</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 270</td>
</tr>
<tr>
<td>TCPIPCS TIMER</td>
<td>Displays information about timers</td>
<td>TCPIPCS TIMESUM</td>
<td><strong>TCP/IP CP</strong> on page 272</td>
</tr>
<tr>
<td></td>
<td>Available for Telnet.</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 272</td>
</tr>
<tr>
<td>TCPIPCS TRACE</td>
<td>Displays TCP/IP CTrace information</td>
<td>TCPIPCS TCA</td>
<td><strong>TCP/IP CP</strong> on page 277</td>
</tr>
<tr>
<td></td>
<td>Available for Telnet.</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 277</td>
</tr>
<tr>
<td>TCPIPCS TREE</td>
<td>Displays information about data stored in Patricia trees</td>
<td>TCPIPCS TREESUM</td>
<td><strong>TCP/IP CP</strong> on page 277</td>
</tr>
<tr>
<td></td>
<td>Available for Telnet.</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 277</td>
</tr>
<tr>
<td>TCPIPCS TSDB</td>
<td>Formats TSDB</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 281</td>
</tr>
<tr>
<td></td>
<td>Available for Telnet.</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 281</td>
</tr>
<tr>
<td>TCPIPCS TSDX</td>
<td>Formats TSDX</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 282</td>
</tr>
<tr>
<td></td>
<td>Available for Telnet.</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 282</td>
</tr>
<tr>
<td>TCPIPCS TSEB</td>
<td>Formats TSEB</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 283</td>
</tr>
<tr>
<td></td>
<td>Available for Telnet.</td>
<td></td>
<td><strong>TCP/IP CP</strong> on page 283</td>
</tr>
</tbody>
</table>
### Table 15. TCP/IP IPCS commands (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Alias</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCPIPCS TTLS</td>
<td>Display state information about AT-TLS connections and groups</td>
<td>TCPIPCS MUCB, TCPIPCS UCB, TCPIPCS UDPSUM</td>
<td>“TCPIPCS TTLS” on page 284</td>
</tr>
<tr>
<td>TCPIPCS UDP</td>
<td>Display UDP control blocks</td>
<td>TCPIPCS MUCB, TCPIPCS UCB</td>
<td>“TCPIPCS UDP” on page 288</td>
</tr>
<tr>
<td>TCPIPCS VMCF</td>
<td>Display information about VMCF and IUCV users</td>
<td>TCPIPCS UDPSUM</td>
<td>“TCPIPCS VMCF” on page 290</td>
</tr>
<tr>
<td>TCPIPCS XCF</td>
<td>Display information about XCF links and dynamic VIPA</td>
<td></td>
<td>“TCPIPCS XCF” on page 292</td>
</tr>
<tr>
<td>ERRNO</td>
<td>Interprets error numbers available for Telnet.</td>
<td></td>
<td>“ERRNO” command” on page 293</td>
</tr>
<tr>
<td>ICMPHDR</td>
<td>Format an ICMP header</td>
<td></td>
<td>“ICMPHDR” on page 296</td>
</tr>
<tr>
<td>IPHDR</td>
<td>Format an IP header</td>
<td></td>
<td>“IPHDR” on page 297</td>
</tr>
<tr>
<td>RESOLVER</td>
<td>Format and summarize resolver control blocks and cache information</td>
<td></td>
<td>“RESOLVER” on page 299</td>
</tr>
<tr>
<td>SETPRINT</td>
<td>Set destination so the IPCS subcommand output is sent to a user ID or the printer available for Telnet.</td>
<td></td>
<td>“SETPRINT” on page 304</td>
</tr>
<tr>
<td>SKMSG</td>
<td>Format a stream message available for Telnet.</td>
<td></td>
<td>“SKMSG” on page 304</td>
</tr>
<tr>
<td>TCPHDR</td>
<td>Format a TCP header</td>
<td></td>
<td>“TCPHDR” on page 306</td>
</tr>
<tr>
<td>TOD</td>
<td>Convert a S/390® 64-bit time-of-day timestamp to readable date and time available for Telnet.</td>
<td></td>
<td>“TOD” on page 307</td>
</tr>
<tr>
<td>UDPHDR</td>
<td>Format UDP header</td>
<td></td>
<td>“UDPHDR” on page 308</td>
</tr>
</tbody>
</table>

### TCPIPCS command

This section describes the TCPIPCS command.

### Syntax

The command syntax for all TCPIPCS subcommands includes an option to specify the TCP stack and to specify whether the title is displayed.
Parameters

The parameters for the TCPIPCS command are described below.

subcommand

Default is STATE.

parameters

Each subcommand has its own parameters.

- If a command has variable parameters, they can be omitted, specified as a single variable, or specified as a list. If no variable parameters are specified, an asterisk must be used as a placeholder if any keyword parameters are specified. If two or more variable parameters are specified, they must be enclosed in parentheses.

- To distinguish between the variable parameters, a parameter is assumed to be one of the following:
  - An index or small number if it is four digits or less, begins with zero to nine, and contains only hexadecimal digits (0–9, a–f, A–F). If a command accepts multiple indices or small numbers, both are compared to the values and the first matching field is used.
  - An address if it is more than four digits, begins with zero to nine, and contains only hexadecimal digits. For example, for the TCPIPCS DUAF command, both the DUCB and ASCB addresses of each DUCB are compared to the address parameter, and the first matching field is used to select the DUCB to display.
  - An IPCS symbol name can also be specified for an address.
  - Otherwise, the parameter is assumed to be a character string variable (such as TCP/IP procedure or job name, user ID, and command name).

- Keyword parameters can be in any order.

- If there are both keyword and variable parameters, all variable parameters must precede the keywords.

TCP

Specifies which TCP/IP stack or Telnet instance. When issuing commands for Telnet, the Telnet procedure name must be specified in the tcp_proc_name variable. The stack can be specified directly or indirectly. A stack can be specified directly by coding the TCP parameter with either tcp_proc_name or tcp_index. If no stack is specified directly, the output is reported for the stack with the lowest index matching the release of the TCPIPCS command. After a particular stack is specified (whether specified directly or indirectly), that stack becomes the default. The stack index is saved as a symbol and is used as the default in future invocations of the TCPIPCS command. An alias for the TCP option is PROC.
Note: All eight stack indices are available when TCP/IP or Telnet starts, so any stack index can be selected. The existence of an index does not necessarily mean this stack can be formatted. If the stack was not included in the dump, then most of the information about a stack cannot be formatted. Most TCP/IP control blocks are in the private TCP address space. All Telnet control blocks are in the private Telnet address space.

The fact that an index exists does not necessarily mean this stack index has ever been used. If you specify a stack index that has not been used, the version and release fields for this stack are zero, so you receive a message indicating the stack is not the same version and release as the TCPIPCS command:

tcp_proc_name
   TCP/IP procedure name or the Telnet procedure name (when the TN3270E Server is running in its own address space).

tcp_index
   TCP/IP stack index (1-8) or Telnet index (9-16).

TITLE
   The title contains information about the dump and about the TCPIPCS command. By default, the title information is displayed.

   The title contains the following information.
   - TCPIPCS command input parameters.
   - Dump data set name.
   - Dump title.
   - TSAB address.
   - Table listing all TCP/IP stacks used in the dump and their
     - TSEB address
     - Stack index
     - Procedure name
     - Stack version
     - TSDB address
     - TSDX address
     - ASID
     - Trace option bits
     - Stack status
   - Count of the number of TCP/IP stacks defined (used).
   - Count of the number of active TCP/IP stacks found.
   - Count of the number of active TCP/IP stacks matching the TCPIPCS command version and release.
   - Procedure name and index of the stack being reported.

NOTITLE
   Suppress the title lines. This is useful when you are processing many commands on the same dump and do not want to see the title information repeated.

Restriction: If you specify multiple keywords from the set [TITLE, NOTITLE], only the last one is used.
Symbols defined

TCPIPCS defines the following IPCS symbols:

**TSEBPTR**
The address of the first TSEB control block.

**TSEB**\(_n\)**
The address of the TSEB control block corresponding to the stack index \(n\).

TCPIPCS subcommands

This section describes the TCPIPCS subcommands.

TCPIPCS API

Use this subcommand to display information about the connects in the Sockets Extended Assembler Macro Application Programming Interface (Macro API) and the Pascal API.

**Note:** The Macro API is the base for the CALL Instruction API, the CICS C API, and the CICS EZACICAL API. Refer to the [z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference](https://www.ibm.com/support/docview.wss?uid=swg27029891) for more information about the native TCP/IP APIs.

Some API control blocks are in the application address space, which might not be available in the dump. If the application address space is available, the API control blocks are formatted.

**Syntax**

```
TCPIPCS API

variable_item (variable_list)

TCP (tcp_proc_name)
```

**Parameters**

If no parameters are specified, only information about the Macro API is summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

**variable_item**

Any one of the following variable parameters.

**variable_list**

You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:
Variable parameters are:

**jobname**
Displays only the API control blocks for this job name. The job name can be a TCP/IP application name or a stack name. Must contain from 1-8 characters.

**ASCB_address**
Displays the API control blocks with this address space control block (ASCB) address. An IPCS symbol name can be specified for the address. The address is specified as 1-8 hexadecimal digits. If an address begins with digit A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

**ASID_number**
Displays the API control blocks with this address space identifier (ASID). The ASID is a hexadecimal number containing 1 - 4 digits.

In addition to the variable parameters, you can specify the following keyword parameters:

**MACRO**
Displays only information for Macro APIs. MACRO is the default.

**PASCAL**
Displays only information for Pascal APIs.

**ALL**
Displays information for both APIs.

**SUMMARY**
Displays the addresses of the control blocks and other data in tables. SUMMARY is the default.

**DETAIL**
Also displays the contents of the control blocks in addition to the SUMMARY display.

**TCP, TITLE, NOTITLE**
See "Parameters" on page 195 for a description of these parameters.

**Restrictions:** Be aware of the following keyword restrictions:

- If you specify multiple keywords from the set {MACRO, PASCAL, ALL}, only the last one is used.
- If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

**Sample output of the TCPIPICS API subcommand**
The following is sample output of the TCPIPICS API subcommand.

The contents of the SDST control blocks are formatted by the TCPIPICS API subcommand if the DETAIL option is coded on the command (SUMMARY is the default and only the address of the SDST will be displayed in this case).

R14 Output:

```
-- Array elements --
:: 
+0082  SDST_LOCAL_IPADDRLEN. 00
+0083  SDST_REMOTE_IPADDRLEN. 00
```
TCPIPCS CONFIG

Use this subcommand to display each device interface, physical interface, and logical interface. The configuration summary table shows each logical interface with the name of its associated device and link.

Syntax

```
TCPIPCS CONFIG (SUMMARY) (DETAIL)
```

Parameters

**SUMMARY**
Displays each device, physical interface, and logical interface, and summarizes them all in one cross-reference table. SUMMARY is the default.

**DETAIL**
In addition to the SUMMARY display, DETAIL also shows the interface cross-reference reports.

**TCP, TITLE, NOTITLE**
See “Parameters” on page 195 for a description of these parameters.

Restriction: If you specify multiple keywords from the set [SUMMARY, DETAIL], only the last one is used.

Sample output of the TCPIPCS CONFIG subcommand

The following is sample output of the TCPIPCS CONFIG subcommand.

```
TCPIPCS CONFIG
Dataset: IPCS.R450697.V6TCBD1
Title: TCPCS2 CLIENT SIDE

The address of the TSAB is: 09DBE1A0
Tseb SI Procedure Version Tsub Tsdx Asid TraceOpts Status
09DBE1E0 1 TCPCS V1R5 096C4000 096C40C8 0033 10841004 Active
09DBE260 2 TCPCS2 V1R5 096C9000 096C90C8 0034 10841004 Active

2 defined TCP/IP(s) were found
2 active TCP/IP(s) were found
2 TCP/IP(s) for CS V1R5 found

Analysis of Tcp/Ip for TCPCS2. Index: 2
Configuration control block summary
IPMAIN found at 095A83D0
IPMAIN6 found at 096CE470
```
TCPIPPCS CONNECTION

Use this subcommand to display information about TCP, UDP, and raw connections. The information includes the following:

- User ID
- Connection ID
- Local IP address
- Foreign IP address
- Connection state (for TCP connections only)
- Protocol name (for raw connections only)

Syntax

```
TCPIPPCS CONNECTION (ACTIVE) TCP (tcp_proc_name tcp_index)
```

Parameters

**ACTIVE**

Display only active connections. This is the default.

**Tip**: The number of connections reported for each protocol includes both inactive and active connections; therefore, the total might be higher than the number of displayed (active) connections.
ALL
Display all connections, regardless of state.

TCP, TITLE, NOTITLE
See "Parameters" on page 195 for a description of these parameters.

Restriction: If you specify multiple keywords from the set {ACTIVE, ALL}, only
the last one is used.

Sample output of the TCPIPCS CONNECTION subcommand
The following is a sample output of the TCPIPCS CONNECTION subcommand. In
this sample, the default option is ACTIVE, so only active connections are shown.
There are 6 active TCP connections, 4 active UDP connections, and 3 active RAW
connections.

TCPIPCS CONNECTION
Dataset: IPCS.R8A0723.RASDUMP
Title: EZRPE005
The address of the TSAB is: 098221F0

<table>
<thead>
<tr>
<th>Userid</th>
<th>Conn State</th>
<th>Socket</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCPCS</td>
<td>0000012</td>
<td>Listening Local Socket : 127.0.0.1..1024</td>
</tr>
<tr>
<td>BPXOINIT</td>
<td>0000019</td>
<td>Listening Local Socket : 127.0.0.1..1024</td>
</tr>
<tr>
<td>TCPCS</td>
<td>0000016</td>
<td>Established Local Socket : 127.0.0.1..1024</td>
</tr>
<tr>
<td>TCPCS</td>
<td>0000014</td>
<td>Established Local Socket : 127.0.0.1..1024</td>
</tr>
</tbody>
</table>

4 TCP IPv4 connections

Active TCP IPv6 Connections:
Userid  Conn State     Socket
FTPUNIX1 00000051 Listening Local ::0..1821
FTPMVS1 00000049 Listening Local ::0..1821

2 TCP IPv6 connections

Active UDP Unicast IPv4 Connections:
Userid  Conn Socket
PORTMAP 00000027 Local 0.0.0.0..111
OSNMPD 00000030 Local 0.0.0.0..161
MISCSRV 00000039 Local 198.11.98.124..7
MISCSRV 0000003E Local 198.11.98.124..9

4 UDP Unicast IPv4 connections

Active UDP Unicast IPv6 Connections:
Userid  Conn Socket

0 UDP Unicast IPv6 connections

Active UDP Multicast IPv4 Connections:
TCPIPCS COUNTERS

Use this subcommand to display information about TCP/IP internal execution statistics.

Syntax

```
TCPIPCS-COUNTERS( ALL | DEVICE | IF | IP | LOCK | RAW | TCP | UDP )
```

Parameters

- **ALL**
  - Display all statistics. This is the default.

- **DEVICE**
  - Display only device statistics.

- **IF**
  - Display only IF layer statistics.

- **IP**
  - Display only IP layer statistics.

- **LOCK**
  - Display only lock statistics.

- **RAW**
  - Display only RAW layer statistics.

- **TCP**
  - Display only TCP layer statistics.

- **UDP**
  - Display only UDP layer statistics.

**TCP, TITLE, NOTITLE**

See “Parameters” on page 195 for a description of these parameters.
Sample output of the TCPIPCS COUNTERS subcommand for IP UDP

The following is sample output of the TCPIPCS COUNTERS subcommand for IP UDP.

TCPIPCS COUNTERS (IP UDP)
Dataset: SYS1.DUMP00
Title: LINKDOWN

The address of the TSAB is: 15136000

<table>
<thead>
<tr>
<th>Tseb</th>
<th>SI Procedure Version Tsdb</th>
<th>Tsdx</th>
<th>Asid TraceOpts</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>15136040</td>
<td>1 TCPCS V1R7</td>
<td>1511E000</td>
<td>1511E0C8 002F 9FF767F 00000000</td>
<td>Active</td>
</tr>
</tbody>
</table>

1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
1 TCP/IP(s) for CS V1R7 found

==============================================================================

Analysis of Tcp/IP for TCPCS. Index: 1

IP Statistics

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonbat</td>
<td>3</td>
</tr>
<tr>
<td>batch</td>
<td>0</td>
</tr>
<tr>
<td>batnum</td>
<td>0</td>
</tr>
<tr>
<td>nonrsm</td>
<td>0</td>
</tr>
<tr>
<td>batrsm</td>
<td>0</td>
</tr>
<tr>
<td>rsmnum</td>
<td>0</td>
</tr>
<tr>
<td>rteadd</td>
<td>28</td>
</tr>
<tr>
<td>rtedel</td>
<td>0</td>
</tr>
<tr>
<td>rteinc</td>
<td>11</td>
</tr>
<tr>
<td>rtedec</td>
<td>8</td>
</tr>
<tr>
<td>rtpadd</td>
<td>14</td>
</tr>
<tr>
<td>rtpdel</td>
<td>0</td>
</tr>
<tr>
<td>rtechg</td>
<td>86</td>
</tr>
<tr>
<td>trredr</td>
<td>0</td>
</tr>
<tr>
<td>trsuspect</td>
<td>0</td>
</tr>
<tr>
<td>trsustain</td>
<td>0</td>
</tr>
<tr>
<td>dupfrg</td>
<td>0</td>
</tr>
<tr>
<td>dataadj1</td>
<td>0</td>
</tr>
<tr>
<td>dataadj2</td>
<td>0</td>
</tr>
</tbody>
</table>

IP6 Statistics

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonbat</td>
<td>32</td>
</tr>
<tr>
<td>batch</td>
<td>0</td>
</tr>
<tr>
<td>batnum</td>
<td>0</td>
</tr>
<tr>
<td>nonrsm</td>
<td>0</td>
</tr>
<tr>
<td>batrsm</td>
<td>0</td>
</tr>
<tr>
<td>rsmnum</td>
<td>0</td>
</tr>
<tr>
<td>rteadd</td>
<td>0</td>
</tr>
<tr>
<td>rtedel</td>
<td>0</td>
</tr>
<tr>
<td>rteinc</td>
<td>0</td>
</tr>
<tr>
<td>rtedec</td>
<td>0</td>
</tr>
<tr>
<td>rtpadd</td>
<td>11</td>
</tr>
<tr>
<td>rtpdel</td>
<td>0</td>
</tr>
<tr>
<td>rtechg</td>
<td>43</td>
</tr>
<tr>
<td>trredr</td>
<td>0</td>
</tr>
<tr>
<td>trsuspect</td>
<td>0</td>
</tr>
<tr>
<td>trsustain</td>
<td>0</td>
</tr>
<tr>
<td>dupfrg</td>
<td>0</td>
</tr>
</tbody>
</table>
dataadj1............ 0
dataadj2............ 0
lifdel.............. 0
noreclaim........... 0
hdrpullup........... 0
dadfailtot.......... 0
dadfailll........... 0

UDP Statistics

rd.................. 7
rdnum............... 7
batch............... 0
nonbat.............. 7

Analysis of Tcp/Ip for TCPCS completed

**TCPIPCS DUAF**

Use this subcommand to display a summary of each dispatchable unit control block (DUCB). Each entry in the dispatchable unit allocation table (DUAT) points to a DUCB. The DUAT entry contains the status of the DUCB and identifies the ASID with which the DUCB is associated. If no parameters are specified, the output contains a summary of the DUAT, followed by a summary of each DUCB.

The status of each DUCB is abbreviated as follows:
- **Ab** The DUCB has ABENDed.
- **Iu** The DUCB is in use.
- **Re** The DUCB is in resume state.
- **Su** The DUCB has been suspended.

The DUCB status might be followed by the recovery stack. There is one line for each register save area (RSA) found in the DUCB (and its DUSA extension, if present). The address of each RSA, its previous pointer, its next pointer, and the module name are shown.

A register save area displayed as RSA* indicates that the RSA is not in the active chain. If all RSAs are shown like this, the DUCB is not in use.

**Syntax**

```
TCPIPCS—DUAF

ALL

variable_item

variable_list

variable_list

TCP (tcp_proc_name tcp_index)

TITLE

NOTITLE
```

**Parameters**

If no parameters are specified, all active DUCBs are summarized.
An asterisk is used as a placeholder if no variable parameters are specified.

variable_item
Any one of the following variable parameters.

variable_list
You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

jobname
Displays only the DUCBs with this job name. The job name can be a TCP/IP application name or a stack name. Must contain from 1-8 characters.

DUCB_address
Displays the DUCB with this address. An IPCS symbol name can be specified for the address. The address is specified as 1-8 hexadecimal digits. If an address begins with characters A–F, prefix the address with a 0 to avoid the address being interpreted as a symbol name or as a character string.

DUCB_index
Displays this DUCB with this index. The index is a hexadecimal number containing one to four digits. The lowest index is 0.

ASCB_address
Displays the DUCB with this address space control block (ASCB) address. An IPCS symbol name can be specified for the address. The address is specified as 1-8 hexadecimal digits. If an address begins with characters A–F, prefix the address with a 0 to avoid the address being interpreted as a symbol name or as a character string.

ASID_number
Displays the DUCB with this ASID. The ASID is a hexadecimal number containing one to four digits.

In addition to the variable parameters described above, you can specify the following keyword parameters:

ALL
Display information for all active DUCBs. This is the default.

ABEND
Display only information for DUCBs that ABENDed.

INUSE
Display only information for DUCBs currently being used.

RESUME
Display only information for DUCBs that are resumed.

SUSPEND
Display only information for DUCBs that are suspended.

NORSA
Do not display the contents of the DUCBs’ register save areas (RSA). By default, the RSA contents are displayed.

TCP, TITLE, NOTITLE
See “Parameters” on page 195 for a description of these parameters.

Restriction: If you specify multiple keywords from the set {ALL, ABEND, INUSE, RESUME, SUSPEND}, only the last one is used.
Sample output of the TCPIPCS DUAF subcommand

The following is a sample output of the TCPIPCS DUAF subcommand:

```
TCPIPCS DUAF( (0876C000 0B) INUSE )
Dataset: IPCS.A594094.DUMPK
Title: TCPCS V2R10: Job(USER15 ) EZBITRAC(HTCP50A 99.266)+
       000304 50C4/00000004 TCB P=0029,S=000E,H=0019

The address of the TSAB is: 08D138C0
```

```
Tseb   SI Procedure Version Tsdx   Asid TraceOpt Status
08013900 1 TCPCS  V2R10  0885A000 0885A0C8   0029 9FFFFFF7F Active

1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
1 TCP/IP(s) for CS  V2R10 found
```

```
================================================================================
Analysis of Tcp/IP for TCPCS. Index: 1
Dispatchable Unit Summary
INDEX DUAE DUCB DUSA ASCB ASID JOBNAME ABEND STATUS
10000003 08859040 0876C000 0876C100 00FB7080 0029 TCPCS 00000000 Iu
 RSA 0876C3F8 Prev 00005D98 Next 0876C8C0 Mod EZBIEOWER
 RSA 0876C8C8 Prev 0876C3F8 Next 00000000 Mod EZBITSTO
 1384 bytes were used

1000000B 08859080 08784000 08784100 00FB7980 0019 USER15 000C4000 Ab Iu
 RSA 087843F8 Prev 09B87978 Next 08784688 Mod EZBPFS0C
 RSA 087846C0 Prev 087843F8 Next 08784988 Mod EZBPF0PN
 RSA 08784990 Prev 087846C0 Next 08784DB0 Mod EZBUDSTR
 RSA 08784DB8 Prev 08784990 Next 087855A8 Mod EZBITRAC
 4536 bytes were used
```

```
82 DU control blocks were found
12 DU control blocks were in use
0 DU control blocks were suspended
0 DU control blocks were resumed
1 DU control blocks had abended
2 DU control blocks were formatted
The maximum DUCB size found is 4536 bytes
```

Analysis of Tcp/IP for TCPCS completed

TCPIPCS DUCB

Use this subcommand to display the contents of each dispatchable unit control block (DUCB). Each entry in the dispatchable unit allocation table (DUAT) points to a DUCB. The DUAT entry contains the status of the DUCB and identifies the ASID with which the DUCB is associated. The DUAT is summarized in the output. The contents of each DUCB are then displayed, followed by each DUSA for the DUCB. The first dispatchable unit stack area (DUSA) is followed by information from each register save area (RSA). Each register from the RSA is listed, showing its address and offset from the other registers in the register save area. The address of the parameter list (pointed to by R1) and the first five words at that address are also given. Each RSA is formatted. The recovery stack is also displayed.
Syntax

Syntax of the TCPIPCS DUCB subcommand:

```
TCP {TCP (tcp_proc_name tcp_index) TITLE NOTITLE}
```

Parameters

If no parameters are specified, all DUCBs are displayed.

* An asterisk is used as a placeholder if no variable parameters are specified.

**variable_item**

Any one of the following variable parameters.

**variable_list**

You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

**jobname**

Displays only the DUCBs with this job name. The job name can be a TCP/IP application name or a stack name. Must contain from 1–8 characters.

**DUCB_address**

Displays the DUCB with this address. An IPCS symbol name can be specified for the address. The address is specified as 1–8 hexadecimal digits. If an address begins with digit A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

**DUCB_index**

Displays this DUCB with this index. The index is a hexadecimal number containing one to four digits. The lowest index is zero.

**ASCB_address**

Displays the DUCB with this address space control block address (ASCB). An IPCS symbol name can be specified for the address. The address is specified as 1–8 hexadecimal digits. If an address begins with digit A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

**asid_number**

Displays the DUCB with this ASID. The ASID is a hexadecimal number containing one to four digits.

**TCP, TITLE, NOTITLE**

See "Parameters" on page 195 for a description of these parameters.

Sample output of the TCPIPCS DUCB subcommand

In the following sample, some lines have been deleted in order to shorten the sample. Deleted lines are indicated with the following:
The following is sample output of the TCPIPCS DUCB subcommand:

```plaintext
TCPIPCS DUCB
Dataset: IPCS.RBA0723.RASDUMP
Title: EZRPE005
The address of the TSAB is: 098221F0

<table>
<thead>
<tr>
<th>Tseb</th>
<th>SI Procedure Version</th>
<th>Tsdb</th>
<th>Tsdx</th>
<th>Asid</th>
<th>TraceOpts</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TCPCS</td>
<td>V1R5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>08E85000</td>
<td>08E850C8</td>
<td>001E</td>
<td>9FFF7E7F</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>08E850C8</td>
<td>001E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>08E85000</td>
<td>08E850C8</td>
<td>001E</td>
<td>9FFF7E7F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2 defined TCP/IP(s) were found
2 active TCP/IP(s) were found
2 TCP/IP(s) for CS V1R5 found

Analysis of Tcp/Ip for TCPCS. Index: 1
DUCB Detail Analysis
Dispatchable Unit Allocation Table: 08E83010
```

### Dispatchable Unit Control Block: DUCB

```plaintext
EZBDUCB: 08D8D010
  +0000 DUCB_EYE................. DUCB
  +0004 DUCB_LENGTH............... 0100
  +0008 DUCB_VERSION.............. 0002
  +000C DUCB_TOKEN................ 08D8D010 0014001E 10000000 00000000
  +0014 DUCB_DUSA................. 08D8D110
  +0018 DUCB_AVAIL_CHAIN.......... 00000000
  +001C DUCB_DUAEP................ 08E83028
  +0020 DUCB_ASID................. 001E
  +0024 DUCB_ASCB................. 00FA4400
  +0028 DUCB_ATCB................. 007EC920
  +002C DUCB_ITCVT................ 08E853C8
  +0030 DUCB_LOCKSHELDCOUNT....... 00000000
  +0034 DUCB_LOCKS_TABLE.......... 08D8D194
  +0038 DUCB_LOCKS_SUSPENDED...... 00000000
  +003C DUCB_LOCKS_SUSPENDED_NEXT 7FFAF1
  +0040 DUCB_LOCKS_SUSPENDED_NEXT 7FFAF1
  +0044 DUCB_SUSPENOTEKEN........ 00000000 40000000
  +0048 DUCB_JOBNAME.............. TCPCS
  +004C DUCB_DUCBNAME............. TCPCS
  +0050 DUCB_DUFAIL... 08E83010
  +0054 DUCB_DUFAIL... 08E83010
  +0058 DUCB_DUFAIL... 08E83010
  +005C DUCB_DUFAIL... 08E83010
  +0060 DUCB_DUFAIL... 08E83010
  +0064 DUCB_DUFAIL... 08E83010
  +0068 DUCB_DUFAIL... 08E83010
  +006C DUCB_DUFAIL... 08E83010
  +0070 DUCB_DUFAIL... 08E83010
  +0074 DUCB_DUFAIL... 08E83010
  +0078 DUCB_DUFAIL... 08E83010
  +007C DUCB_DUFAIL... 08E83010
  +0080 DUCB_DUFAIL... 08E83010
  +0084 DUCB_DUFAIL... 08E83010
  +0088 DUCB_DUFAIL... 08E83010
  +008C DUCB_DUFAIL... 08E83010
  +0090 DUCB_DUFAIL... 08E83010
  +0094 DUCB_DUFAIL... 08E83010
  +0098 DUCB_DUFAIL... 08E83010
  +009C DUCB_DUFAIL... 08E83010
  +00A0 DUCB_DUFAIL... 08E83010
  +00A4 DUCB_DUFAIL... 08E83010
  +00A8 DUCB_DUFAIL... 08E83010
  +00AC DUCB_DUFAIL... 08E83010
  +00B0 DUCB_DUFAIL... 08E83010
  +00B4 DUCB_DUFAIL... 08E83010
  +00B8 DUCB_DUFAIL... 08E83010
  +00BC DUCB_DUFAIL... 08E83010
  +00C0 DUCB_DUFAIL... 08E83010
  +00C4 DUCB_DUFAIL... 08E83010
  +00C8 DUCB_DUFAIL... 08E83010
  +00CC DUCB_DUFAIL... 08E83010
  +00D0 DUCB_DUFAIL... 08E83010
  +00D4 DUCB_DUFAIL... 08E83010
  +00D8 DUCB_DUFAIL... 08E83010
  +00DC DUCB_DUFAIL... 08E83010
  +00E0 DUCB_DUFAIL... 08E83010
  +00E4 DUCB_DUFAIL... 08E83010
  +00E8 DUCB_DUFAIL... 08E83010
  +00EC DUCB_DUFAIL... 08E83010
  +00F0 DUCB_DUFAIL... 08E83010
  +00F4 DUCB_DUFAIL... 08E83010
  +00F8 DUCB_DUFAIL... 08E83010
  +00FC DUCB_DUFAIL... 08E83010
  +0100 DUCB_DUFAIL... 08E83010
  +0104 DUCB_DUFAIL... 08E83010
  +0108 DUCB_DUFAIL... 08E83010
  +010C DUCB_DUFAIL... 08E83010
  +0110 DUCB_DUFAIL... 08E83010
  +0114 DUCB_DUFAIL... 08E83010
```

Dispatchable Unit Stack Area: DUSA0@1

```plaintext
EZBDUSA: 08D8D110
  +0000 DUSA_EYE......... DUSA
  +0004 DUSA_NEXTUSA... 08E8F010
  +0008 DUSA_DUCB...... 08D8D110
  +000C DUSA_DUCB...... 08D8D110
  +0010 DUSA_DUCB...... 08D8D110
  +0014 DUSA_DUCB...... 08D8D110
```

---

208  z/OS V1R11.0 Comm Svr: IP Diagnosis Guide
Chapter 6. IPCS subcommands 209
TCPIPCS FRCA

Use this subcommand to display information about the Fast Response Cache Accelerator (FRCA) connections or about cached objects.

Syntax

```
TCPIPCS FRCA
```

```
CONNECTIONS SUMMARY

variable_item

variable_list

TCB_address

TITIE

ASID

TCB_address

UWSE_address

Parameters

If no parameters are specified, only FRCA connections are summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

ASID Displays only the FRCA information for this address space. The address space ID is 1 - 4 hexadecimal digits in length. If the ASID value begins with digit a - f or A - F, prefix the value with a 0 so that the address is not interpreted as a symbol name or as a character string.

TCB_address Displays the FRCA connection with this address. An address is specified as 1-8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a-f or A-F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

UWSE_address Displays the FRCA shared cache with this address. An address is 1 - 8 hexadecimal digits in length. An IPCS symbol name can be specified for an address. If an address begins with digit a-f or A-F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.
UWSX_address
Displays the FRCA server connection with this address. An address is specified as 1-8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a - f or A - F, prefix the address with a 0 so that the address is not interpreted as a symbol name or as a character string.

jobname
Displays only the FRCA information for this job name. The job name can be a TCP/IP application name or a stack name. The job name contains 1-8 alphanumeric characters.

connection_id
Displays the FRCA information with this connection ID. An ID is specified as 1-8 hexadecimal digits.

In addition to the variable parameters described above, you can specify the following keyword parameters:

CONNECTIONS
Display only information for FRCA connections that use either a shared or an exclusive FRCA cache. CONNECTIONS is the default.

EXCLUSIVE
Display only information for connections that use an exclusive FRCA cache.

SHARED
Display only information for connections that use a shared FRCA cache.

OBJECTS
Display only information for FRCA cached objects.

ALL
Display information for all FRCA connections and cached objects.

SUMMARY
Displays the addresses of the control blocks and other data in tables. SUMMARY is the default.

DETAIL
In addition to the SUMMARY display, DETAIL also shows the contents of the control blocks.

TCP, TITLE, NOTITLE
See “Parameters” on page 195 for a description of these parameters.

Restrictions: Be aware of the following keyword restrictions:
• If you specify multiple keywords from the set {CONNECTIONS, EXCLUSIVE, SHARED, OBJECTS, ALL}, only the last one is used.
• If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

Sample output of the TCPIPCS FRCA subcommand
The following is sample output of the TCPIPCS FRCA subcommand:

1 'IPCSFRCA' 1 12:03:17 01/21/08
+ 0{0 {} tcpipcs frca(* all) }}
1 'IPCSFRCA' 2 12:03:17 01/21/08
+ 0TCPIPCS FRCA(* ALL)
Dataset: IPCS.MVS054.IPCSFRCA.DUMP
Title: 'IPCSFRCA'
-The address of the TSAB is: 15927000

0TSeb SI Procedure Version Tsdb Tsdx Asid TraceOpts Status
015927040 1 TCPCS V1R10 1590A000 1590A0CB 002A 9FFF767F 00000000 Active
0 1 defined TCP/IP(s) were found
0 1 active TCP/IP(s) were found
0 1 TCP/IP(s) for CS V1R10 found
0====================================================================================================
0Analysis of Tcp/Ip for TCPCS. Index: 1
0-FRCA Server Connections
0Uwsx@ Tcb@ Cache@ References Flags Uwse@
1642D728 7F0FD410 00000000 3 4080 16649368
16427118 7F108390 00000000 2 4080 16649368
1630C590 7F107F10 16436560 3 4000 00000000
0 Bucket# Uwco@ References CsmObj@ Length Key
754 16436860 1 15785290 2,976 /index.html
0FRCA Shared Cache
0Uwse@ Cache@ References Asid
16649368 1681FOF8 2 36
0 Bucket# Uwco@ References CsmObj@ Length Key
754 16443240 1 15783810 2,975 /index.html
871 164431A0 1 15785310 2,977 /index2.html
0FRCA Client Connections
0Uwse@ Tcb@ Server@ Object@ Flags
7F108A0A 7F108810 1642D728 00000000 10
0Analysis of Tcp/Ip for TCPCS completed
0{{{{ }}}}
0{{{{ }}}}
0{{{{ }}}}
0{{{{ **** END OF OUTPUT ***** }}}}

TCP/IPCS HASH

Use this subcommand to display information about the structure of TCP/IP hash tables.

Syntax

```
TCP/IPCS HASH
```

Parameters

**ALL**
Display structure of all TCP/IP hash tables. ALL is the default.
ICMPV6
Display only the structure of ICMPV6 hash tables.

IPSEC
Display only the structure of IPSecurity hash tables.

NETACC
Display only the structure of NetAccess hash tables.

POLICY
Display only the structure of Service Policy hash tables.

TCP
Display only the structure of TCP hash tables.

TTLS
Display only the structure of AT-TLS hash tables.

UDP
Display only the structure of UDP hash tables.

XCF
Display only the structure of XCF hash tables.

HEADER
Display hash table header information. Not displayed by default.

SUMMARY
Displays the addresses of the control blocks and other data in tables.
SUMMARY is the default.

DETAIL
In addition to the SUMMARY display, DETAIL also shows the search key values.

BOTH
Display both active and logically deleted table elements. BOTH is the default.

ACTIVE
Display only the active table elements.

DELETE
Display only the logically deleted table elements.

TCP, TITLE, NOTITLE
See “Parameters” on page 195 for a description of these parameters.

Restrictions: Be aware of the following keyword restrictions:
• If you specify multiple keywords from the set (ALL, ICMPV6, IPSEC, NETACC, POLICY, TCP, TTLS, UDP, XCF), all of them are used.
• If you specify multiple keywords from the set (BOTH, ACTIVE, DELETE), only the last one is used.
• If you specify multiple keywords from the set (SUMMARY, DETAIL), only the last one is used.

Sample output of the TCPIPCS HASH subcommand
The following is sample output of the TCPIPCS HASH subcommand.

TCPIPCS HASH ( DETAIL ALL )
Dataset: D74L_KWDEV03.DUMP
Title: ICMP HASHTAB
The address of the TSAB is: 09996F8
Tseb SI Procedure Version Tsdb Tsdx Asid TraceOpts Status
2 defined TCP/IP(s) were found
2 active TCP/IP(s) were found
2 TCP/IP(s) for CS VIR5 found

Analysis of Tcp/IP for TCPCS2. Index: 1
TCP/IP Hash Table Analysis
Policy ID Port Table
Hash Table Header at 7F65E008
  Instance : 1
  Active entries : 0
  Hash buckets : 1,999
  User free routine : 00000000
  Element queue : 08FE9E48
0 elements in Policy ID Port Table
Table Summary:
  Active buckets : 0
  Inactive buckets : 0
  Unused buckets : 1,999
  Max active q length : 0
  Max active q index : 0
  Max active q seqnum : 0
  Max delete q length : 0
  Max delete q index : 0
  Total seqnum : 0

ICMPV6 Table
Hash Table Header at 7F699C08
  Instance : 4
  Active entries : 7
  Hash buckets : 1,024
  User free routine : 00000000
  Element queue : 08FE9E50
  Bucket@ Bucket@ Element@ Status User@ KeyValue
  2 7F699C28 7F2FBE80 Active 0AA6BF8B FEC00000 00000000 00000000 00000000
     Clock Ticks...... 00000003
     Tokens........... 00
     Token Tenths..... 00
  5 7F699C58 7F2F9100 Active 0AA6BFA8 00000000 00000000 00000000 00000000
     Clock Ticks...... 00000000
     Tokens........... 00
     Token Tenths..... 00
  6 7F699C68 7F2F9080 Active 0AA6BF9B 00000000 00000000 00000000 00000000
     Clock Ticks...... 00000000
     Tokens........... 00
     Token Tenths..... 00
7 elements in ICMPV6 Table
Table Summary:
  Active buckets : 6
  Inactive buckets : 0
  Unused buckets : 1,018
  Max active q length : 2
  Max active q index : 6
  Max active q seqnum : 2
  Max delete q length : 0
  Max delete q index : 0
  Total seqnum : 7

TCP V4 Index Table
Hash Table Header at 7F528B88
  Instance : 2
  Active entries : 6
  Hash buckets : 62,533
  User free routine : 88D9523E
Table Summary:
- Active buckets: 6
- Inactive buckets: 1
- Unused buckets: 62,526
- Max active q length: 1
- Max active q index: 0
- Max active q seqnum: 1
- Max delete q length: 0
- Max delete q index: 0
- Total seqnum: 8

TCP V6 Index Table
Hash Table Header at 7F2FDB88
- Instance: 5
- Active entries: 2
- Hash buckets: 62,533
- User free routine: 88D9523E
- Element queue: 08FE9E50

Table Summary:
- Active buckets: 2
- Inactive buckets: 0
- Unused buckets: 62,531
- Max active q length: 1
- Max active q index: 0
- Max active q seqnum: 1
- Max delete q length: 0
- Max delete q index: 0
- Total seqnum: 2

UDP DMUX V4 Table
Hash Table Header at 7F403B88
- Instance: 3
- Active entries: 2
- Hash buckets: 62,533
- User free routine: 88DB0E3C
- Element queue: 08FE9E48

Table Summary:
- Active buckets: 2
- Inactive buckets: 0
- Unused buckets: 62,531
- Max active q length: 1
- Max active q index: 0
- Max active q seqnum: 1
- Max delete q length: 0
- Max delete q index: 0

Chapter 6. IPCS subcommands
TCPIPCS HEADER

Use the TCPIPCS HEADER command to display information from the system dump header and, in some cases, if a DUCB has ABENDed, the DUCB is displayed. The IPCS command STATUS System Cpu Registers Worksheet Faildata is used to display the system dump header.

Depending on the error recovery routine, the DUCB address might or might not be available. If the DUCB address is available, the DUCB is displayed. To find DUCBs that ABENDed, use the TCPIPCS DUAF (* ABEND) command.

Syntax

```
TCPIPCS HEADER TCP tcp_proc_name tcp_index [TITLE] [NOTITLE]
```
Parameters
TCP, TITLE, NOTITLE

See "Parameters" on page 195 for a description of these parameters.

Sample output of the TCPIPCS HEADER subcommand
The following is sample output of the TCPIPCS HEADER subcommand:

TCPIPCS HEADER
Dataset: IPCS.MV21381.DUMPA
Title: SLIP DUMP ID=TC

The address of the TSAB is: 13391BC0

<table>
<thead>
<tr>
<th>Tsdb</th>
<th>SI Procedure Version Tsdb</th>
<th>Tsdx</th>
<th>Asid</th>
<th>TraceOpts</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>13391C00</td>
<td>1 TCPSVT V2R10 12380000 123800C8 07DE 04041405</td>
<td>Active</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13391C00</td>
<td>2 TCPSVT2 V2R10 00000000 00000000 07E8 00000000</td>
<td>Down Stopping</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13391D00</td>
<td>3 TCPSVT3 V2R10 12FC3000 12FC30C8 0080 94FF755F</td>
<td>Active</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13391D00</td>
<td>4 TCPSVT3 V2R10 00000000 00000000 0059 00000000</td>
<td>Down Stopping</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found
4 TCP/IP(s) for CS V2R10 found

=================================================================================================================================

Analysis of Tcp/Ip for TCPSVT. Index: 1
STATUS SUBCOMMAND

MVS Diagnostic Worksheet

Dump Title: SLIP DUMP ID=TC

CPU Model 9672 Version AC Serial no. 041018 Address 00
Date: 03/22/2000 Time: 07:36:57.297123 Local

Original dump dataset: SYS1.DUMP93

Information at time of entry to SVCDUMP:
HASID 000B PASID 000B SASID 000B PSW 440C0000 81584B1C
CML ASCB address 00000000 Trace Table Control Header address 7F45D000

Dump ID: 007
Error ID: N/A

SDWA address N/A

....

CPU STATUS:
PSW=440C0000 81584B1C (RUNNING IN PRIMARY, KEY 0, AMODE 31, DAT ON)
DISABLED FOR I/O EXT
ASID(X'000B') 01584B1C. IEANUC09.IEAVEDS0+1C IN READ ONLY NUCLEUS
ASCB11 at FB0700, JOB(WLM), for the home ASID
ASXB11 at 7FDF00 and TCBI1 at 7FB440 for the home ASID
HOME ASID: 000B PRIMARY ASID: 000B SECONDARY ASID: 000B

GPR VALUES
0-3 00000001 0288E01C 00000C38 0000008
TCPIPCS HELP

Use this subcommand to display TCPIPCS usage and syntax information.

Syntax

/SM590000/SM590000

TCPIPCS HELP

Parameters
If no parameters are specified, the function, operand, and syntax information are displayed for all TCPIPCS commands.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item
Any one of the TCPIPCS subcommand names.

In addition to the variable parameters described above, you can specify the following keyword parameters:

ALL
Display information for all TCPIPCS commands. ALL is the default.

FUNCTION
Display only function information.

OPERANDS
Display only operand information.

SYNTAX
Display only syntax information.

Restriction: If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them are used.

Sample output of the TCPIPCS HELP subcommand
The following is sample output of the TCPIPCS HELP subcommand:
tcipcs help (config function)

Function:

The TCPIPCS command displays selected information about a specific TCP/IP address space.

CONFIG - Produce device configuration report.

Function:

Display information about device, physical, and logical interfaces
TCPIPCS CONFIG(<{SUMMARY|DETAIL}>)

Operands:

SUMMARY - Display summary report.

DETAIL - Display summary and interface cross-reference reports.

***

TCPIPCS IPSEC

Use this subcommand to display information about IP security filters or tunnels, IP security translated ports, or defensive filters.

Syntax

TCPIPCS IPSEC

(parameters)

Variable_item

variable_list

filter_address

Displays the IP security filter or defensive filter that has this address. An address is specified as 1 - 8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with a - f or A - F, prefix the address with a 0 so that the address is not interpreted as a symbol name or as a character string.

tunnel_address

Displays the IP security tunnel that has this address. An address is specified as 1 - 8 hexadecimal digits. An IPCS symbol name can be
specified for an address. If an address begins with a-f or A-F, prefix the address with a 0 so that the address is not interpreted as a symbol name or as a character string.

**source_IP_address**

Displays the IPSecurity NAT SourceIP table entry with this address. An address is specified as 1-8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a-f or A-F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

**translated_port_address**

Displays the IPSecurity NAT Port Translation table entry with this address. An address is specified as 1-8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a-f or A-F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

In addition to the variable parameters previously described, you can specify the following keyword parameters:

**ALL**

Display information for IP security filters, tunnels, NAT traversal remote port translations, and defensive filters. ALL is the default.

**FILTERS**

Display only information for IP security filters.

**TUNNELS**

Display only information for IP security tunnels.

**XLPORTS**

Display only information for IP security NAT-translated ports.

**DFILTERS**

Display only information for defensive filters.

**SUMMARY**

Displays the addresses of the control blocks and other data in tables. SUMMARY is the default.

**DETAIL**

In addition to the SUMMARY display, DETAIL also shows the contents of the control blocks.

**TCP, TITLE, NOTITLE**

See “Parameters” on page 195 for a description of these parameters.

Tips:

- If you specify multiple keywords from the set [ALL, FILTERS, TUNNELS, XLPORTS, DFILTERS], only the last one is used.
- If you specify multiple keywords from the set [SUMMARY, DETAIL], only the last one is used.

**Restriction:** The TCPIPCS IPSEC subcommand works only on stacks configured for IP security.

**Sample output of the TCPIPCS IPSEC subcommand**

The following is sample output of the TCPIPCS IPSEC subcommand:
TCPIP Ipsecurity Analysis

IPSEC on ZIIP No

FWE_GDA at 7F283430
FILTER_DA at 7F172C10
DEFENSIVE_FILTER_DA at 7F172BB0
TUNNEL_DA at 7F172530
ENCYP_DA at 7F282890

Filter set active : Policy
Filter logging : No
Pre-decap filtering : No

Defense Filter Mode : Active
IPv4 Defensive filter inbound list: 00000000
IPv4 Defensive filter inbound count: 0
IPv4 Defensive filter outbound list: 00000000
IPv4 Defensive filter outbound count: 0
IPv6 Defensive filter inbound list: 00000000
IPv6 Defensive filter inbound count: 0
IPv6 Defensive filter outbound list: 00000000
IPv6 Defensive filter outbound count: 0

IPv4 Filters
Filter@ Action SPrt1 SPrt2 DPrt1 DPrt2 Protocol
Src@ Dst@
7C553110 Permit 500 0 500 0 17 (UDP) 0.0.0.0/0 0.0.0.0/0
7BB9D090 Permit 0 0 623 0 6 (TCP) 197.11.107.1 197.11.236.12

IPv6 Filters
Filter@ Action SPrt1 SPrt2 DPrt1 DPrt2 Protocol
Src@ Dst@
7BB9D1610 Permit 500 0 500 0 17 (UDP) ::0/0 ::0/0
7DA22D90 Permit 623 0 0 0 6 (TCP) 2000:197:11:235::101:0:1 2000:197:11:107::1

IPv4 Tunnels
Tunnel@ Policy Format Name
Src@ Dst@
7C514010 000000A5 00000033 Y 1589 DVA-linux 197.11.235.9 16.11.16.126

IPv6 Tunnels
Tunnel@ Policy Format Name
Src@ Dst@
7E5AF010 0000014A 0000000C M 1 IPMVAospfAH03 ::0 ::0

Figure 25. TCPIPCS IPSEC subcommand sample output
TCPIPCS LOCK

Use this subcommand to scan the dump for information about the current locks that are defined and held.

Only nonzero statistics are reported.

Tip: The DUCB lock table entries might conflict with the lockword counters. This is because DUCB lock table entries and lockword counters are not updated in one operation, therefore they can be out of sync. At the time the dump was obtained, the lockword counters might have been updated, but the DUCB has not yet been updated.

Syntax

```
TCPIPCS LOCK [SUMMARY] [DETAIL] [TCP (tcp_proc_name)] [TCP_INDEX] [TITLE] [NOTITLE]
```

Parameters

SUMMARY
Displays each level of each class of lock, the total number of DUCBs found, and a cross-reference for each lock being used. SUMMARY is the default.

DETAIL
In addition to the SUMMARY display, DETAIL also shows lock information for each DUCB.

TCP, TITLE, NOTITLE
See “Parameters” on page 195 for a description of these parameters.

Restriction: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

Sample output of the TCPIPCS LOCK subcommand

The following is sample output of the TCPIPCS LOCK subcommand:

```
TCPIPCS LOCK (DETAIL)
Dataset: IPCS.A594094.DUMPM
Title: TCPSVT V3R10: Job(TCPSVT ) EZBITSTO(HTCP50A 99.281)+00077A S4C5/74BE2500 SRB P=0051,S=0051,H=0051

ItCvt: 12B573C8, Class_Count: 12, Level_Count: 34, Table_Size: 616

Lock statistics at 12E7B208

Class 2 at 12E7B2E8 for 2 levels
Level 0201 ITSTOR_QUE
Suspension - Srb : 1,601
Delays - : 239

Class 6 at 12E7B478 for 4 levels
Level 0602 TCB
Suspension - Srb : 146
Suspension - Tcb : 33

Ix Dchb Lktb Susp Next Ducbi Status
```
0002 12A62000 12A62184 00000000 00000000 10000001 Iu
Lock Class 02: 00000001 00000002 12A62278 00000000
Lock Level 01: 12B57CB8 C0010201 00010000 Held Excl ITSTOR_QUE
Ix Ducb@ Lktb@ Susp@ Next@ DucbIx Status
072E 12B19000 12B19184 00000000 7FFAFAF1 1000003E Iu
Lock Class 06: 00000002 00000004 12B192F0 00000000
Lock Level 02: 7F272D38 80010602 00020100 Held Shr TCB

50 DUCBs found
2 DUCBs held locks
0 DUCBs were waiting for locks

Lockword Cross Reference
Lock Ducb@ Status Name
12B57CB8 Not Held ITSTOR_QUE
7F272D38 12B19000 Held Shr TCB

2 locks were referenced

Lock Class/Level Multiple Usage:

Class Level Names
 03 02 REASM
          PTREE
          MGRP
...
 0C 06 SKITSSL
         TCF6_CLEANUP

Analysis of Tcp/IP for TCPSVT completed

TCP/IPCS MAP

Use this subcommand to display a mapping of TCP/IP storage. This subcommand is useful for finding overlays and abandoned storage.

Each control block referenced is listed in order by its address. Each control block eye-catcher is shown; if none is found, a mnemonic name is given in quotation marks. The size is the number of bytes (in decimal) in the control block. The key is the storage key. The base and offset are the address of a TCP/IP control block and the offset within it that contains the CbAddr in the far left column. Multiple references can exist, so additional references are continued on a separate line.

Tip: Large dumps with many control blocks can take considerable time to process.

Syntax
Parameters

ALL
Display storage usage information for all components.

CACHE
Display only CACHE storage usage information.

DUCB
Display only DUCB storage usage information.

ICMP
Display only ICMP storage usage information.

IF
Display only IF/IP storage usage information.

IP
Display only IF/IP storage usage information.

IPSEC
Display only IPSEC storage usage information.

NETACC
Display only NETACC storage usage information.

POLICY
Display only POLICY storage usage information.

RAW
Display only RAW storage usage information.

SOCKETS
Display only SOCKETS storage usage information.

STREAMS
Display only STREAMS storage usage information.

TCP
Display only TCP storage usage information.
**TELNET**
Display only TELNET storage usage information.

**TIMERS**
Display only TIMERS storage usage information.

**TTLS**
Only display AT-TLS storage usage information.

**UDP**
Display only UDP storage usage information.

**XCF**
Display only XCF storage usage information.

**TCP, TITLE, NOTITLE**
See "Parameters" on page 195 for a description of these parameters.

**Restriction:** If you specify multiple keywords from the set (ALL, CACHE, DUCB, ICMP, IF, IP, IPSEC, NETACC, POLICY, RAW, SOCKETS, STREAMS, TCP, TELNET, TIMERS, TTLS, UDP, XCF), all of them are used.

**Sample output of the TCPIPCS MAP subcommand**

The following is sample output of the TCPIPCS MAP subcommand:

```
TCPIPCS MAP
Dataset: IPCS.MV20767.DUMPA
Title: VERIFY MV20758

The address of the TSAB is: 08DD36F8
```

```

Tseb SI Procedure Version Tsdb Tsdx Asid TraceOpts Status
08DD3738 1 TCPCS V2R10 0876E000 0876E0C8 01F7 92208100 Active

1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
1 TCP/IP(s) for CS V2R10 found
```

Analysis of Tcp/Ip for TCPCS. Index: 1

CbIds enclosed in quotes e.g. "CBID" are not true eyecatchers.

Found 847 References and 1037 Cross-references

```
CbAddr CbId Size Key Base +Offset
00FCC6A0 CVT 1,280 6
01663450 ECVT 576 6 00FCC6A0+0004
0876B488 "ALCCSA" 96 6 0876E5C8+0560
0876B600 "CACSA " 120 6 0876E5C8+0218
0876B688 "CACSMM" 120 6 0876E5C8+0568
0876D700 "CACSA " 120 6 0876B600+000C
0876E8B8 "CACSA " 120 6 0876B600+000C
0876F8B8 "CACSA " 120 6 0876E8B8+0004
...```

`7F6E8B78 SKQU 64 6 7F6E8748+00E8
7F6E8BB8 SKQU 64 6 7F6E8A8A+0004`
Analysis of Tcp/IP for TCPCS completed

TCPIPCS MTABLE

Use this subcommand to access the module tables and display the following:

- Module entry point address
- Name
- Compile date and time
- PTF number
- Load module name

The entries are listed first in entry-point-address order, and then listed again in module-name order.

Syntax

```
TCP/IPCS MTABLE
```

Parameters

If no parameters are specified, all displayable modules are displayed.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

address

Locates the TCP/IP module where this address appears and displays the name and offset. An address is specified as 1–8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.
name  Locates the TCP/IP module with this name. A name is specified as 1-8 characters.

In addition to the variable parameters previously described, you can specify the following keyword parameters:

- **BOTH**  Display modules sorted by address and by name.
- **BYADDR**  Display only modules sorted by address.
- **BYNAME**  Display only modules sorted by name.

**TCP, TITLE, NOTITLE**  
See “Parameters” on page 195 for a description of these parameters.

**Sample output of the TCPIPCS MTABLE subcommand**

The following is a sample output of the TCPIPCS MTABLE subcommand:

```
TCPIPCS MTABLE (12DE3800 12D9B858)
Dataset: IPCS.A594094.DUMMP
Title: TCPSVT V2R10: Job(TCPSVT) EZBITSTO(HTCP50A 99.281)+
       00077A 54C5/74BE2500 SRB P=0051,S=0051,H=0051

The address of the TSAB is: 12EB9BB8

Tsee  SI Procedure Version Tsdb  Tsdx  Asid  Traceopts Status
12EB9BF8  1 TCPSVT  V2R10  12B57000  12B570C8  0051  9FFFF7F  Active

1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found

1 TCP/IP(s) for CS  V2R10 found
```

Analysis of Tcp/IP for TCPSVT. Index: 1

**TCPIPCS POLICY**

Use this subcommand to display policy information.

**Syntax**
Parameters

**SUMMARY**
Displays the policy table addresses. SUMMARY is the default.

**DETAIL**
In addition to the SUMMARY display, DETAIL also shows control block contents.

**TCP, TITLE, NOTITLE**
See "Parameters" on page 195 for a description of these parameters.

**Restriction:** If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

Sample output of the TCPIPCS POLICY subcommand
The following is sample output of the TCPIPCS POLICY subcommand:

TCPIPCS POLICY TCP(1)
Dataset: IPCS.MV21046.DUMPA
Title: BOTSWANA HUNG RUNNING PAGENT DIFFSERV SETTINGS.

The address of the TSAB is: 12EFD010

<table>
<thead>
<tr>
<th>Tseb</th>
<th>SI Procedure</th>
<th>Version</th>
<th>Tsd</th>
<th>Tsdx</th>
<th>Asid</th>
<th>TraceOpts</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>12EFD05B</td>
<td>1 TCPSVT</td>
<td>V2R10</td>
<td>12EB0800</td>
<td>12EB08CB</td>
<td>0058</td>
<td>9CFF755F</td>
<td>Active</td>
</tr>
<tr>
<td>12EFD08B</td>
<td>2 TCPSVT1</td>
<td>V2R10</td>
<td>12A0F000</td>
<td>12A0F0CB</td>
<td>0069</td>
<td>9CFF755F</td>
<td>Active</td>
</tr>
<tr>
<td>12EFD995B</td>
<td>3 TCPSVT2</td>
<td>V2R10</td>
<td>127C9000</td>
<td>127C90CB</td>
<td>070E</td>
<td>9CFF755F</td>
<td>Active</td>
</tr>
<tr>
<td>12EFD90B</td>
<td>4 TCPSVT3</td>
<td>V2R10</td>
<td>126F8000</td>
<td>126F80CB</td>
<td>0054</td>
<td>9CFF755F</td>
<td>Active</td>
</tr>
<tr>
<td>12EFDAD0B</td>
<td>5 TCPSVT4</td>
<td>V2R10</td>
<td>12646000</td>
<td>126460CB</td>
<td>004C</td>
<td>9CFF755F</td>
<td>Active</td>
</tr>
<tr>
<td>12EFD0B8</td>
<td>6 TCPSVT5</td>
<td>V2R10</td>
<td>1260E000</td>
<td>1260E0CB</td>
<td>070D</td>
<td>9CFF755F</td>
<td>Active</td>
</tr>
<tr>
<td>12EFD85B</td>
<td>7 TCPSVT6</td>
<td>V2R10</td>
<td>12383000</td>
<td>123830CB</td>
<td>007A</td>
<td>9CFF755F</td>
<td>Active</td>
</tr>
<tr>
<td>12EFD0D8</td>
<td>8 TCPSVT7</td>
<td>V2R10</td>
<td>11ECE000</td>
<td>11ECE0CB</td>
<td>07DC</td>
<td>9CFF755F</td>
<td>Active</td>
</tr>
</tbody>
</table>

8 defined TCP/IP(s) were found
8 active TCP/IP(s) were found
8 TCP/IP(s) for CS V2R10 found

Analysis of Tcp/Ip for TCPSVT. Index: 1

Policy Control Table at 12F54210
Intrusion Detection Main Table at 13AA6088
Service Classes:

Scentry@ Scope Tos Pri Permission Name
129455F0 Both 60 00 Allowed paPRO-GenImp5
129454F0 Both 00 00 Allowed padefault
TCPIPCS PROFILE

Use this subcommand to show the active configuration information at the time of the dump, in the form of profile data set statements. This profile does not necessarily match the profile used to start TCP/IP because the startup profile might not include the dynamic changes, additions, or deletions made using commands. All the defaults that are in effect are displayed in addition to explicit settings.

Syntax

```
TCPIPCS PROFILE
```

Parameters

ALL
   Display all profile statements.

TCPIP
   Display only TCP/IP profile statements.

TELNET
   Display only Telnet profile statements.

TCP, TITLE, NOTITLE
   See “Parameters” on page 195 for a description of these parameters.

Sample output of the TCPIPCS PROFILE subcommand

The following is sample output of the TCPIPCS PROFILE subcommand:

TCPIPCS PROFILE
Dataset: IPCS.PKI2345.V1R10.TCPIPCS.DUMP
Title: V1R10 TCPIPCS PROFILE SAMPLE IPV6 CINET
The address of the TSAB is: 162E9000
Tseb SI Procedure Version Tsdb Tsdx Asid TraceOpts Status
162E9040 1 TCPCS5 V1R10 15951000 159510C8 0032 9FF777F 00000000 Active
162E90C0  2 TCP/IP(s) were found
2 active TCP/IP(s) were found
2 TCP/IP(s) for CS V1R10 found

================================================================================
Analysis of Tcp/Ip for TCPCS5. Index: 1
;
; Profile generated on 2007/12/12 at 19:51:29
;
; Dump Dataset : IPCS.PK12345.VIR10.TCPIPCS.DUMP
; Dump Time : 2007/12/12 14:46:15.672018
; TCP/IP Jobname: TCPCS5
;
; For informational purposes, only BEGINRoutes
; and SMFCONFIG will be generated in this
; reconstructed profile.
;
; Either a GATEWAY or a BEGINRoutes statement is
; specified in an initial profile data set, or
; in a data set referenced by the
; VARY TCPIP,,OBEYFILE command.
; BEGINRoutes is the recommended way to define
; static routes.
;
; Either an SMFCONFIG or an SMFPARMS statement
; is specified in an initial profile data set,
; or in a data set referenced by the
; VARY TCPIP,,OBEYFILE command.
; SMFCONFIG is the recommended way to define
; SMF processing options.
;
ARPAGE 20
AUTOLog 5
TRMD PARMSTRING "D=TCPDATA5"
ENDAUTOLog
DEVice VIPA4815 VIRTual 0000
LINK VIPA4815L VIRTual 0 VIPA4815
DEVice IUTSAMEH MPCPTP NOAUTORESTART
LINK TOVTAM MPCPTP IUTSAMEH IFSPEED 4500000 CHECKSUM SECLASS 255
NOMONSYSPLEX
DEVice MPC4115 MPCPTP NOAUTORESTART
LINK MPC4115L MPCPTP MPC4115 IFSPEED 4500000 CHECKSUM SECLASS 255
NOMONSYSPLEX
DEVice MPC4145 MPCPTP NOAUTORESTART
LINK MPC4145L MPCPTP MPC4145 IFSPEED 4500000 CHECKSUM SECLASS 255
NOMONSYSPLEX
DEVice MPC4185 MPCPTP NOAUTORESTART
LINK MPC4185L MPCPTP MPC4185 IFSPEED 4500000 CHECKSUM SECLASS 255
NOMONSYSPLEX
DEVice QDIO4105 MPCIPA PRIROUTER NOAUTORESTART
LINK QDIO4105L IPAQENET QDIO4105 IFSPEED 100000000 READSTORAGE GLOBAL
INBPREF BALANCED SECLASS 255 NOMONSYSPLEX
GLOBALConfig NOTCPIPStatistics ECSALIMIT OK POOLLIMIT OK
NOMLSCHKTTERminate SEGMENTATIONOFFload
NOEXPLICITBINDPORTRange SYSPLEXMonitor TIMERSECS 60
NORECOVERY NODELAYJOIN NOAUTOREJOIN NOMONINTERFACE
NODYNROUTE SYSPLEXWLMpo1 60 ZIIP NOIPSECURITY NOIQDIOMULTIWRITE
MAXRECS 100 NOIQDMULTIWRITE WLPriorityQ 10Pri1 0
10Pri1 1 10Pri3 2 3 10Pri4 4 5 6 Fwd

NETMonitor NONATRCService NOPKTRCSerivce NOTCPCONNService SMFService

HOME
10.81.5.5 VIPA4815L
10.51.0.5 TOVTAM
10.11.1.5 MPC4115L

| 230  z/OS VIR11.0 Comm Svr: IP Diagnosis Guide
Chapter 6. IPCS subcommands

10.11.4.5 MPC4145L
10.11.8.5 MPC4185L
172.16.1.5 QDIO4105L

<table>
<thead>
<tr>
<th>10.11.4.5 MPC4145L</th>
<th>10.11.8.5 MPC4185L</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.11.4.5 MPC4145L</td>
<td>10.11.8.5 MPC4185L</td>
</tr>
</tbody>
</table>

INTERFace VIPA6815 DEFINE VIRTUAL6 IPADDR 2001::DB8:10::1::0:5

INTERface IUTSAMEH6 DEFINE MPCPTP6 TRLE IUTSAMEH INTFID ::51:0:5
SECLASS 255 NOMONSYSPLEX IPADDR 2001::DB8:10::51:0:5

INTERFace MPC6115 DEFINE MPCPTP6 TRLE MPC4115 INTFID ::11:1:5
SOURCEVIPAIrface VIPA6815 SECLASS 255 NOMONSYSPLEX IPADDR

INTERFace MPC6245 DEFINE MPCPTP6 TRLE MPC4245 INTFID ::12:4:5
SOURCEVIPAIrface VIPA6815 SECLASS 255 NOMONSYSPLEX IPADDR

INTERFace MPC6185 DEFINE MPCPTP6 TRLE MPC4185 INTFID ::11:8:5
SOURCEVIPAIrface VIPA6815 SECLASS 255 NOMONSYSPLEX IPADDR

INTERFace VIPA6815 DEFINE VIRTUAL6 IPADDR 2001::DB8:10::11:1:5

INTERFace QDIO6105 DEFINE IPAQENET6 PORTNAME QDIO4105 PRIROUTER
DUPADDNRE 1 SOURCEVIPAIrface VIPA6815 NODYNVLANREG
READSTORAGE GLOBAL INBPERF BALANCED INTFID ::16:1:5 SECLASS
255 NOMONSYSPLEX TEMPPREFIX ALL IPADDR 2001::DB8:172::16:1:5 NOISOLATE NOOLM

IPCONFig ARPTO 1200 DATAGRamfwd NOFWDMULTipath DVFRETRYDuraion 4
IPSECurity NOSOURCEVIIPA NOTCPSTACKSOURCEVIIPA NOSYSPLEXRouting
REASSEMBLYtimeout 60 TTL 64 NOPATHMTUDiscovery NOMUXIPPATH
NODYNAMICXCF NODYNOIPRouting FORMAT LONG NODYNAMICXCF NOQDIOACCELERATOR

IPCONFI66 DATAGRamfwd NOFWDMULTipath IPSECurity NOSOURCEVIIPA
NOTCPSTACKSOURCEVIIPA NOMULTIPATH HOPLimit 255 ICMPERRORlimit
3 NOIGNOREROUTERHOPLIMIT NODYNAMICXCF NOTEMPADDRS

ENIPSEC

ITRACE OFF AUTODAEMON
ITRACE OFF COMMAND
ITRACE OFF CONFIG
ITRACE OFF SUBAGENT

PORT 3020 TCP DCICSTS DELAYAcks
PORT 3019 TCP DCICSTS DELAYAcks
PORT 3018 TCP DCICSTS DELAYAcks
PORT 3017 TCP DCICSTS DELAYAcks
PORT 3016 TCP DCICSTS DELAYAcks
PORT 3015 TCP DCICSTS DELAYAcks
PORT 3014 TCP DCICSTS DELAYAcks
PORT 3013 TCP DCICSTS DELAYAcks
PORT 3012 TCP DCICSTS DELAYAcks
PORT 3011 TCP DCICSTS DELAYAcks
PORT 3010 TCP DCICSTS DELAYAcks
PORT 3009 TCP DCICSTS DELAYAcks
PORT 3008 TCP DCICSTS DELAYAcks
PORT 3007 TCP DCICSTS DELAYAcks
PORT 3006 TCP DCICSTS DELAYAcks
PORT 3005 TCP DCICSTS DELAYAcks
PORT 3004 TCP DCICSTS DELAYAcks
PORT 3003 TCP DCICSTS DELAYAcks
PORT 3002 TCP DCICSTS DELAYAcks
PORT 3001 TCP DCICSTS DELAYAcks
PORT 3000 TCP DCICSTS DELAYAcks
PORT 520 UDP * DELAYAcks
PORT 161 UDP OSNMPD DELAYAcks
PORT 53 UDP OMVS DELAYAcks
PORT 53 TCP OMVS DELAYAcks
PORT 23 TCP TCPCS5 DELAYAcks
PORT 21 TCP OMVS DELAYAcks
PORT 19 TCP MISCserv DELAYAcks
PORT 19 UDP MISCserv DELAYAcks
PORT 9 TCP MISCserv DELAYAcks
PORT 9 UDP MISCserv DELAYAcks
PORT 7 TCP MISCserv DELAYAcks
PORT 7 UDP MISCserv DELAYAcks
SACONfig COMMUNITY public AGENT 161 SACACHeTime 30 ENABLed SETSDISAbled
OSADISabled
SMFCONfig TYPE118 NOTCPINIT NOTCPTERM NOFTPCLIENT NOTN3270CLIENT
NOTCPIPStatistics TYPE119 NOTCPINIT NOTCPTERM NOFTPCLIENT
NOTN3270CLIENT NOTCPIPStatistics NOIFStatistics
NOTPORTStatistics NOTCPSACK NOUDPterm NOIPSECURITY

| PROFILE
SOMAXCONN 10
START IUTSAMEH
START IUTSAMEH6
STOP MPC4115
STOP MPC6115
STOP MPC4145
STOP MPC4185
STOP MPC6185
STOP QDIO4105
STOP QDIO6105
STOP MPC6245
TCPCONFIG INTERVAL 120 DELAYAcks
TCPPROFERENCE 16384 TCPSENDBuffer size 16384 TCPMAXRCVBuffer size 262144 FINWAIT2TIME
SENDGarbage FALSE TCPTIMESTAMP NOTTLS
UDPconfig RESTRICTLowports UDPCHKsum UDPSENDBuffer size 65535
UDPPROFERENCE 65535 UDPQueueLimit
BEGINRoutes
ROUTE 10.81.4.0 255.255.255.0 10.11.5.4 MPC4145L MTU DEFAULTSIZE
MAXIMUMretransmittime 120 MINIMUMretransmittime 0.5 ROUNDTripGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2 DELAYAcks NOREPLACEable
ROUTE 10.81.2.0 255.255.255.0 10.11.5.4 MPC4145L MTU DEFAULTSIZE
MAXIMUMretransmittime 120 MINIMUMretransmittime 0.5 ROUNDTripGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2 DELAYAcks NOREPLACEable
ROUTE 10.51.0.4 HOST = TOVtAM MTU DEFAULTSIZE MAXIMUMretransmittime 120 MINIMUMretransmittime 0.5 ROUNDTripGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2 DELAYAcks NOREPLACEable
ROUTE 10.11.5.1 HOST = MPC4115L MTU 4096 MAXIMUMretransmittime 120
MINIMUMretransmittime 0.5 ROUNDTripGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2 DELAYAcks REPLACEable
ROUTE 10.11.5.4 HOST = MPC4145L MTU 4096 MAXIMUMretransmittime 120
MINIMUMretransmittime 0.5 ROUNDTripGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2 DELAYAcks REPLACEable
ROUTE 10.11.5.8 HOST = MPC4185L MTU 4096 MAXIMUMretransmittime 120
MINIMUMretransmittime 0.5 ROUNDTripGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2 DELAYAcks REPLACEable
ROUTE 10.81.8.0 255.255.255.0 10.11.5.8 MPC4185L MTU DEFAULTSIZE
MAXIMUMretransmittime 120 MINIMUMretransmittime 0.5 ROUNDTripGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2 DELAYAcks NOREPLACEable
ROUTE 172.16.0.0 255.255.255.0 = QDIO4105L MTU 1492
MAXIMUMretransmittime 120 MINIMUMretransmittime 0.5 ROUNDTripGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2 DELAYAcks REPLACEable
ROUTE 2001:DB8:10::10::14/128 = MPC6245 MTU DEFAULTSIZE
MAXIMUMretransmittime 120 MINIMUMretransmittime 0.5 ROUNDTripGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2 DELAYAcks NOREPLACEable
ROUTE 2001:DB8:10::11::4/128 = QDIO4105L MTU DEFAULTSIZE
MAXIMUMretransmittime 120 MINIMUMretransmittime 0.5 ROUNDTripGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2 DELAYAcks NOREPLACEable
ROUTE 2001:DB8:10::11::5/128 = MPC6115 MTU 4096 MAXIMUMretransmittime
120 MINIMUMretransmittime 0.5 ROUNDTripGain 0.125 VARIANCEGain
TCPIPCS PROTOCOL

Use this subcommand to display information from TCP, UDP, and RAW protocol control blocks.

Syntax

```
>>> TCPIPCS PROTOCOL
       (SUMMARY)
       (DETAIL)

TCP (tcp_proc_name tcp_index)

TITLE
NOTITLE
```

Parameters

SUMMARY
Formats the MTCB, MUDP, and MRCB contents. Lists all the TCBs, UCBs, and RCBs in separate cross-referenced tables. SUMMARY is the default.

DETAIL
In addition to the SUMMARY display, DETAIL formats the contents of the TCBs, UCBs, and RCBs.

TCP, TITLE, NOTITLE
See "Parameters" on page 195 for a description of these parameters.

Restriction: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

Sample output of the TCPIPCS PROTOCOL subcommand

The following is sample output of the TCPIPCS PROTOCOL subcommand:

TCPIPCS PROTOCOL
Dataset: IPCS.MV21381.DUMPA
Title: SLIP DUMP ID=TC

The address of the TSAB is: 13391BC0

Tseb SI Procedure Version Tsdb Tsdx Asid TraceOpts Status
13391C00 1 TCPVT V2R10 1323B000 1323B0C8 07DE 04041405 Active
13391C80  2 TCPSVT2  V2R10  00000000  00000000  07EB  00000000  Down Stopping
13391D00  3 TCPSVT1  V2R10  12FC3000  12FC30C8  0080  94FF755F  Active
13391D80  4 TCPSVT3  V2R10  00000000  00000000  0059  00000000  Down Stopping

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

4 TCP/IP(s) for CS V2R10 found

===============================================================================
Analysis of Tcp/IP for TCPSVT. Index: 1
TCPIP Raw Control Block Analysis
Master Raw Control Block (MRCB)
MRAWCB: 7F75B048
+0000  RMRCBEYE. MRCB  MRCMUTEX. 00000000  00000000  00000000
D7D60501  RSTKDWN. 00
+0021  RSTKLNKD. 01  RDRVSTAT. 01  RSBCAST.. 00000000
RSDNTRTE. 00000000  RSRCVBUF. 0000FFFF
+0030  RSSNDBUF. 0000FFFF  RDIPTOS.. 00  RDIPTTL.. 00
RIPWRQ.. 7F61D3E8  RIPRDQ.. 7F61D3A8
+0040  RHASH@... 7F75B08C
....

Raw Hash Table Entries
ID  First  Last
 9  7F5513C8  7F5513C8
15  7F712088  7F712088

RCB  ResrcID  ResrcNm  TpiState  DestAddr  ProtocolId
7F5513C8  00000062  OMPROUTE  WLOIDLE  129.11.208.108  89
7F712088  00000008  TCPSVT  WLOIDLE  0.0.0.0  255

2 RCB(s) FOUND
2 RCB(s) FORMATTED

TCP/IP Analysis
TCP/IP Main TCP Control Block (MTCB)
MTCB: 133BE350
+0000  M_MAIN_EYE.........  TCP MAIN
+0008  M_TCP_LWRITE_Q.....  7F781868
+000C  M_TCP_LREAD_Q......  7F781828
+0014  M_TCP_DRIVER_STATE. 01
+0018  M_TCPMTX.............  00000000  00000000  00000000  D7D60601
+0028  M_TCPAQMX............  00000000  00000000  00000000  D7D60604
+0038  MTCB_LIST_LOCK.......  00000000  00000000  00000000  D7D60604
+0048  M_PORT_CEILING.......  0000FFFF
+004C  M_TPI_SEQ#...........  0001C62B
+0050  M_PORT_ARRAY.........  7F712FC8
+0054  M_LAST_PORT_NUM....  00000445
....

TCB  ResrcID  ResrcNm  TcpState  TpiState  Flag1234  UseCount  IPAddr
Port LuName  AppiName  UserID
7F607100  00000002  TCPSVT  Closed  WLOUNBND  00000000  00000000  0.0.0.0
7F60A908  000083D7  FTPUNIX1  Listening  WLOIDLE  00200080  00000001  0.0.0.0
7F608D08  00000013  TCPSVT  Listening  WLOIDLE  00200080  00000001  0.0.0.0
TCPIPCS RAW

Use this subcommand to display the Master Raw Control Block (MRCB) and any Raw protocol Control Blocks (RCBs) defined in the MRCB hash table.

Syntax

TCPIPCS RAW

Chapter 6. IPCS subcommands 235
Parameters
If no parameters are specified, all raw connections are summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item
Any one of the following variable parameters.

variable_list
You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

Variable parameters are:

jobname
Displays only the RCB for this job name. The job name can be a TCP/IP application name or a stack name, and it must contain from 1-8 characters.

RCB_address
Displays only the RCB with this address. An address is specified as 1-8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

connection_id
Displays the RCB with this connection ID. A connection ID is specified as 1-8 hexadecimal digits.

In addition to the variable parameters described above, you can specify the following keyword parameters:

DATAQ
Formats RCBs which have data queued on the RECEIVE queue.

SUMMARY
Formats the MRCB contents and lists all the RCBs in one cross-reference table. SUMMARY is the default.

DETAIL
In addition to the SUMMARY display, DETAIL formats the contents of the RCBs.

TCP, TITLE, NOTITLE
See "Parameters" on page 195 for a description of these parameters.
Restriction: If you specify multiple keywords from the set [SUMMARY, DETAIL], only the last one is used.

Sample output of the TCPIPCS RAW subcommand
The following is sample output of the TCPIPCS RAW subcommand:

TCPIPCS RAW
Dataset: IPCS.R8A0723.RASDUMP
Title: EZRPE005
The address of the TSAB is: 098221F0
Tseb  SI Procedure Version Tsd  Tsdx  Asid Traceopts Status
09822230 1 TCPCS  VIR5  08E85000 08E850CB 001E 9FF7E7F Active
09822280 2 TCPCS2 VIR5  08937000 089370CB 01F6 9FF7E7F Active
  2 defined TCP/IP(s) were found
  2 active TCP/IP(s) were found
  2 TCP/IP(s) for CS VIR5 found

Analysis of Tcp/IP for TCPCS. Index: 1
TCPIPC Raw Control Block Analysis
Master Raw Control Block (MRCB)
MRrawCB: 7F407208
+0000 RMRCBHYE. MRCB RSTKLNKD. 01 RDRVSTAT. 01
+000C R6STKLNKD. 01
+001D R6DRVSTAT. 01
+0010 MRCMUTEX. 00000000 00000000 00000000 D7D60501
RSBCAST.. 00000000 RSDNTRTE. 00000000 RSRC
+002C RSSOBUF. 0000FFFF RDIPTOS.. 00 RDIPTTLS.. 00
RIPRQ@.. 7F621DA8 RIPPRQ@.. 7F621D68 RHAS
+0040 RIP6RQ@. 7F686468 RIP6RQ@. 7F686428 R6HASH@. 7F407374
+004C R6DFFLTR. 7F7B1IFF FFFFFF FFFFFF FFFFFF 003FFFFF
                      FFFFFF FFFFFF FFFFFF
IPv4 Raw Hash Table Entries
  ID  First   Last
   0  7F52C390 7F52C390
   15 7F52C110 7F52C110

IPv6 Raw Hash Table Entries
  ID  First   Last
   0  7F20738C 7F20738C

IPv4 RAW Connections
RCB  ResrcID  ResrcNm  TpiState  ProtocolId  DestAddr
  7F52C38B 00000006 TCPCS  WLOIDLE  0  0.0.0.0
  7F52C10B 00000008 TCPCS  WLOIDLE  255 0.0.0.0

IPv6 RAW Connections
RCB  ResrcID  ResrcNm  TpiState  ProtocolId  DestAddr
  7F20720B 0000000E TCPCS  WLOIDLE  0 ::0

3 RCB(s) FOUND
3 RCB(s) FORMATTED

Analysis of Tcp/IP for TCPCS completed

TCPIPCS ROUTE
Use this subcommand to display the routing control blocks. Each routing table
entry is formatted to display the:
  • Route control block address
  • Device name
  • Type
  • Protocol
  • Destination IP address
• Gateway IP address
• Physical interface control block address

Syntax

```
TCPIP=ROUTE

[IPV4 | IPV6 | IQDIO | PD | PR | QDIOACCEL | RSTAT]
```

Parameters

**ALL**
Display structure of all route table information (including all active and to-be-deleted policy-based routing tables). ALL is the default.

**IPV4**
All IPv4 search tree and update tree routes.

**IPV6**
All IPv6 search tree and update tree routes.

**IQDIO**
All QDIO Accelerator and HiperSockets ™ Accelerator search tree and update tree routes.

**PD**
All search tree and update tree routes for all policy-based routing tables that have been marked for deletion.

**PR**
All search tree and update tree routes for all policy-based routing tables. Also list configured routes that use interfaces that are not defined in the stack and list dynamic routing parameters for all policy-based routing tables.

**QDIOACCEL**
All QDIO Accelerator and HiperSockets Accelerator search tree and update tree routes.

**RSTAT**
All defined replaceable static routes are displayed without regard to whether or not they are currently being used in the active routing table.

**TCP, TITLE, NOTITLE**
See “Parameters” on page 195 for a description of these parameters.

**Restriction:** If you specify multiple keywords from the set {ALL, IPV4, IPV6, IQDIO, QDIOACCEL, RSTAT}, all of them are used.
Sample output of the TCPIPCS ROUTE subcommand

The following is sample output of the TCPIPCS ROUTE subcommand:

TCPIPCS ROUTE
Dataset: IPCS.P414001.PUBDUMPA
Title: DUMP OF TCP STACKS

The address of the TSAB is: 3A467000

Tseb SI Procedure Version Tsdb Tsdx Asid TraceOpts Status
3A467040 1 TCPSVT V1R9 3A449000 3A4490C8 006A 97BF749F C0000000 Active
3A4670C0 2 TCPSVT1 V1R9 397CD000 397CD0C8 007B 97BF749F C0000000 Active
3A467140 3 TCPSVT2 V1R9 38669000 386690C8 007C 97BF749F C0000000 Active

3 defined TCP/IP(s) were found
3 active TCP/IP(s) were found
3 TCP/IP(s) for CS V1R9 found

============================================================================
Analysis of Tcp/Ip for TCPSVT. Index: 1
TCPIP Route Analysis
IPv4 Replaceable Static Routes Configured
Rtioctl@ LinkName IP Addresses
7F453150 LOGETH2 Destination: 174.33.84.237
Gateway : 0.0.0.0
7F452BB0 LOGETH2 Destination: 197.0.0.0
Gateway : 174.33.84.237
IPv6 Replaceable Static Routes Configured
Rtioctl6@ InterfaceName IP Addresses
7F452750 LV6OGETH2 Destination: 2000:176:11:104::237
Gateway : ::0
IPv4 Routes in Search Table
Rte@ LinkName Type/State Protocol Pif@ IP Addresses
7DCABB50 LOGETH2 Host OSPF 7F850490 Destination: 202.77.232.1
Subnet Mask: 255.255.255.255
Gateway : 174.33.84.237
7DFC25D0 LOGETHB Host OSPF 7F850090 Destination: 202.77.230.1
Subnet Mask: 255.255.255.255
Gateway : 174.33.84.237
7DCABE90 LOGETH2 Host OSPF 7F850490 Destination: 202.77.230.1
Subnet Mask: 255.255.255.255
Gateway : 174.33.84.237
7DCABCF0 LOGETHB Host OSPF 7F850090 Destination: 202.77.230.1
Subnet Mask: 255.255.255.255
Gateway : 174.33.84.237
7F453350 LOGETH2 Host Configuration 7F850490 Destination: 197.0.0.0
Subnet Mask: 255.0.0.0
Gateway : 0.0.0.0
7F452E70 LOGETH2 Subnetwork Configuration 7F850490 Destination: 197.0.0.0
Subnet Mask: 255.0.0.0
Gateway : 174.33.84.237
IPv6 Routes in Search Table
Rte@ InterfaceName Type/State Protocol Pif@ IP Addresses
7DCA7730 LV6OGETH2 Host OSPF 7F454650 Destination: fec0:197:11:104::20
IPv4 Routes in Update Table

<table>
<thead>
<tr>
<th>Rte@</th>
<th>LinkName</th>
<th>Type/State</th>
<th>Protocol</th>
<th>Pif@</th>
<th>IP Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>7DCA850</td>
<td>LOGETH2</td>
<td>Host</td>
<td>OSPF</td>
<td>7F850490</td>
<td>Destination: 202.77.232.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Active</td>
<td></td>
<td></td>
<td>Subnet Mask: 255.255.255.255</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gateway : 174.33.84.237</td>
</tr>
<tr>
<td>7DFC250</td>
<td>LOGETH2</td>
<td>Host</td>
<td>OSPF</td>
<td>7F850090</td>
<td>Destination: 202.77.232.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Active</td>
<td></td>
<td></td>
<td>Subnet Mask: 255.255.255.255</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gateway : 174.33.84.237</td>
</tr>
<tr>
<td>7DCAE90</td>
<td>LOGETH2</td>
<td>Host</td>
<td>OSPF</td>
<td>7F850490</td>
<td>Destination: 202.77.230.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Active</td>
<td></td>
<td></td>
<td>Subnet Mask: 255.255.255.255</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gateway : 174.33.84.237</td>
</tr>
<tr>
<td>7F45350</td>
<td>LOGETH2</td>
<td>Host</td>
<td>Configuration</td>
<td>7F850490</td>
<td>Destination: 174.33.84.237</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Active</td>
<td></td>
<td></td>
<td>Subnet Mask: 255.255.255.255</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gateway : 0.0.0.0</td>
</tr>
<tr>
<td>7F45270</td>
<td>LOGETH2</td>
<td>Subnetwork</td>
<td>Configuration</td>
<td>7F850490</td>
<td>Destination: 197.0.0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Active</td>
<td></td>
<td></td>
<td>Subnet Mask: 255.0.0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gateway : 174.33.84.237</td>
</tr>
</tbody>
</table>

IPv6 Routes in Update Table

<table>
<thead>
<tr>
<th>Rte@</th>
<th>InterfaceName</th>
<th>Type/State</th>
<th>Protocol</th>
<th>Pif@</th>
<th>IP Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>7DA7730</td>
<td>LV60GETH2</td>
<td>Host</td>
<td>OSPF</td>
<td>7F454650</td>
<td>Destination: fec0:197:11:104::20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Active</td>
<td></td>
<td></td>
<td>Prefix : 124</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gateway : fe80::11:176:50:104</td>
</tr>
<tr>
<td>7EBF450</td>
<td>LV61UTIQD00</td>
<td>Host</td>
<td>Configuration</td>
<td>7F455250</td>
<td>Destination: fe80::2440:ff:f10c:2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Active</td>
<td></td>
<td></td>
<td>Prefix : 128</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gateway : ::0</td>
</tr>
<tr>
<td>7EBF280</td>
<td>LV61UTIQD02</td>
<td>Host</td>
<td>Configuration</td>
<td>7F4567F0</td>
<td>Destination: fe80::2440:2ff:f10c:42</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Active</td>
<td></td>
<td></td>
<td>Prefix : 128</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gateway : ::0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Active</td>
<td></td>
<td></td>
<td>Prefix : 128</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gateway : ::0</td>
</tr>
</tbody>
</table>

Analysis of Tcp/IP for TCPSVT completed

**TCPINPCS SOCKET**

Use this subcommand to display information from TCP/IP socket control blocks.

**Syntax**

```
TCPINPCS SOCKET
   \-\-TCPINPCS SOCKET \-SUMMARY \-DETAIL \-variable_list \-variable_item \(\) \(\)
```

240  z/OS V1R11.0 Comm Svr: IP Diagnosis Guide
Parameters
If no parameters are specified, all sockets are summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item
Any one of the following variable parameters.

variable_list
You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

SCB_address
Displays only the socket control block (SCB) with this address. An address is specified as 1–16 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

connection_id
Displays the SCB with this connection ID. A connection ID is specified as 1–8 hexadecimal digits.

In addition to the variable parameters described above, the following keyword parameters can be specified:

SUMMARY
Summarizes the sockets. SUMMARY is the default.

DETAIL
In addition to the SUMMARY display, DETAIL formats the contents of the SCBs.

TCP, TITLE, NOTITLE
See [Parameters” on page 195 for a description of these parameters.

Restriction: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

Sample output of the TCPIPCS SOCKET subcommand
The following is sample output of the TCPIPCS SOCKET subcommand:

TCPIPCS SOCKET
Dataset: IPCS.MV21381.DUMPA
Title: SLIP DUMP ID=TC

The address of the TSAB is: 13391BC0

<table>
<thead>
<tr>
<th>Tseb</th>
<th>SI Procedure Version Tsd</th>
<th>Tsdx</th>
<th>Asid TraceOpts Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>13391C00 1 TCPSVT V2R10</td>
<td>13238000 132380C8 07DE 04041405</td>
<td>Active</td>
<td></td>
</tr>
<tr>
<td>13391C80 2 TCPSVT2 V2R10</td>
<td>00000000 00000000 07E8 00000000</td>
<td>Down Stopping</td>
<td></td>
</tr>
<tr>
<td>13391D00 3 TCPSVT3 V2R10</td>
<td>12FC3000 12FC30C8 0080 94FF755F</td>
<td>Active</td>
<td></td>
</tr>
<tr>
<td>13391D80 4 TCPSVT4 V2R10</td>
<td>00000000 00000000 0059 00000000</td>
<td>Down Stopping</td>
<td></td>
</tr>
</tbody>
</table>
4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

4 TCP/IP(s) for CS   V2R10 found

Analysis of Tcp/Ip for TCPSVT. Index: 1

TCPIIP Socket Analysis

<table>
<thead>
<tr>
<th>SCB</th>
<th>CID</th>
<th>Protocol</th>
<th>SockOpt</th>
<th>ScbFlags</th>
<th>ResrcNm</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000112D42120</td>
<td>0000000B</td>
<td>RAW</td>
<td>000000 000280000 TCPSVT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000112D422A0</td>
<td>0000000B</td>
<td>UDP</td>
<td>000000 000280000 TCPSVT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000112D42420</td>
<td>0000000C</td>
<td>TCP</td>
<td>000200 00280000 TCPSVT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000112D425A0</td>
<td>0000000E</td>
<td>UDP</td>
<td>000000 000280000 TCPSVT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000112D42720</td>
<td>0000000F</td>
<td>TCP</td>
<td>000000 00280000 TCPSVT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000112D428A0</td>
<td>00000010</td>
<td>TCP</td>
<td>002000 00280000 TCPSVT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000112D42A20</td>
<td>00000067</td>
<td>TCP</td>
<td>080000 00280000 OMPROUTE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000112D42A00</td>
<td>00000012</td>
<td>TCP</td>
<td>004000 00280000 TCPSVT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000112D42D20</td>
<td>00000013</td>
<td>TCP</td>
<td>004000 00280000 TCPSVT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000112D42E00</td>
<td>00000014</td>
<td>UDP</td>
<td>000000 00280000 PORTMAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000112D43020</td>
<td>00000015</td>
<td>TCP</td>
<td>000000 00280000 PORTMAP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...  

79 Socket control blocks were found
79 Socket control blocks were formatted

Analysis of Tcp/Ip for TCPSVT completed

**TCPIPICS STATE**

Use this subcommand to provide an overall view of TCP/IP. The following are displayed:
- Major control block addresses
- Subtasks
- Storage usage
- Dispatchable units
- Trace
- Configuration

**Syntax**

```
TCPIPICS--STATE
    TCP((tcp_proc_name))
    ALL
    CONFIG
    DUCB
    SNMP
    TRACE
```
**Parameters**

**ALL**
Display all state information. **ALL** is the default.

**CONFIG**
Display only configuration state information.

**DUCB**
Display only DUCB state information.

**SNMP**
Display only SNMP and CONFIG information. (SNMP information makes sense only in the context of the configuration, so the configuration information is also displayed.)

**TRACE**
Display only trace state information.

**TCP, TITLE, NOTITLE**
See "Parameters" on page 195 for a description of these parameters.

**Restriction:** If you specify multiple keywords from the set {ALL, CONFIG, DUCB, SNMP, TRACE}, all of them are used.

**Sample output of the TCPIPCS STATE subcommand**

The following is sample output of the TCPIPCS STATE subcommand:

```
Dataset: IPCS.X370012.TRAK0006.STEP2B.DUMP
Title: TRAK0006 STEP1
The address of the TSAB is: 15137000
Tseb SI Procedure Version Tsdb Tsdx Asid TraceOpts Status
15137040 1 TCPCS V1R8 14F5A000 14F5A0C8 0030 9FFF767F 00000000 Active
1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
1 TCP/IP(s) for CS V1R8 found
```

Analysis of Tcp/Ip for TCPCS. Index: 1

**TCPIP State**

**TCPIP Status:**

- Procedure: TCPCS
- Version: V1R8
- Status: Active
- Asid: 0030
- Started: 2005/08/24 16:21:15
- Ended: 2005/08/24 16:38:29
- Active: 00:17:13.628105 hours

**Major Control Blocks**
TCPII Subtasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Tcb FirstRB</th>
<th>EotECB StopEcb</th>
<th>CmpCode RsnCode RTWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBTCPIP</td>
<td>008FF2A0</td>
<td>008FF218</td>
<td>808FD2D0 00000000 00000000</td>
</tr>
<tr>
<td>EPWPITSK</td>
<td>008E4E88</td>
<td>008DC088</td>
<td>00000000 00000000 00000000</td>
</tr>
<tr>
<td>EZBITUB</td>
<td>008E4C0F</td>
<td>008E8340</td>
<td>00000000 808DC198 00000000</td>
</tr>
<tr>
<td>EZACDMMSM</td>
<td>008DE088</td>
<td>008DD4A8</td>
<td>00000000 00000000 00000000</td>
</tr>
<tr>
<td>EZBIPSUB</td>
<td>008E4B58</td>
<td>008DC110</td>
<td>00000000 808DC110 00000000</td>
</tr>
<tr>
<td>EZBIEDER</td>
<td>008E4D08</td>
<td>008E4710</td>
<td>00000000 808E4710 904C4000 00000004 00000000</td>
</tr>
<tr>
<td>EZBYLMST</td>
<td>008E4578</td>
<td>008E4100</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>EZACFMNN</td>
<td>008E4240</td>
<td>008E4100</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>EZBTZMST</td>
<td>008E4B70</td>
<td>008E4038</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>EZBTSSL</td>
<td>008DE0D0</td>
<td>008DE088</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>EZBTMCTL</td>
<td>008DEA78</td>
<td>008DE4D0</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>EZACFALG</td>
<td>008DE8E0</td>
<td>008DE858</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
<tr>
<td>EZASASUB</td>
<td>008DE987</td>
<td>008DE4038</td>
<td>00000000 00000000 00000000 00000000</td>
</tr>
</tbody>
</table>

Storage Cache Information

- Total CSA Allocated: 7,705,448
- Tcp/ip CSA Limit: 2,147M
- Total CSA Elements: 59
- Cache Delay: 210 seconds
- Scan Delay: 75 seconds
- Total cache allocated: 91,688
- Total cache elements: 8
- Total freed elements: 0
- Last cache scan time: 2005/08/24 20:37:12

CSM Status

- ECSA Storage: OK
- Data Space Storage: OK
- Fixed Storage: OK
- Alet: 01FF0014 Dspname: CSM64001
- Alet: 00000000 Dspname: ........

Dispatchable Unit Status

- DUCB Initializations: 11,741
- DUCB Expansions: 769
- Percent DUCB expansions: 6%
- Last DUCB scan time: 2005/08/24 20:34:24

- 1 DUAT control block(s) were found in the DUAF at 14F56018
- 124 Dispatchable units were found.
- No DUs indicate abend.

CTrace Status:

- Member Name: CTIEZBN0
- Buffer Size: 4,194,304
- Options: Init Opqmds Opqmsg Sockt AFP XCF Access PFS
- API Engine Queue RAW UDP TCP ICMP ND CLAW
- LCS Internet Message WorkUnit Config SNMP
- IOCTL Firewall VtamData Telnet Vtam

- Asid List: ()
- JobNameList: ()
- PortList: ()
IpAddrList : ()
Xwriter : Disconnected
Dwriter : Disconnected
Trace Count : 25,553
Lost Count : 2
Lost Time : 2005/08/24 20:21:16
Wrap Count : 1
Wrap Time : 2005/08/24 20:36:05
==============================================================================
Device Interface: 7F5DC410
  Device: LOOPBACK  Devtype: LOOPBACK  State: Active
  Address: **** ****

Physical Interface: 7F5DA230
  Name: LOOPBACK  Protocol: LOOPBACK  State: Active
  NetNum: 0  QueSize: 0  Bytein: 13,554  Byteout: 13,554
  Index: 2
  Bsd Routing Parameters:
    MtuSize: 0  Metric: 0
    SubnetMask: 0.0.0.0  DestAddr: 0.0.0.0
  SNMP Input Counters:
    Octets: 13,554  Unicast: 214
    NonUnicast: 0  Discarded: 0
    Error: 0  Unkn Type: 0
    Broadcast: 0  Multicast: 0
  SNMP Output Counters:
    Octets: 13,554  Unicast: 214
    NonUnicast: 0  Discarded: 0
    Error: 0  Queue Len: 0
    Broadcast: 0  Multicast: 0

IPv4 Search Patricia tree  Address: 7F55CF10
  Search Ptree Reader Count: 0
Route: 7F5DC270
  Name: LOOPBACK  Type: Host  State: Active
  Subnet Mask: 255.255.255.255  Addr: 127.0.0.1
  Protocol: Configuration  Gate: 0.0.0.0
  Mtu Size: 65535  Ref Cnt: 4  Tos: 0
  Metric1: 0  Metric2: -1
  Metric3: -1  Metric4: -1
  Metric5: -1  Age: 2005/08/24 20:21:19

IPv6 Search Patricia tree  Address: 7F55C370
  Search Ptree Reader Count: 0

Logical Interface: 7F55C110
  Name: LOOPBACK  Protocol: LOOPBACK  State: Active
  Subnet Mask: 255.255.255.255  Addr: 127.0.0.1
  Mtu Size: 65535
Physical Interface: 7F3E2510
  Name: LOOPBACK6  Protocol: LOOPBACK6  State: Active
  NetNum: 0  QueSize: 0  Bytein: 0  Byteout: 0
  Index: 3
  Bsd Routing Parameters:
    MtuSize: 0  Metric: 0
    SubnetMask: 0.0.0.0  DestAddr: 0.0.0.0
  SNMP Input Counters:
    Octets: 0  Unicast: 0
    NonUnicast: 0  Discarded: 0
    Error: 0  Unkn Type: 0
    Broadcast: 0  Multicast: 0
  SNMP Output Counters:
    Octets: 0  Unicast: 0
IPv4 Search Patricia tree Address: 7F55CF10
Search Ptree Reader Count: 0

IPv6 Search Patricia tree Address: 7F55C370
Search Ptree Reader Count: 0

Route: 7F3B4730
Name: LOOPBACK6 Type: Host State: Active
Subnet Prefix: 128 Addr: ::1
Protocol : Configuration Gate: ::0
Mtu Size: 65535 Ref Cnt: 0 Tos: 0
Metric1: 0 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/24 20:21:23

Logical Interface: 7F3B4A90
Name: LOOPBACK6 Protocol: LOOPBACK6 State: Active
Subnet Prefix: 128 Addr: ::1
Mtu Size: 65535

==============================================================================
Device Interface: 7F271410
Device: VIPA16 Devtype: VIRTual State: Active
Address: **** ****

Physical Interface: 7F3E2D10
Name: VIPA16 Protocol: VIRTUAL6 State: Active
NetNum: 0 QueSize: 0 Bytein: 0 Byteout: 0
Index: 5
Bsd Routing Parameters:
MtuSize: 0 Metric: 0
SubnetMask: 0.0.0.0 DestAddr: 0.0.0.0
SNMP Input Counters:
Octets: 0 Unicast: 0 NonUnicast: 0 Discarded: 0 Error: 0 Unkn Type: 0
Broadcast: 0 Multicast: 0
SNMP Output Counters:
Octets: 0 Unicast: 0 NonUnicast: 0 Discarded: 0 Error: 0 Queue Len: 0
Broadcast: 0 Multicast: 0

IPv4 Search Patricia tree Address: 7F55CF10
Search Ptree Reader Count: 0

IPv6 Search Patricia tree Address: 7F55C370
Search Ptree Reader Count: 0

Route: 7F3D32B0
Name: VIPA16 Type: Host State: Active
Protocol : Configuration Gate: ::0
Mtu Size: 65535 Ref Cnt: 0 Tos: 0
Metric1: 0 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/24 20:21:23

Logical Interface: 7F2791A8
Name: VIPA16 Protocol: VIRTUAL6 State: Active
### Subnet Prefix: 0
Addr: 50c9:c2d4::a:9:42:130:161
Mtu Size: 65535

---

### Device Interface: 7F26F410
Device: IUTSAMEH
Devtype: MPCPTP
State: Setup
Address: **** ****

#### SAP:
UserID: 10010000
TransId: 00010140
ProviderId: 00010148
Data0: 85B99378
ReqSignal0: 85B99378
RspSignal0: 85B99378
State: Unknown
Retry: 0
Restart: 0
Xstatus: 0

### Physical Interface: 7F3E3110
Name: IUTSAMEH6
Protocol: MPCPTP6
State: Inactive

- NetNum: 0
- QueSize: 0
- Bytein: 0
- Byteout: 0
- Index: 7

#### BSD Routing Parameters:
- MtuSize: 0
- Metric: 0
- SubnetMask: 0.0.0.0
- DestAddr: 0.0.3.232

#### SNMP Input Counters:
- Octets: 0
- Unicast: 0
- NonUnicast: 0
- Discarded: 0
- Error: 0
- Unkn Type: 0
- Broadcast: 0
- Multicast: 0

#### SNMP Output Counters:
- Octets: 0
- Unicast: 0
- NonUnicast: 0
- Discarded: 0
- Error: 0
- Queue Len: 0
- Broadcast: 0
- Multicast: 0

### IPv4 Search Patricia tree
Address: 7F55CF10
Search Ptree Reader Count: 0

### IPv6 Search Patricia tree
Address: 7F55C370
Search Ptree Reader Count: 0

---

### Route: 7F2697F0
Name: IUTSAMEH6
Type: Host
State: Inactive
Subnet Prefix: 128
Protocol: Configuration
Gate: ::0
Addr: fe80::b47d:2e3f:c8c2:9117
Mtural Size: 65535
Ref Cnt: 0
Tos: 0
Metric1: 0
Metric2: -1
Metric3: -1
Metric4: -1
Metric5: -1
Age: 2005/08/24 20:21:23

### Route: 7F3D6448
Name: IUTSAMEH6
Type: Host
State: Inactive
Subnet Prefix: 128
Protocol: Configuration
Gate: ::0
Addr: fec0::42:105:75:161
Mtural Size: 65535
Ref Cnt: 0
Tos: 0
Metric1: 0
Metric2: -1
Metric3: -1
Metric4: -1
Metric5: -1
Age: 2005/08/24 20:21:23

---

### Logical Interface: 7F269AD0
Name: IUTSAMEH6
Protocol: MPCPTP6
State: Inactive
Subnet Prefix: 128
Addr: fe80::b47d:2e3f:c8c2:9117
Mtural Size: 65535

### Logical Interface: 7F27A100
Name: IUTSAMEH6
Protocol: MPCPTP6
State: Inactive
Subnet Prefix: 0
Addr: fec0::42:105:75:161
Mtural Size: 65535

---

### Device Interface: 7F26E410
Device: OSAQDIO2
Devtype: MPCIPA
State: Active
Address: **** ****

Chapter 6. IPCS subcommands  247
SAP:
UserID: 10020000 TransId: 00010141 ProviderId: 00010145
Data@: 94EE174C ReqSignal@: 85B99378 RspSignal@: 85B99378
State: Unknown Retry: 0 Restart: 0 Xstatus: 0

Connection 2:
UserID: 00000000 ProviderId: 90020001
Data@: 00000000 ReqSignal@: 00000000 RspSignal@: 00000000
State: Reset linknum: 00 flags 00

Physical Interface: 7F3E3510
Name: OSAQDIO26 Protocol: IPAQENET6 State: Active
NetNum: 0 QueSize: 0 Bytein: 1,008 Byteout: 1,304
Index: 9
Bsd Routing Parameters:
MtuSize: 0 Metric: 0 SubnetMask: 0.0.0.0 DestAddr: 0.0.3.232
SNMP Input Counters:
Octets: 1,008 Unicast: 0
NonUnicast: 0 Discarded: 0
Error: 0 Unkn Type: 0
Broadcast: 0 Multicast: 6
SNMP Output Counters:
Octets: 1,304 Unicast: 3
NonUnicast: 0 Discarded: 0
Error: 0 Queue Len: 0
Broadcast: 0 Multicast: 10

IPv4 Search Patricia tree Address: 7F55CF10
Search Ptree Reader Count: 0
IPv6 Search Patricia tree Address: 7F55C370
Search Ptree Reader Count: 0

Route: 7F24D270
Name: OSAQDIO26 Type: Host State: Active
Subnet Prefix: 128 Addr: fe80::9:6b00:f71a:422
Protocol : Configuration Gate: ::0
Mtu Size: 9000 Ref Cnt: 0 Tos: 0

Route: 7F1C2090
Name: OSAQDIO26 Type: Host State: Active
Protocol : Configuration Gate: ::0
Mtu Size: 9000 Ref Cnt: 0 Tos: 0
Metric1: 0 Metric2: -1 Metric3: -1 Metric4: -1 Metric5: -1 Age: 2005/08/24 20:21:27

Route: 7F1D08B0
Name: OSAQDIO26 Type: Direct State: Active
Subnet Prefix: 64 Addr: 50c9:c2d4::109:0:0:0
Protocol : ICMP Gate: ::0
Mtu Size: 9000 Ref Cnt: 0 Tos: 0
Metric1: 0 Metric2: -1 Metric3: -1 Metric4: -1 Metric5: -1 Age: 2005/08/24 20:21:25

Route: 7F1D0830
Name: OSAQDIO26 Type: Direct State: Active
Subnet Prefix: 64 Addr: 50c9:c2d4::1a:0:0:0
Protocol : ICMP Gate: ::0
Mtu Size: 9000 Ref Cnt: 0 Tos: 0
Metric1: 0 Metric2: -1 Metric3: -1 Metric4: -1
Logical Interface: 7F24D550
Name: OSAQDI026 Protocol: IPAQENET6 State: Active
Subnet Prefix: 128 Addr: fe80::9:6b00:f71a:422
Mtu Size: 9000

Logical Interface: 7F384190
Name: OSAQDI026 Protocol: IPAQENET6 State: Active
Mtu Size: 9000

Device Interface: 7F26D410
Device: OSAQDI04 Devtype: MPCIPA State: Active
Address: **** ****
SAP:
UserID: 10030000 TransId: 00010142 ProviderId: 00010146
Data@: 94EE174C ReqSignal@: B5B99378 RspSignal@: B5B99378
State: Unknown Retry: 0 Restart: 0 Xstatus: 0

Connection 2:
UserID: 00000000 ProviderId: 90030001
Data@: 00000000 ReqSignal@: 0 Unk Type: 0
State: Reset linknum: 00 flags 00

Physical Interface: 7F3E3910
Name: OSAQDI046 Protocol: IPAQENET6 State: Active
NetNum: 0 QueSize: 0 Bytein: 3,120 Byteout: 1,444
Index: 11

Bsd Routing Parameters:
MtuSize: 0 Metric: 0
SubnetMask: 0.0.0.0 DestAddr: 0.0.3.232

SNMP Input Counters:
Octets: 3,120 Unicast: 2
NonUnicast: 0 Discarded: 0
Error: 0 Unkn Type: 0
Broadcast: 0 Multicast: 19

SNMP Output Counters:
Octets: 1,444 Unicast: 4
NonUnicast: 0 Discarded: 0
Error: 0 Queue Len: 0
Broadcast: 0 Multicast: 10

IPv4 Search Patricia tree Address: 7F55CF10
Search Ptree Reader Count: 0

IPv6 Search Patricia tree Address: 7F55C370
Search Ptree Reader Count: 0

Route: 7F2214E8
Name: OSAQDI0106 Type: Host State: Active
Subnet Prefix: 128 Addr: fe80::9:6b01:f1a:684
Protocol : Configuration Gate: ::0
Mtu Size: 1500 Ref Cnt: 0 Tos: 0
Metric1: -1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/24 20:21:24

Route: 7F1C7A10
Name: OSAQDI0106 Type: Host State: Active
Protocol : Configuration Gate: ::0
Mtu Size: 1500 Ref Cnt: 0 Tos: 0
Metric1: -1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/24 20:21:29
Route: 7F3D3730
Name: OSAQDIO46 Type: Direct State: Active
Subnet Prefix: 64 Addr: 50c9:c2d4::0
Protocol: Configuration Gate: ::0
Mtu Size: 1492 Ref Cnt: 0 Tos: 0
Metric1: -1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/24 20:21:23

Route: 7F1C7870
Name: OSAQDIO46 Type: Direct State: Active
Subnet Prefix: 64 Addr: 50c9:c2d4::a:0:0:0:0
Protocol: ICMP Gate: ::0
Mtu Size: 1500 Ref Cnt: 0 Tos: 0
Metric1: -1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/24 20:21:29

Route: 7F3D3590
Name: OSAQDIO46 Type: Default State: Active
Subnet Prefix: 0 Addr: ::0
Protocol: Configuration Gate: 50c9:c2d4::206:2aff:fe71:4400
Mtu Size: 1492 Ref Cnt: 0 Tos: 0
Metric1: 1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/24 20:21:23

Logical Interface: 7F2217C8
Name: OSAQDIO46 Protocol: IPAQENET6 State: Active
Subnet Prefix: 128 Addr: fe80::9:6b01:f1a:684
Mtu Size: 1500

Logical Interface: 7F3B4050
Name: OSAQDIO46 Protocol: IPAQENET6 State: Active
Mtu Size: 1500

Device Interface: 7F26C410
Device: OSAQDIO7 Devtype: MPCIPA State: Active
Address: **** ****
SAP:
UserID: 10040000 TransId: 00010143 ProviderId: 00010147
Data@: 94EE174C ReqSignal@: 85B99378 RspSignal@: 85B99378
State: Unknown Retry: 0 Restart: 0 Xstatus: 0

Connection 2:
UserID: 00000000 ProviderId: 90040001
Data@: 00000000 ReqSignal@: 00000000 RspSignal@: 00000000
State: Reset linknum: 00 flags 00

Physical Interface: 7F3E3D10
Name: OSAQDIO76 Protocol: IPAQENET6 State: Active
NetNum: 0 QueSize: 0 Bytein: 3,120 Byteout: 1,430
Index: 13
Bsd Routing Parameters:
MtuSize: 0 Metric: 0
SubnetMask: 0.0.0.0 DestAddr: 0.0.3.232
SNMP Input Counters:
Octets: 3,120 Unicast: 3
NonUnicast: 0 Discarded: 0
Error: 0 UnkType: 0
Broadcast: 0 Multicast: 18
SNMP Output Counters:
Octets: 1,430 Unicast: 3
NonUnicast: 0 Discarded: 0
Error: 0 Queue Len: 0
Broadcast: 0 Multicast: 11
IPv4 Search Patricia tree Address: 7F55CF10
Search Ptree Reader Count: 0

IPv6 Search Patricia tree Address: 7F55C370
Search Ptree Reader Count: 0

Route: 7F1C7C90
Name: OSAQDIO76 Type: Host State: Active
Protocol : Configuration Gate: ::0
Mtu Size: 1500 Ref Cnt: 0 Tos: 0
Metric1: 0 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/24 20:21:28

Route: 7F1C26F0
Name: OSAQDIO76 Type: Direct State: Active
Subnet Prefix: 64 Addr: 50c9:c2d4::0:0:0:0
Protocol : ICMP Gate: ::0
Mtu Size: 1500 Ref Cnt: 0 Tos: 0
Metric1: 0 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/24 20:21:26

Route: 7F21B190
Name: OSAQDIO76 Type: Host State: Active
Subnet Prefix: 128 Addr: fe80::9:6b00:151a:594
Protocol : Configuration Gate: ::0
Mtu Size: 1500 Ref Cnt: 1 Tos: 0
Metric1: 0 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/24 20:21:24

Logical Interface: 7F21B470
Name: OSAQDIO76 Protocol: IPAQENET6 State: Active
Subnet Prefix: 128 Addr: fe80::9:6b00:151a:594
Mtu Size: 1500
Logical Interface: 7F3D38D0
Name: OSAQDIO76 Protocol: IPAQENET6 State: Active
Mtu Size: 1500

No IPv4 Lan Groups

IPv6 LAN Group Summary
LanGroup: 1 7F1C24B0
IntfName IntfStatus NDOwner VipaOwner
---------- ---------- ------- ---------
OSAQDIO1076 Active OSAQDIO1076 Yes
OSAQDIO1046 Active OSAQDIO1046 No
LanGroup: 2 7F26F030
IntfName IntfStatus NDOwner VipaOwner
---------- ---------- ------- ---------
OSAQDIO1026 Active OSAQDIO1026 Yes

Analysis of Tcp/IP for TCPCS completed

==== example output for a stack which is not IPv6 enabled ===========
The address of the TSAB is: 1524C000

1524C040 1 TCPCS V1R8 1500E000 1500E0C8 0030 9FFF767F 00000000 Active

1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
1 TCP/IP(s) for CS V1R8 found

Analysis of Tcp/Ip for TCPCS. Index: 1

TCPIP State

TCPIP Status:
Procedure: TCPCS
Version: V1R8
Status: Active
Asid: 0030
Ended: 2005/08/23 15:01:12
Active: 01:11:47.721301 hours

Major Control Blocks
TSEB: 1524C040 TSDB: 1500E000
TSDX: 1500E0C8 TCA: 15B6B418
ITCVT: 1500E398 ITSTOR: 1500E608
DUAF: 1506B018 MRCB: 7F44F0F0
MTCB: 15B858B08 MUCB: 7F4250F0
IPMAIN: 1505D398 Streams_root: 7F604D10
TosMains: 15B8594B8 MIB2: 15B6B078
CdCb: 15B83510 User: 15EC2000
Conf: 15B853CB Stks: 15B83450

TCPIP Subtasks

Task Tcb FirstRB EotECB StopEcb CmpCode RsnCode RTWA
EZBTCPIP 008FF2A0 008FF218 80FFF0DCB 00000000 00000000 00000000 00000000
EPWPITSK 008E4E88 00BDCC10 00000000 00000000 00000000 00000000 00000000
EZBITITTUB 0008E4F0 0008B340 00000000 00000000 00000000 00000000 00000000
EZACCMSSM 0008E4F8B 000FF4A8 00000000 00000000 00000000 00000000 00000000
EZBIIPXSUB 0008E49C0 0008E608 00000000 00000000 00000000 00000000 00000000
EZBIEJEDER 0008E4720 0008E4698 0008FF218 8008E4698 940C4000 00000000 00000000
EZBTLMST 0008E4500 0008E478 0008FF218 00000000 00000000 00000000 00000000
EZACQMNN 0008E143 0008E4160 0008FF218 0008E4160 00000000 00000000 00000000
EZBIZMST 0008E470 0008E43E0 0008FF218 00000000 00000000 00000000 00000000
EZBTSSL 0008D090 0008D080 0008E3E0 00000000 00000000 00000000 00000000
EZBMTCL 0008D078 0008D068 0008E3E0 00000000 00000000 00000000 00000000
EZACFLG 0008DC80 0008DC58 0008FF218 00000000 00000000 00000000 00000000
EZASASUB 0008E680 0008CC70 0008FF218 0008DC70 00000000 00000000 00000000

Storage Cache Information
Total CSA Allocated: 7,703,656
Tcp/ip CSA Limit: 2,147M
Total CSA Elements: 47
Cache Delay: 300 seconds
Scan Delay: 120 seconds
Total cache allocated: 91,760
Total cache elements: 9
Total freed elements: 0
Last cache scan time: 2005/08/23 18:59:41

CSM Status
ECSA Storage: OK
Data Space Storage: OK
Fixed Storage: OK
Alet: 01FF0014 Dspname: CSM64001
Alet: 00000000 Dspname: ........

==============================================================================
Dispatchable Unit Status
DUCB Initializations: 64,057
DUCB Expansions: 3,684
Percent DUCB expansions: 5
Last DUCB scan time: 2005/08/23 19:00:44

1 DUAT control block(s) were found in the DUAF at 1506B018
124 Dispatchable units were found.
No DUs indicate abend.
==============================================================================
CTrace Status:
Member Name: CTIEZBN0
Buffer Size: 4,194,304
Options: Init Opcmds Opmsgs Socket AFP XCF Access PFS
         API Engine Queue RAW UDP TCP ICMP ARP ND CLAW
         LCS Internet Message WorkUnit Config SNMP
         IOCTL FireWall VtamData TelnVtam Telnet Vtam

Asid List: ()
JobNameList: ()
PortList: ()
IpAddrList: ()
Xwriter: Disconnected
Dwriter: Disconnected
Trace Count: 409,675
Lost Count: 2
Lost Time: 2005/08/23 17:49:25
Wrap Count: 35
Wrap Time: 2005/08/23 18:59:56
==============================================================================
Device Interface: 7F607410
Device: LOOPBACK Devtype: LOOPBACK State: Active
Address: **** ****

Physical Interface: 7F604230
Name: LOOPBACK Protocol: LOOPBACK State: Active
NetNum: 0 QueSize: 0 Bytein: 43,385 Byteout: 43,385
Index: 2
Bsd Routing Parameters:
SubnetMask: 0.0.0.0 DestAddr: 0.0.0.0
SNMP Input Counters:
Octets: 43,385 Unicast: 677
NonUnicast: 0 Discarded: 0
Error: 0 Unkn Type: 0
Broadcast: 0 Multicast: 0
SNMP Output Counters:
Octets: 43,385 Unicast: 677
NonUnicast: 0 Discarded: 0
Error: 0 Queue Len: 0
Broadcast: 0 Multicast: 0

IPv4 Search Patricia tree Address: 7F58EF70
Search Ptree Reader Count: 0
Route: 7F607270
Name: LOOPBACK Type: Host State: Active
Subnet Mask: 255.255.255.255 Addr: 127.0.0.1
Protocol: Configuration Gate: 0.0.0.0
Mtu Size: 65535 Ref Cnt: 5 Tos: 0

Chapter 6. IPCS subcommands 253
Logical Interface: 7F58E110
Name: LOOPBACK  Protocol: LOOPBACK  State: Active
Subnet Mask: 255.255.255.255  Addr: 127.0.0.1
Mtup Size: 65535
==============================================================================
Logical Interface: 7F58E110
Name: LOOPBACK  Protocol: LOOPBACK  State: Active
Subnet Mask: 255.255.255.255  Addr: 127.0.0.1
Mtup Size: 65535
==============================================================================
Logical Interface: 7F58E110
Name: LOOPBACK  Protocol: LOOPBACK  State: Active
Subnet Mask: 255.255.255.255  Addr: 127.0.0.1
Mtup Size: 65535
==============================================================================
Logical Interface: 7F58E110
Name: LOOPBACK  Protocol: LOOPBACK  State: Active
Subnet Mask: 255.255.255.255  Addr: 127.0.0.1
Mtup Size: 65535
==============================================================================
Device Interface: 7F3DF410
Device: IUTSAMEH  Devtype: MPCPTP  State: Setup
Address: **** ****
SAP:
UserID: 10010000  TransId: 00010130  ProviderId: 00010136
Data0: 83BED5E0  ReqSignal0: 83BED5E0  RspSignal0: 83BED5E0
State: Unknown  Retry: 0  Restart: 0  Xstatus: 0
Physical Interface: 7F414510
Name: LSAMEH  Protocol: MPCPTP  State: Inactive
NetNum: 0  QueSize: 0  Bytein: 0  Byteout: 0
Index: 4
Bsd Routing Parameters:
MtupSize: 576  Metric: 0
SubnetMask: 255.0.0.0  DestAddr: 0.0.0.0
SNMP Input Counters:
Octets: 0  Unicast: 0  NonUnicast: 0  Discarded: 0  Error: 0  Broadcast: 0
SNMP Output Counters:
Octets: 0  Unicast: 0  NonUnicast: 0  Discarded: 0  Error: 0  Queue Len: 0  Broadcast: 0
IPv4 Search Patricia tree  Address: 7F58EF70
Search Ptree Reader Count: 0
Route: 7F40B2EB
Name: LSAMEH  Type: Host  State: Inactive
Subnet Mask: 255.255.255.255  Addr: 10.1.0.161
Protocol : Configuration  Gate: 0.0.0.0
Mtup Size: 65535  Ref Cnt: 0  Tos: 0
Metric1: 0  Metric2: -1  Metric3: -1  Metric4: -1  Metric5: -1  Age: 2005/08/23 17:49:32

Logical Interface: 7F3DF090
Name: LSAMEH  Protocol: MPCPTP  State: Inactive
Subnet Mask: 255.255.255.255  Addr: 10.1.0.161
Mtup Size: 65535
==============================================================================
Device Interface: 7F3DD410
Device: OSAQDIO2  Devtype: MPCIPA  State: Active
Address: **** ****
SAP:
UserID: 30040000  TransId: 00010145  ProviderId: 00010147
Data0: 9500474C  ReqSignal0: 83BED5E0  RspSignal0: 83BED5E0
State: Unknown  Retry: 0  Restart: 0  Xstatus: 0
Connection 2:
UserID: 00000000  ProviderId: 90040001
Data0: 00000000  ReqSignal0: 00000000  RspSignal0: 00000000
State: Reset  linknum: 00  flags 00
Physical Interface: 7F414910
Name: LOSAQDIO2   Protocol: IPAQENET   State: Active
NetNum: 0 QueSize: 0 Bytein: 297,780 Byteout: 32,736
Index: 6
Bsd Routing Parameters:
MtuSize: 576 Metric: 1
SubnetMask: 255.255.255.128 DestAddr: 0.0.0.0
SNMP Input Counters:
Octets: 297,780 Unicast: 6
NonUnicast: 0 Discarded: 0
Error: 0 Unkn Type: 0
Broadcast: 66 Multicast: 2,027
SNMP Output Counters:
Octets: 32,736 Unicast: 10
NonUnicast: 0 Discarded: 0
Error: 0 Queue Len: 0
Broadcast: 0 Multicast: 218
IPv4 Search Patricia tree   Address: 7F58EF70
Search Ptree Reader Count: 0

Route: 7F66ECD0
Name: LOSAQDIO2   Type: Host   State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.130.173
Protocol : OSPF   Gate: 9.42.105.139
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 2 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:31:15

Route: 7F66E990
Name: LOSAQDIO2   Type: Host   State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.130.134
Protocol : OSPF   Gate: 9.42.105.158
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 2 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:31:15

Route: 7F66EE70
Name: LOSAQDIO2   Type: Subnetwork   State: Active
Subnet Mask: 255.255.255.252 Addr: 9.42.130.172
Protocol : OSPF   Gate: 9.42.105.139
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 2 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:31:15

Route: 7F66E7F0
Name: LOSAQDIO2   Type: Host   State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.130.85
Protocol : OSPF   Gate: 9.42.105.136
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 2 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:31:15

Route: 7F66E650
Name: LOSAQDIO2   Type: Host   State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.130.46
Protocol : OSPF   Gate: 9.42.105.149
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 2 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:31:15

Chapter 6. IPCS subcommands 255
Route: 7F326AF0
Name: LOSAQDIO2 Type: Subnetwork State: Active
Subnet Mask: 255.255.255.252 Addr: 9.42.130.44
Protocol : OSPF Gate: 9.42.105.149
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 2 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:31:15

Route: 7F66EB30
Name: LOSAQDIO2 Type: Subnetwork State: Active
Subnet Mask: 255.255.255.252 Addr: 9.42.130.132
Protocol : OSPF Gate: 9.42.105.158
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 2 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:31:15

Route: 7F585310
Name: LOSAQDIO2 Type: Host State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.105.153
Protocol : Configuration Gate: 0.0.0.0
Mtu Size: 576 Ref Cnt: 1 Tos: 0
Metric1: 0 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:31:04

Route: 7F320D10
Name: LOSAQDIO2 Type: Direct State: Active
Subnet Mask: 255.255.255.128 Addr: 9.42.105.128
Protocol : OSPF Gate: 0.0.0.0
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:31:04

Route: 7F326950
Name: LOSAQDIO2 Type: Subnetwork State: Active
Subnet Mask: 255.255.255.128 Addr: 9.42.103.128
Protocol : OSPF Gate: 9.42.105.129
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 2 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:31:15

Address Translate Entry: 7F31FB90
 addr: 9.42.105.153 flags: C0 ttldlt: 0
 retries: 0
Address Translate Entry: 7F31FC50
 addr: 9.42.105.184 flags: C0 ttldlt: 0
 retries: 0
Address Translate Entry: 7F31FD10
 addr: 9.42.105.143 flags: C0 ttldlt: 0
 retries: 0
Address Translate Entry: 7F31FFD0
 addr: 9.42.105.141 flags: C0 ttldlt: 0
 retries: 0
Address Translate Entry: 7F31FE90
 addr: 9.42.105.138 flags: C0 ttldlt: 0
 retries: 0
Address Translate Entry: 7F31FF50
 addr: 9.42.105.136 flags: C0 ttldlt: 0
 retries: 0
Address Translate Entry: 7F66E050
 addr: 9.42.105.130 flags: C0 ttldlt: 0
 retries: 0
Chapter 6. IPCS subcommands 257
<table>
<thead>
<tr>
<th>Name: LOSAQDIO4</th>
<th>Type: Host</th>
<th>State: Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Mask: 255.255.255.255</td>
<td>Addr: 9.42.130.117</td>
<td></td>
</tr>
<tr>
<td>Protocol: OSPF</td>
<td>Gate: 9.42.105.110</td>
<td></td>
</tr>
<tr>
<td>Mtu Size: 576</td>
<td>Ref Cnt: 0</td>
<td>Tos: 0</td>
</tr>
<tr>
<td>Metric1: 2</td>
<td>Metric2: -1</td>
<td></td>
</tr>
<tr>
<td>Metric3: -1</td>
<td>Metric4: -1</td>
<td></td>
</tr>
<tr>
<td>Metric5: -1</td>
<td>Age: 2005/08/23 18:16:51</td>
<td></td>
</tr>
</tbody>
</table>

Route: 7F3D7170
<table>
<thead>
<tr>
<th>Name: LOSAQDIO4</th>
<th>Type: Host</th>
<th>State: Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Mask: 255.255.255.255</td>
<td>Addr: 9.42.130.47</td>
<td></td>
</tr>
<tr>
<td>Protocol: OSPF</td>
<td>Gate: 9.42.105.65</td>
<td></td>
</tr>
<tr>
<td>Mtu Size: 576</td>
<td>Ref Cnt: 0</td>
<td>Tos: 0</td>
</tr>
<tr>
<td>Metric1: 3</td>
<td>Metric2: -1</td>
<td></td>
</tr>
<tr>
<td>Metric3: -1</td>
<td>Metric4: -1</td>
<td></td>
</tr>
<tr>
<td>Metric5: -1</td>
<td>Age: 2005/08/23 18:16:51</td>
<td></td>
</tr>
</tbody>
</table>

Route: 7F338290
<table>
<thead>
<tr>
<th>Name: LOSAQDIO4</th>
<th>Type: Host</th>
<th>State: Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Mask: 255.255.255.255</td>
<td>Addr: 9.42.130.11</td>
<td></td>
</tr>
<tr>
<td>Protocol: OSPF</td>
<td>Gate: 9.42.105.65</td>
<td></td>
</tr>
<tr>
<td>Mtu Size: 576</td>
<td>Ref Cnt: 0</td>
<td>Tos: 0</td>
</tr>
<tr>
<td>Metric1: 3</td>
<td>Metric2: -1</td>
<td></td>
</tr>
<tr>
<td>Metric3: -1</td>
<td>Metric4: -1</td>
<td></td>
</tr>
<tr>
<td>Metric5: -1</td>
<td>Age: 2005/08/23 18:16:51</td>
<td></td>
</tr>
</tbody>
</table>

Route: 7F33B0D0
<table>
<thead>
<tr>
<th>Name: LOSAQDIO4</th>
<th>Type: Subnetwork</th>
<th>State: Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Mask: 255.255.255.252</td>
<td>Addr: 9.42.130.8</td>
<td></td>
</tr>
<tr>
<td>Protocol: OSPF</td>
<td>Gate: 9.42.105.65</td>
<td></td>
</tr>
<tr>
<td>Mtu Size: 576</td>
<td>Ref Cnt: 0</td>
<td>Tos: 0</td>
</tr>
<tr>
<td>Metric1: 3</td>
<td>Metric2: -1</td>
<td></td>
</tr>
<tr>
<td>Metric3: -1</td>
<td>Metric4: -1</td>
<td></td>
</tr>
<tr>
<td>Metric5: -1</td>
<td>Age: 2005/08/23 18:16:51</td>
<td></td>
</tr>
</tbody>
</table>

Route: 7F3D75F0
<table>
<thead>
<tr>
<th>Name: LOSAQDIO4</th>
<th>Type: Host</th>
<th>State: Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Mask: 255.255.255.255</td>
<td>Addr: 9.42.130.48</td>
<td></td>
</tr>
<tr>
<td>Protocol: OSPF</td>
<td>Gate: 9.42.105.65</td>
<td></td>
</tr>
<tr>
<td>Mtu Size: 576</td>
<td>Ref Cnt: 0</td>
<td>Tos: 0</td>
</tr>
<tr>
<td>Metric1: 3</td>
<td>Metric2: -1</td>
<td></td>
</tr>
<tr>
<td>Metric3: -1</td>
<td>Metric4: -1</td>
<td></td>
</tr>
<tr>
<td>Metric5: -1</td>
<td>Age: 2005/08/23 18:16:51</td>
<td></td>
</tr>
</tbody>
</table>

Route: 7F33B2F0
<table>
<thead>
<tr>
<th>Name: LOSAQDIO4</th>
<th>Type: Subnetwork</th>
<th>State: Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Mask: 255.255.255.252</td>
<td>Addr: 9.42.130.48</td>
<td></td>
</tr>
<tr>
<td>Protocol: OSPF</td>
<td>Gate: 9.42.105.65</td>
<td></td>
</tr>
<tr>
<td>Mtu Size: 576</td>
<td>Ref Cnt: 0</td>
<td>Tos: 0</td>
</tr>
<tr>
<td>Metric1: 3</td>
<td>Metric2: -1</td>
<td></td>
</tr>
<tr>
<td>Metric3: -1</td>
<td>Metric4: -1</td>
<td></td>
</tr>
<tr>
<td>Metric5: -1</td>
<td>Age: 2005/08/23 18:16:51</td>
<td></td>
</tr>
</tbody>
</table>

Route: 7F33B5E0
<table>
<thead>
<tr>
<th>Name: LOSAQDIO4</th>
<th>Type: Subnetwork</th>
<th>State: Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Mask: 255.255.255.252</td>
<td>Addr: 9.42.130.12</td>
<td></td>
</tr>
<tr>
<td>Protocol: OSPF</td>
<td>Gate: 9.42.105.65</td>
<td></td>
</tr>
<tr>
<td>Mtu Size: 576</td>
<td>Ref Cnt: 0</td>
<td>Tos: 0</td>
</tr>
<tr>
<td>Metric1: 3</td>
<td>Metric2: -1</td>
<td></td>
</tr>
<tr>
<td>Metric3: -1</td>
<td>Metric4: -1</td>
<td></td>
</tr>
<tr>
<td>Metric5: -1</td>
<td>Age: 2005/08/23 18:16:51</td>
<td></td>
</tr>
</tbody>
</table>

Route: 7F3392E0
<table>
<thead>
<tr>
<th>Name: LOSAQDIO4</th>
<th>Type: Subnetwork</th>
<th>State: Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Mask: 255.255.255.252</td>
<td>Addr: 9.42.130.12</td>
<td></td>
</tr>
<tr>
<td>Protocol: OSPF</td>
<td>Gate: 9.42.105.65</td>
<td></td>
</tr>
<tr>
<td>Mtu Size: 576</td>
<td>Ref Cnt: 0</td>
<td>Tos: 0</td>
</tr>
<tr>
<td>Metric1: 3</td>
<td>Metric2: -1</td>
<td></td>
</tr>
<tr>
<td>Metric3: -1</td>
<td>Metric4: -1</td>
<td></td>
</tr>
<tr>
<td>Metric5: -1</td>
<td>Age: 2005/08/23 18:16:51</td>
<td></td>
</tr>
</tbody>
</table>

Route: 7F3381E0
<table>
<thead>
<tr>
<th>Name: LOSAQDIO4</th>
<th>Type: Subnetwork</th>
<th>State: Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Mask: 255.255.255.252</td>
<td>Addr: 9.42.130.12</td>
<td></td>
</tr>
<tr>
<td>Protocol: OSPF</td>
<td>Gate: 9.42.105.65</td>
<td></td>
</tr>
<tr>
<td>Mtu Size: 576</td>
<td>Ref Cnt: 0</td>
<td>Tos: 0</td>
</tr>
<tr>
<td>Metric1: 3</td>
<td>Metric2: -1</td>
<td></td>
</tr>
<tr>
<td>Metric3: -1</td>
<td>Metric4: -1</td>
<td></td>
</tr>
<tr>
<td>Metric5: -1</td>
<td>Age: 2005/08/23 18:16:51</td>
<td></td>
</tr>
</tbody>
</table>
Metric1: 3   Metric2: -1
Metric3: -1   Metric4: -1
Metric5: -1   Age: 2005/08/23 18:16:51

Route: 7F585030
Name: LOSAQDIO4   Type: Host   State: Active
Subnet Mask: 255.255.255.255   Addr: 9.42.105.75
Protocol : Configuration   Gate: 0.0.0.0
Mtu Size: 576   Ref Cnt: 0   Tos: 0
Metric1: 0   Metric2: -1
Metric3: -1   Metric4: -1
Metric5: -1   Age: 2005/08/23 17:49:32

Route: 7F343070
Name: LOSAQDIO4   Type: Direct   State: Active
Subnet Mask: 255.255.255.128   Addr: 9.42.105.0
Protocol : OSPF   Gate: 0.0.0.0
Mtu Size: 576   Ref Cnt: 0   Tos: 0
Metric1: 1   Metric2: -1
Metric3: -1   Metric4: -1
Metric5: -1   Age: 2005/08/23 18:16:51

Route: 7F329AD0
Name: LOSAQDIO4   Type: Host   State: Active
Subnet Mask: 255.255.255.255   Addr: 9.42.104.188
Protocol : OSPF   Gate: 9.42.105.65
Mtu Size: 576   Ref Cnt: 0   Tos: 0
Metric1: 1   Metric2: -1
Metric3: -1   Metric4: -1
Metric5: -1   Age: 2005/08/23 18:16:52

Route: 7F329450
Name: LOSAQDIO4   Type: Host   State: Active
Subnet Mask: 255.255.255.255   Addr: 9.42.104.189
Protocol : OSPF   Gate: 9.42.105.65
Mtu Size: 576   Ref Cnt: 0   Tos: 0
Metric1: 1   Metric2: -1
Metric3: -1   Metric4: -1
Metric5: -1   Age: 2005/08/23 18:16:52

Route: 7F322780
Name: LOSAQDIO4   Type: Host   State: Active
Subnet Mask: 255.255.255.255   Addr: 9.42.104.186
Protocol : OSPF   Gate: 9.42.105.63
Mtu Size: 576   Ref Cnt: 0   Tos: 0
Metric1: 1   Metric2: -1
Metric3: -1   Metric4: -1
Metric5: -1   Age: 2005/08/23 18:16:52

Route: 7F32A150
Name: LOSAQDIO4   Type: Host   State: Active
Subnet Mask: 255.255.255.255   Addr: 9.42.104.185
Protocol : OSPF   Gate: 9.42.105.63
Mtu Size: 576   Ref Cnt: 0   Tos: 0
Metric1: 1   Metric2: -1
Metric3: -1   Metric4: -1
Metric5: -1   Age: 2005/08/23 18:16:52

Route: 7F322950
Name: LOSAQDIO4   Type: Host   State: Active
Subnet Mask: 255.255.255.255   Addr: 9.42.104.187
Protocol : OSPF   Gate: 9.42.105.65
Mtu Size: 576   Ref Cnt: 0   Tos: 0
Metric1: 1   Metric2: -1
Metric3: -1   Metric4: -1
Metric5: -1   Age: 2005/08/23 18:16:52

Chapter 6. IPCS subcommands  259
Route: 7F32A930
Name: LOSAQDIO4 Type: Host State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.104.176
Protocol : OSPF Gate: 9.42.105.65
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:16:52

Route: 7F32A430
Name: LOSAQDIO4 Type: Host State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.104.180
Protocol : OSPF Gate: 9.42.105.63
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:16:52

Route: 7F32B930
Name: LOSAQDIO4 Type: Host State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.104.170
Protocol : OSPF Gate: 9.42.105.65
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:16:52

Route: 7F32C290
Name: LOSAQDIO4 Type: Host State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.104.168
Protocol : OSPF Gate: 9.42.105.126
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:16:52

Route: 7F32BE70
Name: LOSAQDIO4 Type: Host State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.104.161
Protocol : OSPF Gate: 9.42.105.45
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:16:52

Route: 7F32B3F0
Name: LOSAQDIO4 Type: Host State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.104.162
Protocol : OSPF Gate: 9.42.105.65
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:16:52

Route: 7F32CE70
Name: LOSAQDIO4 Type: Host State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.104.161
Protocol : OSPF Gate: 9.42.105.65
Mtu Size: 576 Ref Cnt: 0 Tos: 0
Metric1: 1 Metric2: -1
Metric3: -1 Metric4: -1
Metric5: -1 Age: 2005/08/23 18:16:52

Route: 7F32D108
Name: LOSAQDIO4 Type: Host State: Active
Subnet Mask: 255.255.255.255 Addr: 9.42.104.161
Protocol : OSPF Gate: 9.42.105.45
Mtusize: 576  RefCnt: 0  Tos: 0
Metric1: 2  Metric2: -1
Metric3: -1  Metric4: -1
Metric5: -1  Age: 2005/08/23 18:16:51
Route: 7F32D470
Name: LOSAQDIO4 Type: Host  State: Active
SubnetMask: 255.255.255.255 Addr: 9.42.104.160
Protocol: OSPF  Gate: 9.42.105.65
Mtusize: 576  RefCnt: 0  Tos: 0
Metric1: 1  Metric2: -1
Metric3: -1  Metric4: -1
Metric5: -1  Age: 2005/08/23 18:16:52
Route: 7F32C9F0
Name: LOSAQDIO4 Type: Host  State: Active
SubnetMask: 255.255.255.255 Addr: 9.42.104.163
Protocol: OSPF  Gate: 9.42.105.65
Mtusize: 576  RefCnt: 0  Tos: 0
Metric1: 1  Metric2: -1
Metric3: -1  Metric4: -1
Metric5: -1  Age: 2005/08/23 18:16:52
Route: 7F32C570
Name: LOSAQDIO4 Type: Host  State: Active
SubnetMask: 255.255.255.255 Addr: 9.42.104.166
Protocol: OSPF  Gate: 9.42.105.121
Mtusize: 576  RefCnt: 0  Tos: 0
Metric1: 1  Metric2: -1
Metric3: -1  Metric4: -1
Metric5: -1  Age: 2005/08/23 18:16:52
Route: 7F32AE70
Name: LOSAQDIO4 Type: Host  State: Active
SubnetMask: 255.255.255.255 Addr: 9.42.104.172
Protocol: OSPF  Gate: 9.42.105.65
Mtusize: 576  RefCnt: 0  Tos: 0
Metric1: 1  Metric2: -1
Metric3: -1  Metric4: -1
Metric5: -1  Age: 2005/08/23 18:16:52
Route: 7F399B80
Name: LOSAQDIO4 Type: Host  State: Active
SubnetMask: 255.255.255.255 Addr: 9.42.104.159
Protocol: OSPF  Gate: 9.42.105.45
Mtusize: 576  RefCnt: 0  Tos: 0
Metric1: 1  Metric2: -1
Metric3: -1  Metric4: -1
Metric5: -1  Age: 2005/08/23 18:16:51
Route: 7F32E570
Name: LOSAQDIO4 Type: Host  State: Active
SubnetMask: 255.255.255.255 Addr: 9.42.104.156
Protocol: OSPF  Gate: 9.42.105.65
Mtusize: 576  RefCnt: 0  Tos: 0
Metric1: 1  Metric2: -1
Metric3: -1  Metric4: -1
Metric5: -1  Age: 2005/08/23 18:16:52
Route: 7F32DE70
Name: LOSAQDIO4 Type: Host  State: Active
SubnetMask: 255.255.255.255 Addr: 9.42.104.157
Protocol: OSPF  Gate: 9.42.105.65
Mtusize: 576  RefCnt: 0  Tos: 0
Metric1: 1  Metric2: -1
Metric3: -1  Metric4: -1
Metric5: -1  Age: 2005/08/23 18:16:52
<table>
<thead>
<tr>
<th>Route</th>
<th>Name</th>
<th>Type</th>
<th>State</th>
<th>Subnet Mask</th>
<th>Addr</th>
<th>Protocol</th>
<th>Gate</th>
<th>Mtu Size</th>
<th>Ref Cnt</th>
<th>Tos</th>
<th>Metric1</th>
<th>Metric2</th>
<th>Metric3</th>
<th>Metric4</th>
<th>Metric5</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>7F32EE70</td>
<td>LOSAQDIO4</td>
<td>Host</td>
<td>Active</td>
<td>255.255.255.255</td>
<td>9.42.104.154</td>
<td>OSPF</td>
<td>9.42.105.65</td>
<td>576</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>2005/08/23 18:16:52</td>
</tr>
<tr>
<td>7F32F4F0</td>
<td>LOSAQDIO4</td>
<td>Host</td>
<td>Active</td>
<td>255.255.255.255</td>
<td>9.42.104.152</td>
<td>OSPF</td>
<td>9.42.105.65</td>
<td>576</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>2005/08/23 18:16:52</td>
</tr>
<tr>
<td>7F32E9F0</td>
<td>LOSAQDIO4</td>
<td>Host</td>
<td>Active</td>
<td>255.255.255.255</td>
<td>9.42.104.155</td>
<td>OSPF</td>
<td>9.42.105.65</td>
<td>576</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>2005/08/23 18:16:52</td>
</tr>
<tr>
<td>7F32FE70</td>
<td>LOSAQDIO4</td>
<td>Host</td>
<td>Active</td>
<td>255.255.255.255</td>
<td>9.42.104.150</td>
<td>OSPF</td>
<td>9.42.105.65</td>
<td>576</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>2005/08/23 18:16:52</td>
</tr>
<tr>
<td>7F3224D0</td>
<td>LOSAQDIO4</td>
<td>Host</td>
<td>Active</td>
<td>255.255.255.255</td>
<td>9.42.104.149</td>
<td>OSPF</td>
<td>9.42.105.126</td>
<td>576</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>2005/08/23 18:16:52</td>
</tr>
<tr>
<td>7F330190</td>
<td>LOSAQDIO4</td>
<td>Host</td>
<td>Active</td>
<td>255.255.255.255</td>
<td>9.42.104.146</td>
<td>OSPF</td>
<td>9.42.105.65</td>
<td>576</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>2005/08/23 18:16:52</td>
</tr>
<tr>
<td>7F32F9F0</td>
<td>LOSAQDIO4</td>
<td>Host</td>
<td>Active</td>
<td>255.255.255.255</td>
<td>9.42.104.151</td>
<td>OSPF</td>
<td>9.42.105.65</td>
<td>576</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>2005/08/23 18:16:52</td>
</tr>
<tr>
<td>7F330C90</td>
<td>LOSAQDIO4</td>
<td>Host</td>
<td>Active</td>
<td>255.255.255.255</td>
<td>9.42.104.142</td>
<td>OSPF</td>
<td>9.42.105.65</td>
<td>576</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>2005/08/23 18:16:52</td>
</tr>
</tbody>
</table>
Logical Interface: 7F3E7068
Name: LOSAQDIO4    Protocol: IPAQENET    State: Active
Subnet Mask: 255.255.255.255    Addr: 9.42.105.75
Mtu Size: 1492

Device Interface: 7F3DB410
Device: OSAQDIO7    Devtype: MPCIPA    State: Active
Address: **** ****
SAP:
UserID: 10030000    ProviderId: 00010135
Data0: 9500474C    ReqSignal0: 83BED5E0    RspSignal0: 83BED5E0
State: Unknown    Retry: 0    Restart: 0    Xstatus: 0

Connection 2:
UserID: 00000000    ProviderId: 90030001
Data0: 00000000    ReqSignal0: 00000000    RspSignal0: 00000000
State: Reset    Linknum: 00    flags 00

Physical Interface: 7F415110
Name: LOSAQDIO7    Protocol: IPAQENET    State: Active
NetNum: 0    QueSize: 0    Bytein: 583,077    Byteout: 76,876
Index: 10
Bsd Routing Parameters:
MtuSize: 576    Metric: 1
SubnetMask: 255.255.255.128    DestAddr: 0.0.0.0
SNMP Input Counters:
Octets: 583,077    Unicast: 36
NonUnicast: 0    Discarded: 0
Error: 0    Unkn Type: 0
Broadcast: 10    Multicast: 4,121

SNMP Output Counters:
Octets: 76,876    Unicast: 38
NonUnicast: 0    Discarded: 0
Error: 0    Queue Len: 0
Broadcast: 0    Multicast: 482

IPv4 Search Patricia tree
Address: 7F58EF70
Metric5: -1    Age: 2005/08/23 18:16:52

Address Translate Entry: 7F337030
addr: 9.42.105.85    flags: C0    ttl: 0
retries: 0

Logical Interface: 7F3E8100
Name: LOSAQDIO7    Protocol: IPAQENET    State: Active
Subnet Mask: 255.255.255.255    Addr: 9.42.105.85
Mtu Size: 1492

IPv4 LAN Group Summary
LanGroup: 1 7F3DAB00
LnkName    LnkStatus    ArpOwner    VipaOwner
----------    ---------    --------    ---------
LOSAQDIO4    Active    LOSAQDIO4    Yes
LOSAQDIO7    Active    LOSAQDIO7    No

LanGroup: 2 7F320B90
LnkName    LnkStatus    ArpOwner    VipaOwner
----------    ---------    --------    ---------
LOSAQDIO2    Active    LOSAQDIO2    Yes

Analysis of Tcp/Iq for TCPCS completed
TCPIPCS STORAGE

Use this subcommand to display the TCP/IP storage summary referenced in common cached storage.

Under the heading Storage Summary, a "c" in column "c" indicates the address is on the cache queue. A "p" in column "p" indicates that the control block is part of a pool.

Cache storage has 12 bytes from offset four overlaid with a chain pointer and time stamp. This might show incorrect data for cached control blocks.

Tip: The TCPIPCS STORAGE command only reports storage found in caches in common storage. Use the TCPIPCS MAP command to report both common and TCP/IP private storage usage.

Syntax

```
>>-TCPIPCS-STORAGE
(<variable_item>)

(<variable_list>)

<TITLE>

NOTITLE
```

Parameters

ALL
Display information about all allocated storage.

CACHE
Display only information about cached storage.

CSA
Display only information about in-use CSA storage.

CSM
Display only information about in-use CSM storage.

TCP, TITLE, NOTITLE
See "Parameters" on page 195 for a description of these parameters.

variable_item
Any one of the following variable parameters.

variable_list
You can repeat from 1-32 of the following variable parameters, each separated by a blank space, within parentheses:

Variable parameters are:

storage_address
Formats the storage header(s) for the TCPIP storage element at address
storage_address. An address is specified as 1-8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a-f or A-F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

When a hexadecimal address is specified as variable_item (or more than one address is specified as variable_list) the timestamp, module name, and an indication of whether the storage is allocated or freed, will be formatted.

**Restriction:** If you specify multiple keywords from the set {ALL,CACHE,CSA,CSM}, all of them are used.

When a BLS18100I message indicating an access failure appears in the report, any counts or analysis dependent on this information cannot be included in the TCPIPCS STORAGE output. Also, an access failure can occur as a result of insufficient user region size. If a BLS18100I message is received for data that is included in the dump, increase the user region size and attempt the TCPIPCS STORAGE subcommand again.

**Sample output of the TCPIPCS STORAGE subcommand**

The following is sample output of the TCPIPCS STORAGE subcommand:

```
TCPIPCS STORAGE
Dataset: IPCS.A594094.DUMPM
Title: TCPSVT V2R10: Job(TCPSVT ) EZBITSTO(HTCP50A 99.281)+
       00077A S4C5/74BE2500 SRB P=0051,S=0051,H=0051

TCPIPC Storage Analysis

Storage Statistics
  cache_delay 0 seconds before cache is freed
  com_totstor 177,578,656 total storage for CSA elements
  com_totelem 21,469 total number of CSA elements
  scan_delay 120 seconds between full scans
  stor_cache 48,416 storage in cache after scan
  num_cache 11 elements in cache after scan
  num_freed 2 elements freed during last scan
  scan_time 1999/10/24 04:06:12 time of last scan
  dsa_init 10,375,262 # of DUCB initializations
  dsa_exp 2,180,028 # of DUCB expansions

The control block at 000AC010 (Prev: 00000000) has already been added
The control block at 12A26410 (Prev: 137CB0A0) has already been added

21,907 storage elements found
177,228K bytes of storage allocated

Cached Storage

<table>
<thead>
<tr>
<th>Addr</th>
<th>Size</th>
<th>Key</th>
<th>Sp</th>
<th>Cblk</th>
<th>Time Stamp (GMT)</th>
<th>Alloc</th>
<th>Common Pool</th>
<th>Addr</th>
<th>Module Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>12E6DCB0</td>
<td>304</td>
<td>6</td>
<td>241</td>
<td>CFGM</td>
<td>B30A8EDF19BD18C3</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12774310</td>
<td>3056</td>
<td>6</td>
<td>241</td>
<td>CFGM</td>
<td>B30A8E3DDBBB1943</td>
<td>10</td>
<td>Index was 29</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The control block at 0E289010 (prev: 12857650) was not available
Unable to locate storage at 0E289010

Cache pointers are in a loop at 12774310 for index 29
The control block at 0E289010 (prev: 12857730) was not available
Unable to locate storage at 0E289010

2 control blocks found for Common non-fetch protected storage
3376 bytes allocated in Common non-fetch
4366931 total allocations

```

Chapter 6. IPCS subcommands 265
Storage Summary Statistics

<table>
<thead>
<tr>
<th>Type</th>
<th>Count</th>
<th>Size</th>
<th>Count</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Non-fetch protected</td>
<td>21460</td>
<td>177489K</td>
<td>2</td>
<td>3392</td>
</tr>
<tr>
<td>Common Fetch protected</td>
<td>369</td>
<td>68488</td>
<td>141</td>
<td>36936</td>
</tr>
<tr>
<td>Common persistent</td>
<td>3</td>
<td>192</td>
<td>3</td>
<td>192</td>
</tr>
<tr>
<td>Common SCB pool</td>
<td>80</td>
<td>21128</td>
<td>32</td>
<td>8448</td>
</tr>
<tr>
<td>Private Non-fetch protected</td>
<td>492</td>
<td>395848</td>
<td>156</td>
<td>65192</td>
</tr>
<tr>
<td>Total</td>
<td>22571</td>
<td>178149K</td>
<td>334</td>
<td>114160</td>
</tr>
</tbody>
</table>

22599 blocks of storage for 1807728 bytes were obtained to create this report.

Analysis of Tcp/Ip for TCPSVT completed

**TCPIPCS STREAM**

Use this subcommand to display the stream control blocks.

**Syntax**

```
TCPIPCS STREAM
```

**Parameters**

If no parameters are specified, all stream control blocks are summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

**variable_item**

Any one of the following variable parameters.

**variable_list**

You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

Variable parameters are:

**CB_address**

An address is specified as 1-8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string. Displays only the Stream control block associated with one of the following:
connection_id
Displays the Stream control block with this connection ID. A connection ID is specified as 1-8 hexadecimal digits.

In addition to the variable parameters described above, you can specify the following keyword parameters:

**SUMMARY**
Formats the Stream control blocks in one cross-reference table. SUMMARY is the default.

**DETAIL**
In addition to the SUMMARY display, DETAIL formats the contents of the Stream control blocks.

**TCP, TITLE, NOTITLE**
See "Parameters" on page 195 for a description of these parameters.

**Restriction:** If you specify multiple keywords from the set [SUMMARY, DETAIL], only the last one is used.

### Sample output of the TCPIPCS STREAM subcommand

The following is a sample output of the TCPIPCS STREAM subcommand:

```
TCPIPCS STREAM
Dataset: IPCS.A594094.DUMPM
Title: TCPSVT V2R10: Job(TCPSVT ) EZIPSTO(HTCP50A 99.281)+
      00077A 5AC5/24BE2500 SRB P=0051,S=0051,H=0051

The address of the TSAB is: 12E89BB8

Tseb SI Procedure Version Tsdb Tsdx Asid TraceOpts Status
12E89BF8 1 TCPSVT V2R10 12B57000 12B570C8 0051 9FFFFF7F Active

1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
1 TCP/IP(s) for CS V2R10 found

==================================================================

Analysis of Tcp/IP for TCPSVT. Index: 1

TCPIP Stream Analysis

SKRT at 7F78BD88

Sksc@ Sksh@ CID Driver Api@ Skcb@ Ascb@ Tcb@
7F776EC8 7F777EC8 00000007 IP/NAM 00000000 00000000 00000000 00000000
7F70F0B8 7F61A0B8 00000006 RAW 00000000 00000000 00000000 00000000
7F70F1A8 7F61A6A8 00000005 IP/NAM 00000000 00000000 00000000 00000000
7F70F8CB 7F70F348 00000004 UDP 00000000 00000000 00000000 00000000
7F70F9B8 7F70FA48 00000003 IP/NAM 00000000 00000000 00000000 00000000
```

Chapter 6. IPCS subcommands 267
TCPIPCS TCB

Use this subcommand to display the Master Transmission Control Block (MTCB) and any Transmission protocol Control Blocks (TCBs) defined in the TCP hash table.

Syntax

```
TCPIPCS TCB
```

<table>
<thead>
<tr>
<th>variable_item</th>
<th>SUMMARY</th>
<th>DATAQ</th>
<th>DETAIL</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>TCP (tcp_proc_name)</th>
<th>TITLE</th>
<th>NOTITLE</th>
</tr>
</thead>
</table>

Parameters

If no parameters are specified, all TCP control blocks are summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

Variable parameters are:

jobname

Displays only the TCBs with this job name. The job name can be a TCP/IP application name or a stack name. A job name is 1–8 alphanumeric characters.

TCB_address

Displays only the TCB with this address. An address is specified as 1-8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.
connection_id
Displays the TCB with this connection ID. A connection ID is specified as 1-8 hexadecimal digits.

In addition to the variable parameters described above, you can specify the following keyword parameters:

**DATAQ**
Formats TCBs which have data queued on the SEND or RECEIVE queue.

**SUMMARY**
Formats the MTCB contents and lists all the TCBs in one cross-reference table. SUMMARY is the default.

**DETAIL**
In addition to the SUMMARY display, DETAIL formats the contents of the TCBs.

**TCP, TITLE, NOTITLE**
See "Parameters" on page 195 for a description of these parameters.

**Restriction:** If you specify multiple keywords from the set [SUMMARY, DETAIL], only the last one is used.

Sample output of the TCPIPCS TCB subcommand
The following is sample output of the TCPIPCS TCB subcommand:

TCPIPCS TCB
Dataset: IPCS_MV21372.DUMPA
Title: SLIP DUMP ID=TC

The address of the TSAB is: 131B8120

Tseb SI Procedure Version Tsdb Tsdx Asid TraceOpts Status
131B8160 1 TCPSVT V2R10 13C9F000 13C9FOCB 07D3 94FF755F Active

1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
1 TCP/IP(s) for CS V2R10 found

Analysis of Tcp/Ip for TCPSVT. Index: 1

TCP/IP Analysis
TCPIP Main TCP Control Block (MTCB)
MTCB: 13C9E890
+0000 M_MAIN_EYE......... TCP MAIN
+0008 M_TCP_LWRITE_Q..... 7F782868
+000C M_TCP_LREAD_Q...... 7F782828
+0014 M_TCP_DRIVER_STATE. 01
+0018 M_TCPMTX............ 00000000 00000000 D7D60601
+0028 M_TCPMQMX............ 00000000 00000000 D7D60604
+0038 M_TCBLIST_LOCK..... 00000000 00000000 D7D60604
+0048 M_PORT_CEILING..... 000000FF
+004C M_TPI_SEQ#.......... 00000008
+0050 M_PORT_ARRAY....... 7F711FCB
+0054 M_LAST_PORT_NUM.... 0000040C
...

TCB ResrcID ResrcNm TcpState TpiState Local IPAddr/Port Remote IPAddr/Port LuName ApplName UserID
7F603108 00000002 TCPSVT Closed WLOUNBND 0.0.0.0.0 0.0.0.0.0
7F605D08 00000017 FTPUNIX1 Listening WLOIDLE 0.0.0.0.21 0.0.0.0.0
7F605108 00000013 TCPSVT Listening WLOIDLE 0.0.0.0.625 0.0.0.0.0
7F603508 0000000A TCPSVT Listening WLOIDLE 0.0.0.1025 0.0.0.0.0
7F604508 000000EA TCPSVT Established WLOXFER 197.66.103.1..23 197.11.108.1..1032
TCPIPPCS TELNET

Use this subcommand to display either the address, or address and contents, of Telnet control blocks. These include the following:

- TCMA
- TCFG
- TPDB
- Optionally, the TKCB and CVB for a selected session
- A partial TCFG that is being built is also displayed (if found)

Syntax

```
TCPIPPCS TELNET
```

Parameters

If no parameters are specified, all TCP control blocks are summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

**variable_item**

Any one of the following variable parameters.

**variable_list**

You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

Variable parameters are:

**LUname**

Displays only the session control blocks for the 8-character logical unit name. If the name is less than eight characters, it is padded on the right with blanks.

**CVB_address**

Displays only the CVB with this address. An address is specified as 1-8 hexadecimal digits. An IPCS symbol name can be specified for an address.
If an address begins with digit a-f or A-F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

token Displays only the session control blocks for the token. The token is a 16-digit hexadecimal value. If the token is less than 16 digits, it is padded on the left with hex zeros.

In addition to the variable parameters described above, you can specify the following keyword parameters:

**All**
Display Telnet connection and XCF information.

**Conn**
Display only Telnet connection information.

**DETAIL**
Displays the contents of the control blocks.

**SUMMARY**
Displays the address of the control blocks. SUMMARY is the default.

**TCP, TITLE, NOTITLE**
See “Parameters” on page 195 for a description of these parameters.

**Xcf**
Display only XCF information.

**Restriction:** If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

**Sample output of the TCP/IPCS TELNET subcommand**
The following is sample output of the TCPIPCS TELNET subcommand:

```
TCPIPCS TELNET
Dataset: IPCS.MV21381.DUMPA
Title: SLIP DUMP ID=TC

The address of the TSAB is: 13391BC0
Tseb SI Procedure Version Tsdx Asid TraceOpts Status
13391C00 1 TCPSVT V2R10 13238000 132380CB 07DE 04041405 Active
13391C80 2 TCPSVT2 V2R10 00000000 00000000 07EB 00000000 Down Stopping
13391D00 3 TCPSVT1 V2R10 12FC3000 12FC30C8 0080 94FF755F Active
13391D80 4 TCPSVT3 V2R10 00000000 00000000 0059 00000000 Down Stopping

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found
4 TCP/IP(s) for CS V2R10 found

Analysis of Tcp/Ip for TCPSVT. Index: 1

TCPIP Telnet Analysis
TMCA at 7F5B1188
Tpdb@ Port Tcfg@ Prof Tkcb@ Token Cvb@ LUname
```

Chapter 6. IPCS subcommands 271
TCPIPCS TIMER

Use this subcommand to display the timer control blocks.

Syntax

```
TCPIPCS TIMER (SUMMARY) (DETAIL) TCP (tcp_proc_name tcp_index) TITLE NOTITLE
```

Parameters

SUMMARY

Displays the contents of the timer control blocks. The timer queue elements (TQEs) and timer IDs (TIDs) are presented in tabular form. SUMMARY is the default.

DETAIL

The timer control blocks are displayed as in the SUMMARY form of the command. In addition, each TQE and each TID is fully displayed.

TCP, TITLE, NOTITLE

See "Parameters" on page 195 for a description of these parameters.

Restriction: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

Sample output of the TCPIPCS TIMER subcommand

The following is sample output of the TCPIPCS TIMER subcommand:

```
TCPIPCS TIMER
Dataset: IPCS.A594094.DUMPF
Title: CHECK NOT ADDR

The address of the TSAB is: 08CE28C0
Tseb SI Procedure Version Tsdb Tsdx Asid TraceOpts Status
08CE2900 1 TCPCS V2R10 086D8000 086D80C8 01F8 10000100 Active
1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
1 TCP/IP(s) for CS V2R10 found
==============================================================================
Analysis of Tcp/Ip for TCPCS. Index: 1

Timer tables at 086DBF80
```
Global TQE Queue for Slot 63:

<table>
<thead>
<tr>
<th>Tqe</th>
<th>Tid</th>
<th>Ecb</th>
<th>Mod</th>
<th>Parm</th>
<th>Msec</th>
<th>TqeFlag</th>
<th>TidFlag</th>
</tr>
</thead>
<tbody>
<tr>
<td>08EDDD58</td>
<td>08EDDD44</td>
<td>00000000</td>
<td>EZBIFIU2</td>
<td>08EDDD40</td>
<td>100</td>
<td>00</td>
<td>20</td>
</tr>
</tbody>
</table>

1 TQE(s) for slot 63 with 0 msec timer offset

Global TQE Queue for Slot 122:

<table>
<thead>
<tr>
<th>Tqe</th>
<th>Tid</th>
<th>Ecb</th>
<th>Mod</th>
<th>Parm</th>
<th>Msec</th>
<th>TqeFlag</th>
<th>TidFlag</th>
</tr>
</thead>
<tbody>
<tr>
<td>086C9020</td>
<td>7F4CEBD0</td>
<td>7F4CEBCC</td>
<td>00000000</td>
<td>00000000</td>
<td>1200000</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>

1 TQE(s) for slot 122 with 128000 msec timer offset

2 TQE(s) were found

No cancelled TQE(s) were found

Analysis of Tcp/Ip for TCPCS completed

TCPIPCS TRACE

Use this subcommand to display information about CTrace.

Syntax

```
TCPIPCS TRACE (SUMMARY)
TCPIPCS TRACE (DETAIL)
TCP (tcp_proc_name)
TCP (tcp_index)

TITLE
NOTITLE
```

Parameters

SUMMARY
Displays a summary of the CTrace status. SUMMARY is the default.

DETAIL
In addition to the SUMMARY information, lists the individual trace buffer entries.

TCP, TITLE, NOTITLE
See “Parameters” on page 195 for a description of these parameters.

Restriction: If you specify multiple keywords from the set [SUMMARY, DETAIL], only the last one is used.
Sample output of the TCPIPCS TRACE subcommand

The following is sample output of the TCPIPCS TRACE subcommand:

TCPIPCS TRACE
Dataset: IPCS.R8A0723.RASDUMP2
Title: EZRPE005

The address of the TSAB is: 09C445D0

<table>
<thead>
<tr>
<th>Tseb</th>
<th>SI Procedure Version Tsdb</th>
<th>Tsdx</th>
<th>Asid</th>
<th>TraceOpts</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>09C44610</td>
<td>TCPCS V1R5</td>
<td>093C1000</td>
<td>093C10C8</td>
<td>0029</td>
<td>9FFF7E7F</td>
</tr>
<tr>
<td>09C44690</td>
<td>TCPCS2 V1R5</td>
<td>00000000</td>
<td>00000000</td>
<td>02A</td>
<td>00000000</td>
</tr>
</tbody>
</table>

2 defined TCP/IP(s) were found
1 active TCP/IP(s) were found
2 TCP/IP(s) for CS V1R5 found

================================================================================================
Analysis of Tcp/IP for TCPCS. Index: 1

Parmlib Member for SYSTCPIP Trace: CTIEZB00
Parmlib Member for SYSTCPIIS Trace: CTIIDS00

Trace Control Area
TCA: 092BD410
+0000 TCAACRONYM........... TCA
+0006 TCAVERSION........... 0006
+0008 TCASIZE.............. 0000CBD0
+0010 TCACURTBE........... 092BE5C8
+0014 TCACURENT.......... 0059C4C0
+0018 TCIABSZ............. 01000000
+001C TCANUMBF............ 00000100
+0020 TCABUFSZ............ 00000010
+0024 TCAMXDAT............ 00000010
+0028 TCAALET............ 01000000
+002C TCARCNT............ 00000410
+0030 TCACNT............. 00000410
+0034 TCALCNT............ 00000000
+0038 TCALLCT............ 00000000 00000000
+0040 TCACOMP............. 00000000
+0044 TCAFILTER.......... 00000000
+0048 TCXWRTSEQ.......... 00000000
+004C TCACNTSSWNK........ 00000000 00000000
+0054 TCAACNT............ 0000

-- Array elements --
+0058 TCACFILTER_ASID..... 0000
+005A TCACFILTER_ASID..... 0000
+005C TCACFILTER_ASID..... 0000
+005E TCACFILTER_ASID..... 0000
+0060 TCACFILTER_ASID..... 0000
+0062 TCACFILTER_ASID..... 0000
+0064 TCACFILTER_ASID..... 0000
+0066 TCACFILTER_ASID..... 0000
+0068 TCACFILTER_ASID..... 0000
+006A TCACFILTER_ASID..... 0000
+006C TCACFILTER_ASID..... 0000
+006E TCACFILTER_ASID..... 0000
+0070 TCACFILTER_ASID..... 0000
+0072 TCACFILTER_ASID..... 0000
+0074 TCACFILTER_ASID..... 0000
+0076 TCACFILTER_ASID..... 0000
Event Trace Statistics for SYSTCPIP

Size of the Trace Control Area: 52176
Size of the trace buffer: 16384K
Size of a trace segment: 64K
Number of trace segments: 256
Maximum trace record size: 14,336
Number of trace records requested: 16,643
Number of trace records recorded: 16,643
Number of trace segments filled: 89
Average records per segment: 187
Average records per table: 47,872
Trace status: Active
XWriter status: Disconnected
Number of buffers written: 0
Lost record count: 0
Lost record time: 1900/01/01 00:00:00.000000
Trace table wrap count: 1
Trace table wrap time: 2001/09/05 12:41:47.461043
Average records per wrap: 16,643

Data Trace Statistics for SYSTCPDA
Size of the trace buffer . . . 32768K
Size of a trace segment . . . 64K
Number of trace segments . . 512
Number of trace records requested . . 104
Number of trace records recorded . . 104
Number of trace segments filled . . 1
Trace status . . . . . . . . . Active
XWriter status . . . . . . . . Connected
Number of lost records . . . 0

IDSTRACE Statistics for SYSTCPIP
Size of the trace buffer . . . 32768K
Size of a trace segment . . . 64K
Number of trace segments . . 512
Number of trace records requested . . 0
Number of trace records recorded . . 0
Number of trace segments filled . . 1
Trace status . . . . . . . . . Active
XWriter status . . . . . . . . Disconnected
Number of lost records . . . 0
Lost record time . . . . . . . 1900/01/01 00:00:00.000000
Trace table wrap count . . . . 0
Trace table wrap time . . 1900/01/01 00:00:00.000000

TsebTrace_Opts: 9FFF7E7F
Options: Init Opcmds Opmsgs Socket AFP XCF Access PFS API
Engine Streams Queue RAW UDP TCP ICMP ARP CLAW LCS
Internet Message WorkUnit Config SNMP IOCTL FireWall
VtamdA TelnVtam Telnet Vtam

256 SYSTCPIP Trace Buffer Elements were found
0 SYSTCPIP Trace Buffer Elements were formatted
512 SYSTCPDA Trace Buffer Elements were found
0 SYSTCPDA Trace Buffer Elements were formatted
512 SYSTCPIP Trace Buffer Elements were found
0 SYSTCPIP Trace Buffer Elements were formatted

Analysis of Tcp/Ip for TCPCS completed

TCPIPCS TREE
Use this subcommand to display the structure of TCP/IP Patricia trees.

Syntax

"TCPIPCS"
### Parameters

**ALL**
Display structure of all TCP/IP trees. ALL is the default.

**ARP**
Display only structure of ARP trees.

**IPSEC**
Display only structure of IP security trees.

**IQDIO**
Display only structure of iQDIO and QDIOACCEL trees.

**ND**
Display only structure of Neighbor Discovery trees.

**NETACC**
Display only structure of NetAccess trees.

**NETACCV4**
Display only structure of IPv4 NetAccess trees.

**NETACCV6**
Display only structure of IPv6 NetAccess trees.

**POLICY**
Display only structure of Service Policy trees.

**QDIOACCEL**
Display only structure of iQDIO and QDIOACCEL trees.

**ROUTE**
Display only structure of both IPv4 and IPv6 route trees.

**ROUTEV4**
Display only structure of IPv4 route trees.
ROUTEV6
   Display only structure of IPv6 route trees.

TCP
   Display only structure of TCP trees.

TELNET
   Display only structure of Telnet trees.

XCF
   Display only structure of XCF trees.

HEADER
   Display tree header information. Not displayed by default.

SUMMARY
   Display the addresses of the control blocks and other data in trees. SUMMARY
   is the default.

DETAIL
   In addition to the SUMMARY display, DETAIL also shows the search key
   values.

BOTH
   Display both active and logically deleted tree nodes. BOTH is the default.

ACTIVE
   Display only active tree nodes.

DELETE
   Display only logically deleted tree nodes

TCP, TITLE, NOTITLE
   See "Parameters" on page 195 for a description of these parameters.

Restrictions:

- If you specify multiple keywords from the set
  (ALL,ARP,IPSEC,IPQDIO,ND,NETACC,NETACCv4,NETACCv6,
  POLICY,IPQDIOACCEL,ROUTE,ROUTEV4,ROUTEV6,TCP,TELNET,XCF), all of
  them are used.

- If you specify multiple keywords from the set (BOTH, ACTIVE, DELETE), only
  the last one is used.

- If you specify multiple keywords from the set (SUMMARY, DETAIL), only the
  last one is used.

Sample output of the TCPIPICS TREE subcommand
The following is sample output of the TCPIPICS TREE subcommand:

TCPIP Tree Analysis

IPv4 NetAccess Search Tree

<table>
<thead>
<tr>
<th>Node@</th>
<th>Bit</th>
<th>Parent</th>
<th>LChild</th>
<th>RChild</th>
<th>Key</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>2B420D678</td>
<td>255</td>
<td>00000000</td>
<td>7F042D90</td>
<td>7F040D50</td>
<td>7F040D50</td>
<td>7F040D50</td>
</tr>
<tr>
<td>7F040D230</td>
<td>3</td>
<td>7F040D50</td>
<td>7F040D60</td>
<td>7F040D230</td>
<td>7F040D50</td>
<td>2B1F48B0</td>
</tr>
<tr>
<td>7F040D100</td>
<td>3</td>
<td>7F040D50</td>
<td>7F040D60</td>
<td>7F040D230</td>
<td>7F040D50</td>
<td>2B1F48B0</td>
</tr>
<tr>
<td>7F040D50</td>
<td>4</td>
<td>7F040D90</td>
<td>7F040D50</td>
<td>7F040D50</td>
<td>7F040D50</td>
<td>2B1F48B0</td>
</tr>
<tr>
<td>7F040C50</td>
<td>9</td>
<td>7F040C90</td>
<td>7F040C50</td>
<td>7F040C50</td>
<td>7F040C50</td>
<td>2B1F48B0</td>
</tr>
<tr>
<td>7F040C80</td>
<td>9</td>
<td>7F040C90</td>
<td>7F040C80</td>
<td>7F040C80</td>
<td>7F040C80</td>
<td>2B2064B8</td>
</tr>
<tr>
<td>7F040C20</td>
<td>9</td>
<td>7F040C90</td>
<td>7F040C20</td>
<td>7F040C20</td>
<td>7F040C20</td>
<td>2B2064B8</td>
</tr>
<tr>
<td>7F040C50</td>
<td>10</td>
<td>7F040B10</td>
<td>7F040C50</td>
<td>7F040C50</td>
<td>7F040C50</td>
<td>2B2064B8</td>
</tr>
<tr>
<td>7F040B90</td>
<td>9</td>
<td>7F040B80</td>
<td>7F040B90</td>
<td>7F040B90</td>
<td>7F040B90</td>
<td>2B206638</td>
</tr>
<tr>
<td>7F040B50</td>
<td>9</td>
<td>7F040B80</td>
<td>7F040B50</td>
<td>7F040B50</td>
<td>7F040B50</td>
<td>2B206638</td>
</tr>
<tr>
<td>7F040B10</td>
<td>10</td>
<td>7F040B10</td>
<td>7F040B80</td>
<td>7F040B90</td>
<td>7F040B90</td>
<td>2B206698</td>
</tr>
</tbody>
</table>
### IPv4 NetAccess Search Tree

<table>
<thead>
<tr>
<th>Node@</th>
<th>Bit</th>
<th>Parent</th>
<th>LChild</th>
<th>RChild</th>
<th>Key</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>2B21E818</td>
<td>255</td>
<td>00000000</td>
<td>7F042E10</td>
<td>7F04D670</td>
<td>7F04C2D0</td>
<td>9</td>
</tr>
</tbody>
</table>

11 elements in IPv4 NetAccess Search Tree

### IPv4 NetAccess Update Tree

<table>
<thead>
<tr>
<th>Node@</th>
<th>Bit</th>
<th>Parent</th>
<th>LChild</th>
<th>RChild</th>
<th>Key</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>2B21E818</td>
<td>255</td>
<td>00000000</td>
<td>7F042E10</td>
<td>7F04D670</td>
<td>7F04C2D0</td>
<td>9</td>
</tr>
</tbody>
</table>

11 elements in IPv4 NetAccess Update Tree

### IPv6 NetAccess Search Tree

<table>
<thead>
<tr>
<th>Node@</th>
<th>Bit</th>
<th>Parent</th>
<th>LChild</th>
<th>RChild</th>
<th>Key</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>2B180B8</td>
<td>255</td>
<td>00000000</td>
<td>7F04D830</td>
<td>7F0B2010</td>
<td>7F0D7F0</td>
<td>32</td>
</tr>
</tbody>
</table>

15 elements in IPv6 NetAccess Search Tree

### IPv6 NetAccess Update Tree

<table>
<thead>
<tr>
<th>Node@</th>
<th>Bit</th>
<th>Parent</th>
<th>LChild</th>
<th>RChild</th>
<th>Key</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>2B180B8</td>
<td>255</td>
<td>00000000</td>
<td>7F04D830</td>
<td>7F0B2010</td>
<td>7F0D7F0</td>
<td>32</td>
</tr>
</tbody>
</table>

15 elements in IPv6 NetAccess Update Tree
TCPIPCS TSDB

Use this subcommand to display the TSDB server data block.

Syntax

```
TCPIPCS TSDB TCP (tcp_proc_name) tcp_index TITLE NOTITLE
```

Parameters

TCP, TITLE, NOTITLE
See “Parameters” on page 195 for a description of these parameters.

Sample output of the TCPIPCS TSDB subcommand

The following is sample output of the TCPIPCS TSDB subcommand:

```
TCPIPCS TSDB
Dataset: IPCS.MV213B1.DUMPA
Title: SLIP DUMP ID=TC

The address of the TSAB is: 13391BC0
```

<table>
<thead>
<tr>
<th>Tseb</th>
<th>SI Procedure Version Tsdb</th>
<th>Tsdx</th>
<th>Asid TraceOpts Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>13391C00</td>
<td>TCPSVT</td>
<td>V2R10</td>
<td>1323B000 1323B0C8 07DE 04041405</td>
</tr>
<tr>
<td>13391C80</td>
<td>TCPSVT2</td>
<td>V2R10</td>
<td>00000000 00000000 07E8 00000000</td>
</tr>
<tr>
<td>13391D00</td>
<td>TCPSVT1</td>
<td>V2R10</td>
<td>12FC3000 12FC30C8 0080 94FF755F</td>
</tr>
<tr>
<td>13391D80</td>
<td>TCPSVT3</td>
<td>V2R10</td>
<td>00000000 00000000 0059 00000000</td>
</tr>
</tbody>
</table>

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found
4 TCP/IP(s) for CS V2R10 found

Analysis of Tcp/Ip for TCPSVT. Index: 1

TSDB control block summary

TSDB: 1323B000
+0000 TSDB_ACRONYM............ TSDB
+0004 TSDB_LENGTH............... 00C8
+0006 TSDB_VERSION.............. 0003
+0008 TSDB_STATE............... 0015
+000A TSDB_ASI D................. 07DE

-- Array elements --
+0010 TSDB_MT.................. 11A7E870
+0014 TSDB_MT.................. 962F5E00
....
+0060 TSDB_CTRACE_PARMLIB_NAME. CTIEZB02
+006C TSDB_SMCA................. 00000000
+0070 TSDB_TSRT................ 00000000
+0074 TSDB_FLAGS............... 00000000
+0078 TSDB_CONFIG_PORT........ 00000401
+007C TSDB_OSASF_PORT.......... FFFFFFFF
+0080 TSDB_EZBITMSN0........... 91A8BF90
+0084 TSDB_TERMINATING ECB..... 807EC758
+0088 TSDB_DU AF............... 00000000
+008C TSDB_TSCA................. 13236A58
### TCPIPCS TSDX

Use this subcommand to display the TSDX server data extension.

#### Syntax

```
TCPIPCS TSDX
```

#### Parameters

**TCP, TITLE, NOTITLE**

See "Parameters" on page 195 for a description of these parameters.

#### Sample output of the TCPIPCS TSDX subcommand

The following is sample output of the TCPIPCS TSDX subcommand:

**TCPIPCS TSDX**

Dataset: IPCS.MV21381.DUMPA

Title: SLIP DUMP ID=TC

The address of the TSAB is: 13391BC0

<table>
<thead>
<tr>
<th>Tseb</th>
<th>SI Procedure</th>
<th>Version</th>
<th>Tsd</th>
<th>Asid</th>
<th>TraceOpts</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>13391C00</td>
<td>1 TCPSVT</td>
<td>V2R10</td>
<td>1323B0C8</td>
<td>07DE</td>
<td>04041405</td>
<td>Active</td>
</tr>
<tr>
<td>13391C80</td>
<td>2 TCPSVT2</td>
<td>V2R10</td>
<td>00000000</td>
<td>00000000</td>
<td>07E8</td>
<td>Down Stopping</td>
</tr>
<tr>
<td>13391D00</td>
<td>3 TCPSVT1</td>
<td>V2R10</td>
<td>12FC30C8</td>
<td>008B</td>
<td>94FF755F</td>
<td>Active</td>
</tr>
<tr>
<td>13391D80</td>
<td>4 TCPSVT3</td>
<td>V2R10</td>
<td>00000000</td>
<td>00000000</td>
<td>005B</td>
<td>Down Stopping</td>
</tr>
</tbody>
</table>

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found

4 TCP/IP(s) for CS V2R10 found

Analysis of Tcp/IP for TCPSVT. Index: 1

TSDX control block summary

```
TSDX: 1323B0C8
+0000 TSDX_ACRONYM............ TSDX
+0004 TSDX_LENGTH.............. 0300
+0006 TSDX_VERSION............. 0003
+0008 TSDX_FLAGS............... 00000000
+000C TSDX_ASCB................ 00000000
+0010 TSDX_PROCNAME.......... TCPSVT
+0014 TSDX_TCB................. 00000000
+0018 TSDX_CART.............. 00000000
+0020 TSDX_CONSID............ 00000000
+0024 TSDX_TCB.............. 00000000
```

Analysis of Tcp/IP for TCPSVT completed
TCPIPCS TSEB

Use this subcommand to display the TSEB server anchor block.

Syntax

```
TCPIPCS TSEB
```

**Parameters**

TCP, TITLE, NOTITLE

See “Parameters” on page 195 for a description of these parameters.

Sample output of the TCPIPCS TSEB subcommand

The following is sample output of the TCPIPCS TSEB subcommand:

TCPIPCS TSEB

Dataset: IPCS.MV21381.DUMPA
Title: SLIP DUMP ID=TC

The address of the TSAB is: 13391BC0

Tseb  SI Procedure Version Tsdx  Asid TraceOpts Status
13391C00  1 TCPSVT V2R10  13238000 132380C8 07DE 0E041405 Active
13391C80  2 TCPSVT2 V2R10  00000000 00000000 07EB 00000000 Down Stopping
13391D00  3 TCPSVT1 V2R10  12FC3000 12FC30C8 0080 94FF755F Active
13391D80  4 TCPSVT3 V2R10  00000000 00000000 0059 00000000 Down Stopping

4 defined TCP/IP(s) were found
2 active TCP/IP(s) were found
4 TCP/IP(s) for CS V2R10 found

Analysis of Tcp/IP for TCPSVT. Index: 1

TSEB control block summary

TSEB: 13391C00
+0000 TSEB_ACRONYM.......... TSEB
+0004 TSEB_LENGTH.......... 0080
+0006 TSEB_VERSION.......... 0003
+0008 TSEB_FLAGS.......... 82000000
TCPIPCS TTLS

Display information about Application Transparent Transport Layer Security (AT-TLS), AT-TLS groups, and AT-TLS connections.

Syntax

```
TCPIPCS TTLS [variable_item] [variable_list]
```

Parameters

If no parameters are specified, both AT-TLS connections and AT-TLS groups are summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.
variable_list
From 1–32 of the following variable parameters can be repeated, each
separated by a blank space, within parentheses:

Variable parameters are:

TCB_address
Displays AT-TLS information for the connection with this address. An
address is specified as 1–8 hexadecimal digits. An IPCS symbol name can
be specified for an address. If an address begins with a–f or A–F, prefix the
address with a zero to avoid the address being interpreted as a symbol
name or as a character string.

group_address
Displays information for the AT-TLS group with this address. An address
is specified as 1–8 hexadecimal digits. An IPCS symbol name can be
specified for an address. If an address begins with a–f or A–F, prefix the
address with a zero to avoid the address being interpreted as a symbol
name or as a character string.

connection_id
Displays AT-TLS information for the connection with this connection ID.
An ID is specified as 1–8 hexadecimal digits.

group_id
Displays information for the AT-TLS group with this group ID. An ID is
specified as 1–8 hexadecimal digits.

In addition to the variable parameters described above, the following keyword
parameters can be specified:

CONN
Display only information for AT-TLS connections.

GROUP
Display only information for AT-TLS groups.

ALL
Display information for both AT-TLS connections and groups. ALL is the
default.

SUMMARY
Displays the addresses of the control blocks and other data in tables.
SUMMARY is the default.

DETAIL
In addition to the SUMMARY display, DETAIL also shows the contents of the
control blocks.

TCP, TITLE, NOTITLE
See "Parameters" on page 195 for a description of these parameters.

Restrictions:
• If you specify multiple keywords from the set {CONN, GROUP, ALL}, only the
  last one is used.
• If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the
  last one is used.

Sample output of the TCPIPCS TTLS subcommand
The following is sample output of the TCPIPCS TTLS subcommand:
Analysis of Tcp/IP for TCPSVT. Index: 1

TCP/IP Analysis
TCPIP Main TTLS Control Block (EZBZTTLS)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Value Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0000</td>
<td>TTLS_ACRONYM</td>
<td>EZBZTTLS</td>
</tr>
<tr>
<td>+0008</td>
<td>LOCK_CDS</td>
<td>00000000</td>
</tr>
<tr>
<td>+0008</td>
<td>LOCK_SUSPENDED_GLOBAL</td>
<td>00000000</td>
</tr>
<tr>
<td>+0010</td>
<td>LOCK_SUSPENDED_LOCAL</td>
<td>00000000</td>
</tr>
<tr>
<td>+0014</td>
<td>LOCK_INFO</td>
<td>02D2530A</td>
</tr>
<tr>
<td>+0014</td>
<td>LOCK_INIT</td>
<td>02D25</td>
</tr>
<tr>
<td>+0014</td>
<td>LOCK_INIT1</td>
<td>02D25</td>
</tr>
<tr>
<td>+0016</td>
<td>LOCK_INIT2</td>
<td>02D25</td>
</tr>
<tr>
<td>+0017</td>
<td>LOCK_CLASS</td>
<td>03</td>
</tr>
<tr>
<td>+0018</td>
<td>LOCK_LEVEL</td>
<td>0A</td>
</tr>
<tr>
<td>+0018</td>
<td>TTLS_FLAG1</td>
<td>E8000000</td>
</tr>
<tr>
<td>+001D</td>
<td>TTLS_GRPCNT</td>
<td>13</td>
</tr>
<tr>
<td>+001E</td>
<td>TTLS_TCBCMP_OFF</td>
<td>11</td>
</tr>
<tr>
<td>+001F</td>
<td>TTLS_ABEND_COUNT</td>
<td>00</td>
</tr>
<tr>
<td>+0020</td>
<td>TTLS_INSTABEND</td>
<td>00000000</td>
</tr>
<tr>
<td>+0024</td>
<td>TTLS_TCBPTR</td>
<td>006EB5C8</td>
</tr>
<tr>
<td>+0028</td>
<td>TTLS_RESMGR_TOKEN</td>
<td>00000000</td>
</tr>
<tr>
<td>+0030</td>
<td>TTLS_NBNDPART</td>
<td>00000000</td>
</tr>
<tr>
<td>+0033</td>
<td>TTLS_OUTNBNDPART</td>
<td>00000000</td>
</tr>
<tr>
<td>+003B</td>
<td>TTLS_PCT_STATE</td>
<td>00000000</td>
</tr>
<tr>
<td>+003C</td>
<td>TTLS_PCT_INSTANCEID</td>
<td>00000000</td>
</tr>
<tr>
<td>+0040</td>
<td>TTLS_WORKQ</td>
<td>00000000</td>
</tr>
<tr>
<td>+0040</td>
<td>ITLFPUBLIC</td>
<td>00000000</td>
</tr>
<tr>
<td>+0044</td>
<td>ITLFPUBLIC</td>
<td>00000000</td>
</tr>
<tr>
<td>+0044</td>
<td>ITLFTAIL</td>
<td>00000000</td>
</tr>
<tr>
<td>+0048</td>
<td>TTLS_TERM_ECB</td>
<td>00000000</td>
</tr>
<tr>
<td>+004C</td>
<td>TTLS_TERM_ECB</td>
<td>00000000</td>
</tr>
<tr>
<td>+0050</td>
<td>TTLS_EOT_ECB</td>
<td>00000000</td>
</tr>
<tr>
<td>+0054</td>
<td>TTLS_WORKQ_ECB</td>
<td>00000000</td>
</tr>
<tr>
<td>+005B</td>
<td>TTLS_CLEANUP_TIMER</td>
<td>00000000</td>
</tr>
<tr>
<td>+005B</td>
<td>TTLS_CLEANUP_TIMER</td>
<td>00000000</td>
</tr>
<tr>
<td>+005B</td>
<td>TTLS_CLEANUP_TIMER</td>
<td>00000000</td>
</tr>
<tr>
<td>+005C</td>
<td>TTLS_MSEC</td>
<td>00000000</td>
</tr>
<tr>
<td>+0060</td>
<td>TTLS_TQE</td>
<td>00000000</td>
</tr>
<tr>
<td>+0068</td>
<td>TTLS_MDLST</td>
<td>00000000</td>
</tr>
<tr>
<td>+006D</td>
<td>TTLS_MTQ</td>
<td>00000000</td>
</tr>
<tr>
<td>+0070</td>
<td>TTLS_MDLRA</td>
<td>00000000</td>
</tr>
<tr>
<td>+007B</td>
<td>TTLS_MDLRA</td>
<td>00000000</td>
</tr>
<tr>
<td>+007C</td>
<td>TTLS_MAX_SRBS</td>
<td>00000000</td>
</tr>
<tr>
<td>+0080</td>
<td>TTLS_CURR_SRBS</td>
<td>00000000</td>
</tr>
<tr>
<td>+0084</td>
<td>TTLS_WE_CNT</td>
<td>00000000</td>
</tr>
<tr>
<td>+0088</td>
<td>TTLS_PIPI_ECB</td>
<td>00000000</td>
</tr>
<tr>
<td>+008C</td>
<td>TTLS_PIPI_POOLPTR</td>
<td>00000000</td>
</tr>
<tr>
<td>+0090</td>
<td>TTLS_PIPI_SUSPQ</td>
<td>00000000</td>
</tr>
<tr>
<td>+0090</td>
<td>ITLFPUBLIC</td>
<td>00000000</td>
</tr>
<tr>
<td>+0090</td>
<td>ITLFPUBLIC</td>
<td>00000000</td>
</tr>
<tr>
<td>+0094</td>
<td>ITLFPUBLIC</td>
<td>00000000</td>
</tr>
<tr>
<td>+0094</td>
<td>ITLFPUBLIC</td>
<td>00000000</td>
</tr>
<tr>
<td>+0098</td>
<td>TTLS_ENVNUM</td>
<td>00000000</td>
</tr>
<tr>
<td>+009C</td>
<td>TTLS_GBLTHD</td>
<td>00000000</td>
</tr>
<tr>
<td>+00A0</td>
<td>TTLS_SECOND_HT_TOKEN</td>
<td>00000000</td>
</tr>
</tbody>
</table>
0 TLMST Work Requests Formatted

**TTLS Secondary Map hashtable entries**

<table>
<thead>
<tr>
<th>Pri_TCB0</th>
<th>PID</th>
<th>Local_IP</th>
<th>Remote_IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>7D83B110</td>
<td>0000016D</td>
<td>7D9ED0B0</td>
<td>.. .. WEBSTCP SVTWSRV 00000000 .......</td>
</tr>
<tr>
<td>LocalSocket: 197.11.203.11..1026</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RemoteSocket: 198.11.22.103..1033</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tcb_tcp_state: Established</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tcb_TtlsFlags:Ttls_Gate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSX_Flags1: LookUp_Done</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTLSRule(7E828110): prTTLS-DEFAULT-RULE-OFF (Stale)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTLSGroupAction(7E715190): paTTLS-GLOBAL-OFF (Stale)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 7D5E0910 | 00000924 | 7D442390 | .. .. WEBSTCP SVTWSRV 00000000 ....... |
| LocalSocket: 197.11.105.1..80 |
| RemoteSocket: 197.11.107.1..1066 |
| Tcb_tcp_state: Established |
| Tcb_TtlsFlags:Ttls_Gate Ttls_Enabled Ttls_Started Ttls_Initial_Hs |
| TLSX_Flags1: LookUp_Done HSTimerSet NeedInitSSL |
| TTLSRule(7E6F34D0): Web_Server |
| TTLSGroupAction(7D74B10): Webs_group_action_80 |
| TTLSEnvironmentAction(7D750590): Webs_Environment_80 |

... 

135 TCBs Found
22 TCBs Formatted

**TTLS Group: TNs_group_action_923**

<table>
<thead>
<tr>
<th>Address Group Id</th>
<th>Conns</th>
<th>Tasks</th>
<th>Elements</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>7E5CB7B0</td>
<td>21</td>
<td>0</td>
<td>4</td>
<td>0 2005/01/14 14:38:32</td>
</tr>
</tbody>
</table>

-----TTLS Environments----------------

0 TTLS Environments Formatted

-----TTLS Worker Tasks----------------

**TTLS Worker Task: 7D0B0610**

<table>
<thead>
<tr>
<th>Ducb FuncCode</th>
<th>Rcode</th>
<th>Busy Idle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A138000</td>
<td>3</td>
<td>0 2005/01/14 14:38:31</td>
</tr>
</tbody>
</table>

**TTLS Worker Task: 7D0B0690**

<table>
<thead>
<tr>
<th>Ducb FuncCode</th>
<th>Rcode</th>
<th>Busy Idle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A13E000</td>
<td>3</td>
<td>0 2005/01/14 14:38:31</td>
</tr>
</tbody>
</table>

**TTLS Worker Task: 7D0B06A0**

<table>
<thead>
<tr>
<th>Ducb FuncCode</th>
<th>Rcode</th>
<th>Busy Idle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A141000</td>
<td>3</td>
<td>0 2005/01/14 14:38:31</td>
</tr>
</tbody>
</table>

**TTLS Worker Task: 7D0B05A0**

<table>
<thead>
<tr>
<th>Ducb FuncCode</th>
<th>Rcode</th>
<th>Busy Idle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A126000</td>
<td>3</td>
<td>0 2005/01/14 14:38:31</td>
</tr>
</tbody>
</table>

4 TTLS Worker Tasks Formatted

0 TGRP Work Requests Formatted
0 TGRP Log Requests Formatted

... 

================================================================================
19 TTLS Group Found
19 TTLS Group Formatted
Analysis of Tcp/IP for TCPSVT completed

**TCPIPCS UDP**

Use this subcommand to display the Master UDP Control Block (MUCB) and any UDP Control Blocks (UCBs) in the UDP hash tables or link list.

**Syntax**

```
TCPIPCS UDP
```

```
variable_item
variable_list
TCP tcp_proc_name tcp_index

SUMMARY
DATAQ DETAIL

TITLE NOTITLE
```

**Parameters**

If no parameters are specified, all UDP control blocks are summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

- `variable_item`
  Any one of the following variable parameters.

- `variable_list`
  You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

Variable parameters are:

- `jobname`
  Displays only the UDP control blocks with this job name. The job name can be a TCP/IP application name or a stack name. A job name is 1–8 alphanumeric characters.

- `UCB_address`
  Displays only the UDP control block with this address. An address is specified as 1-8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.
connection_id
Displays the UDP control block with this connection ID. A connection ID is specified as 1–8 hexadecimal digits.

In addition to the variable parameters described above, you can specify the following keyword parameters:

**DATAQ**
Formats UCBs which have data queued on the RECEIVE queue.

**SUMMARY**
Formats the MUCB contents and lists all the UDPs in one cross-reference table. SUMMARY is the default.

**DETAIL**
In addition to the SUMMARY display, DETAIL formats the contents of the UCBs.

**TCP, TITLE, NOTITLE**
See [Parameters](#) for a description of these parameters.

**Restriction:** If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

### Sample output of the TCPIPCS UDP subcommand

The following is sample output of the TCPIPCS UDP subcommand:

```
TCPIPCS UDP
Dataset: IPCS.R8A0723.RASDUMP
Title: EZRPE005
The address of the TSAB is: 098221F0
Tseb Si Procedure Version Tsdb Tsdx Asid TraceOpts Status
09822230 1 TCPCS V1R5 08E85000 08E850C8 001E 9FFF7E7F Active
098222B0 2 TCPCS2 V1R5 08937000 089370C8 01F6 9FFF7E7F Active
  2 defined TCP/IP(s) were found
  2 active TCP/IP(s) were found
  2 TCP/IP(s) for CS V1R5 found
Analysis of Tcp/ip for TCPCS. Index: 1
User Datagram Protocol Control Block Summary
MUCB: 7F50B248
+0000 UMCBETE. MUCB USTKLNKD. 01 UDRVSTAT. 00
+0008 UMCB6FLG. ........ 00010000
+0009 USTKLNKD. ........ 01
+0008 U6DRVSTAT......... 00
+000C UOPENPRT. 00000000 UFREEPRT. 041C MCBMUTEX. 00000000
  00000000 00000000 0D60402
+0028 UDPFCFG... 00000001 0000FFFF 0000FFFF 00000001 80000000
  00000000
+0040 UDPNCFG2... 00000001 0000FFFF 0000FFFF 00000001 80000000
  00000000
+0058 UDPIMB... 00000008 0000004B 00000000 0000004D
  USSCAST.. 00000000 USLPBACK. 00000000 USDN
+0074 USRCVBUF. 0000FFFF USSNDBUF. 0000FFFF UFGRPC... 00
  USERIALV. 00000003 USERIAL1. 00000000 ULAS
+008C ULASTPRT. 0000 ULASTUCB. 00000000 USERIAL2. 00000000
  UIPRQ0.. 7F407968 UIPRQ00.. 7F407928 UIIP
+00A4 UIIP6DQ@ 7F207928
+00BC UDMULTI_NUM. ....... 00000000
+00CD UDMUX_TOKEN......... 7F407BBB 00000003
+00DD UDMULTI@. 00000000
+00D4 UDMULTI_NUM. ....... 00000000
+00D8 UDMULTI@. ......... 00000000
+00DC UDMUX_TOKEN......... 7F207BBB 00000006
+00E4 UDMUX_MULTI_TOKEN. 00000000 00000000
```

Chapter 6. IPCS subcommands  289
IPv6 Unicast Hash Table
UCB ResrcID ResrcNm TpiState Port IPAddr
7F2FCD00 0000000C TCPCS WLOUNBND ::0
1 UCB(s) FOUND
1 UCB(s) FORMATTED

IPv4 Unicast Hash Table
UCB ResrcID ResrcNm TpiState Port IPAddr
7F50C000 00000004 TCPCS WLOUNBND 0.0.0.0
1 UCB(s) FOUND
1 UCB(s) FORMATTED

IPv6 Multicast Hash Table
0 UCB(s) FOUND
0 UCB(s) FORMATTED

IPv4 Multicast Link List
0 UCB(s) FOUND
0 UCB(s) FORMATTED

Analysis of Tcp/Ip for TCPCS completed

TCPIPICS VMCF
Use this subcommand to display information about VMCF (Virtual Machine Communication Facility) and IUCV (Inter-User Communication Vehicle) users.

Syntax

```
TCP (tcp_proc_name) tcp_index
```

Parameters
If no parameters are specified, all VMCF control blocks are summarized.
* An asterisk is used as a placeholder if no variable parameters are specified.

`variable_item`
Any one of the following variable parameters.

`variable_list`
You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

Variable parameters are:

`user_id`
Displays only the VMCF control block associated with this user ID. A user_id is 1 - 8 alphanumeric characters.
**ASCB_address**
Displays only the VMCF control blocks associated with this address space control block address. An address is specified as 1-8 hexadecimal digits. An IPCS symbol name can be specified for an address. If an address begins with digit a–f or A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

**ASID_number**
Displays only the VMCF control blocks associated with this address space identifier. An ASID is specified as one to four hexadecimal digits.

In addition to the variable parameters described above, you can specify the following keyword parameters:

**SUMMARY**
Formats the VMCF control blocks in one cross-reference table. SUMMARY is the default.

**DETAIL**
In addition to the SUMMARY display, DETAIL formats the contents of selected VMCF USER control blocks.

**TCP, TITLE, NOTITLE**
See “Parameters” on page 195 for a description of these parameters.

**Restriction:** If you specify multiple keywords from the set [SUMMARY, DETAIL], only the last one is used.

**Sample output of the TCPIPCS VMCF subcommand**
The following is sample output of the TCPIPCS VMCF subcommand:

```
TCPIPCS VMCF ((*) SUMMARY)
Dataset: IPCS.JW11111.DUMPA
Title: IPCS VMCF DUMP

The address of the TSAB is: 08EBC180

<table>
<thead>
<tr>
<th>Tseb</th>
<th>SI Procedure Version</th>
<th>Tsd</th>
<th>Tsdx</th>
<th>Asid</th>
<th>TraceOpts</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>08EBC1C0</td>
<td>1 TCP/ICS</td>
<td>V2R10</td>
<td>089DC000</td>
<td>089DC0C0</td>
<td>01F7</td>
<td>9FFFFF7F</td>
</tr>
</tbody>
</table>

1 defined TCP/IP(s) were found
1 active TCP/IP(s) were found

1 TCP/IP(s) for CS V2R10 found
```

Analysis of Tcp/Ip for TCPCS. Index: 1

**TCP/IP VMCF Analysis**

```
XINF at 09813000
  VMCF CVT : 06A44078
  User Array : 09813090
  Userid Count : 1
  Userid Array : 09817050
  Userid : VMCF
  MSGBUILD : 89802838
  MVPMSGS : 8981A290
  EcB : 00000000
  TNF CVT : 06A63808
  VMCF QD : 00000000
```

Chapter 6. IPCS subcommands  291
TCPIPCS XCF

Use this subcommand to produce a cross-system coupling facility (XCF) analysis report.

Syntax

```
TCPIPCS XCF
   (ALL SUMMARY)
   (CONN DETAIL)
   (DEST)
   (WLM)
```

```
TCP (tcp_proc_name tcp_index)
   TITLE
   NOTITLE
```

Parameters

If no parameters are specified, the dynamic VIPA hash table and partner tables are summarized.

**CONN**
Display only connection hash table optional information.

**DEST**
Display only destination hash table optional information.

**WLM**
Display only workload manager optional information.

**ALL**
Display all optional information. ALL is the default.

**SUMMARY**
Formats the XCF control blocks in one cross-reference table. SUMMARY is the default.

**DETAIL**
In addition to the SUMMARY display, DETAIL formats the contents of XCF control blocks.

**TCP, TITLE, NOTITLE**
See “Parameters” on page 195 for a description of these parameters.

Restrictions: Be aware of the following keyword restrictions:

- If you specify multiple keywords from the set {ALL, CONN, DEST, WLM}, all of them are used.
Sample output of the TCPIPCS XCF subcommand

The following is sample output of the TCPIPCS XCF subcommand:

TCPIPCS XCF
Dataset: IPCS.MV20603.DUMPA

----XFCVT information----
XFCVT@ 12CC7410 Member Name RUSSIATCPSVT
Local PTB 12CC752C PTB Chain 1276A410
DVIPAHashT@ 13239408 IPHashT@ 12A9C010
ConnRteHashT@ 12A9B010 DPTHashT@ 1277D010
WLMData@ 00000000 PolicyPart@ 7F63510B

----DVIPA Hash Table----
DVIPA Hash Table at 13239408
Hash table has size 2056 bytes

DVIPA address 197.11.200.2 index 3
MVSName/TCPName Status/Rank

RUSSIA/TCPSVT 32/255
GERMANY/TCPSVT 12/0

Found 9 entries in the DVIPA Hash Table.

Local Partner Table

----Partner Table Control Block----
Partner Table at 12CC752C
NextPtr: 00000000
MVSName: RUSSIA CPName: RUSSIA
TCPname: TCPSVT IPTable: 12D6F140
IPCount: 21 IPEntries@: 1322D0E8

----Dynamic VIPA Table----
Sending Partner@: 128E1410 GERMANY/TCPSVT
Current Dynamic Home Address: 199.11.87.104
Table Address: 12A98C10 Table Length: 8208
Number of Table Entries: 7

DVIPA entry at 12A98C40
DVIPA origin: DEFINE Dist Status: Unknown:0
DVIPA Flags: MoveImmed
DVIPA Flag2: ()
IP address: 197.11.104.10 Mask: 255.255.255.0

ERRNO command

Use the ERRNO command to search for the name and description of constants used for ERRNO, ErrnoJr, module ID, reason code, and ABEND reason code.
Syntax

```
> ERRNO <type> <value>
```

**Parameters**

*type*

The optional type of value provided:

- **A**: Abend code
- **E**: Errno
- **J**: ErrnoJr
- **M**: Module ID
- **R**: Reason code (default)

*value*

The decimal or hexadecimal value to be converted. By default, the value is assumed to be a hexadecimal number. If the value is less than the maximum size for its type, the value is padded on the left with zeros. Choices are:

- **hhhhhhhh**: An address consisting of 1-8 hexadecimal digits ending with a period. The value at that address is interpreted.
- **hhhhhhhh**: An ERRNO, ERRNO junior, reason code, ABEND code, or module ID consisting of 1-8 hexadecimal digits.
- **hhhhhhhx**: An ERRNO, ERRNO junior, or a module ID consisting of 1-8 hexadecimal digits followed by the letter x.
- **ddddddddn**: An ERRNO, ERRNO junior, or a module ID consisting of 1-8 decimal digits followed by the letter n.

*name*

The name of a module, an ERRNO, an ErrnoJr, or an ABEND reason code.

**Note**: If the name is not found, ERRNO attempts to interpret the name as a hexadecimal value.

**Sample output of the ERRNO command**

This section shows sample outputs of the ERRNO command.

For reason code by hexadecimal value output, code the following:

```
Command ===> errno r 74be72e9
```

ReasonCode: 74BE72E9
Module: EZIPITSTO  ErrnoJr: 29417  JRCMNOCM
Description: Cache Manager encountered a CSM storage shortage

For reason code by address, where the value at address 07093F98 is 74717273, code the following. Type R (reason code) is the default.
For reason code by Errno in decimal, code the following:

Command ===> errno e 129

Errno: 00000081(129) : ENOENT
Description: No such file, directory, or IPC member exists

For reason code by ErrnoJr in hexadecimal, code the following:

Command ===> errno j 6c

ErrnoJr: 0000006C(108) : JRFILENAME THERE
Description: The requested file does not exist

For reason code by abend code in decimal, code the following:

Command ===> errno a 9473

Abend Reason Code: 00002501
Module: Unknown Reason: TcpStorNoCSMstorage
Description: No CSM storage available

For reason code by module ID in hexadecimal, code the following:

Command ===> errno m 74be

ModuleId: 74BE(29886) : EZBITSTO EZBITIINI

For reason code by module name, code the following:

Command ===> errno ezbifinbn

ModuleId: 7418(29720) : EZBITIINI

For reason code by ERRNO name, code the following:

Command ===> errno ebadf

Errno: 00000071(113) : EBADEF
Description: The file descriptor is incorrect

For reason code by ErrnoJr name, code the following:

Command ===> errno jmaxuids

ErrnoJr: 00000013(19) : JRMAXUIDS
Description: The maximum number of OpenMVS user IDs is exceeded

For reason code by ABEND reason name, code the following:

Command ===> errno tcpbadentrycode

Abend Reason Code: 00000401
Module: Unknown Reason: TcpBadEntryCode
Description: Bad Entry code to module
ICMPHDR

This section describes the ICMPHDR command.

Use the ICMPHDR command to display the ICMP header fields.

Syntax

```
ICMPHDR -icmp_address
-skdb_address
-skmb_address
-size
-HELP

```

Parameters

* To omit this positional parameter when using the HELP keyword.

`icmp_address`
The address of an ICMP header or the symbol for the address.

`skdb_address`
The address of an SKDB control block or the symbol for the address.

`skmb_address`
The address of an SKMB control block or the symbol for the address.

`size`
The amount of data to display. If the size is greater than the size of the header, the variable portion of the header is displayed if it exists. Must be one to three hexadecimal digits.

HELP
Display IPHDR usage and syntax information instead of the control blocks.

ALL
Display function, operands, and syntax information for the IPHDR command. ALL is the default.

FUNCTION
Display only function information.

OPERANDS
Display only operand information.

SYNTAX
Display only syntax information.

Restriction: If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them are used.

Sample output of the ICMPHDR command
Following is sample output of the ICMPHDR command.

```
ICMP HDR 9D77428 256

ICMPv6
Type/Code    : ECHO Request    CheckSum: 4F51 0000
Id           : 0028             Seq: 0
Time         : 2002/05/23 18:43:00.332756
```

z/OS V1R11.0 Comm Svr: IP Diagnosis Guide
Use the IPHDR command to display the IP header fields.

**Parameters**

* To omit this positional parameter when using the HELP keyword.

`ip_header_address`  
The address of an IP header or the symbol for the address.

`rcb_address`  
The address of a raw control block or the symbol for the address.

`tcb_address`  
The address of a TCP/IP TCB control block or the symbol for the address.

`ucb_address`  
The address of a UDP control block or the symbol for the address.

`skmb_address`  
The address of an SKMB control block or the symbol for the address.

`skdb_address`  
The address of an SKDB control block or the symbol for the address.
**size**
The amount of data to display. If the size is greater than the size of the header, additional protocol headers (if any) are displayed. Must be one to three hexadecimal digits.

**HELP**
Display IPHDR usage and syntax information instead of the control blocks.

**ALL**
Display function, operands, and syntax information for the IPHDR command. ALL is the default.

**FUNCTION**
Display only function information.

**OPERANDS**
Display only operand information.

**SYNTAX**
Display only syntax information.

**Restriction:** If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them are used.

**Sample output of the IPHDR command**
The following is a sample output of the IPHDR command:

```plaintext
IPHDR 09D77400 300

IP Header: 09D77400
IpHeader: Version : 6 Header Length: 40
Class: : 00 Flow: 000000
Payload Length : 264
Hops : 255 Protocol: ICMPv6
Source : ::1 Destination : ::1

ICMPv6
Type/Code : ECHO Request CheckSum: 4F51 0000
Id : 0028 Seq: 0
Time : 2002/05/23 18:43:00.332756
Echo Data : 256

000000 3CE03B34 000513D4 00000000 00000000 00000000 00000000 00000000 00000000
000010 01011213 14151617 16171819 18191A1B 1C1D1E1F
000020 02022223 24252627 26272829 28292A2B 2B2C2D2E2F
000030 03033334 35363738 393A3B3C 3D3E3F 01020304 01020304 01020304 01020304
000040 0A0B0C0D 0E0F1011 12131415 16171819 1A1B1C1D
000050 02030405 06070809 0A0B0C0D 0E0F1011 12131415
000060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000070 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000090 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000A0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000B0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000D0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000F0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

IP Header : 40
000000 60000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Protocol Header : 8
000000 80004F51 00210000
```

z/OS V1R11.0 Comm Svr: IP Diagnosis Guide
RESOLVER

Use the RESOLVER command to format and summarize resolver control blocks and cache information.

Syntax

```
RESOLVER

SUMMARY

variable_item

DETAIL

variable_list

TITLE

NOTITLE

HELP

FUNCTION

OPERANDS

SYNTAX
```

Parameters

If no parameters are specified, information about the Resolver is summarized.

* An asterisk is used as a placeholder if no variable parameters are specified.

variable_item

Any one of the following variable parameters.

variable_list

You can repeat from 1–32 of the following variable parameters, each separated by a blank space, within parentheses:

Variable parameters are:
**jobname**
Displays only the Resolver control blocks for this job name. The job name can be a TCP/IP application name or a stack name. Must be from 1-8 characters.

**ASCB_address**
Displays the Resolver control blocks with this address space control block (ASCB) address. An IPCS symbol name can be specified for the address. The address is specified as 1-8 hexadecimal digits. If an address begins with digit A–F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

**ASID_number**
Displays the Resolver control blocks with this Address Space Identifier (ASID). The ASID is a hexadecimal number containing one to four digits.

**Restriction:** To display the resolver cache information, the resolver address space must be included in the dump and be specified as one of the address spaces to be displayed.

In addition to the variable parameters described above, you can describe the following keyword parameters:

**HELP**
Display RESOLVER usage and syntax information instead of the control blocks.

**ALL**
Displays help about the function, operands, and syntax information for the RESOLVER command. ALL is the default.

**FUNCTION**
Display only function help information.

**OPERANDS**
Display only operands help information.

**SYNTAX**
Display only syntax help information.

**SUMMARY**
Displays the addresses of the control blocks and other data in tables. SUMMARY is the default.

**DETAIL**
In addition to the SUMMARY display, DETAIL also shows the contents of the control blocks.

**TITLE**
The title contains information about the dump and about the RESOLVER command. The title information is displayed as the default. The title contains the following information:
- RESOLVER command input parameters.
- Dump data set name.
- Dump title.

**NOTITLE**
Suppress the title lines. Use this when you are processing lots of commands on the same dump and do not need to see the title information repeated.

**Restrictions:**
If you specify multiple keywords from the set {TITLE, NOTITLE}, only the last one is used.

If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them are used.

**Sample output of the RESOLVER command**

The following is sample output of the RESOLVER command with only DETAIL specified:

```
RESOLVER * DETAIL
Dataset: IPCS.M999999.DUMPA
Title: RESOLVER V1R5:Job(BPXAS) EZBRERSR(HIP6140 20020503)+
       00000006 SOC1/00000001 P=0027,S=0027,H=0027
==============================================================================
Resolver Analysis
RCRT: 0178B058
+0000 CRTENT#. 0000001F CRTRCVT. 89E93000
   -- Array elements --
   +0008 CRTENT#. 0178B128 CRTENT#. 0178B1E0 CRTENT#. 0178B298
   +0014 CRTENT#. 0178B350 CRTENT#. 0178B408 CRTENT#. 0178B4C0
   +0020 CRTENT#. 0178B578 CRTENT#. 0178B630 CRTENT#. 0178B6E8
   +002C CRTENT#. 0178B7A0 CRTENT#. 0178B858 CRTENT#. 0178B910
   +0038 CRTENT#. 0178BCB CRTENT#. 0178BA16 CRTENT#. 0178BA64
   +0044 CRTENT#. 0178BBD1 CRTENT#. 0178BBD4 CRTENT#. 0178BC84
   +0050 CRTENT#. 0178BDD4 CRTENT#. 0178DFC CRTENT#. 0178E848
   +005C CRTENT#. 0178F02 CRTENT#. 0178FC00 CRTENT#. 0178FC00
   +0068 CRTENT#. 0178FC00 CRTENT#. 0178FC00 CRTENT#. 0178FC00
   +0074 CRTENT#. 0178FC00 CRTENT#. 0178FC00 CRTENT#. 0178BF50
   -- End of array --
RCVT: 89E93000
+0000 CRTVTID... CRTV CRTVASCV. 000890D00 CRTVETKN. 7FFCA358
+000C CRTVLX... 00028000 CRTVREFR. 00000004 CRTVTC.. 766C69E8
+0018 CRTVTTOPT. FFBBFFCRTVRTSM. 093F4000 CRTVDFA. 00000001
+0024 CRTVDFA. 00000002 CRTVGBK. 0000003 CRTVGBLI. 00000004
   -- Array elements --
   +0030 CRTVCFG.... -----------------------------------------------
   +006C
   +00A8
   +00E4
   +0120
   +0130 CRTVCFG.... -----------------------------------------------
   +016C
   +01A8
   +01E4
   +0220
   +0230 CRTVCFG.... SYS1.TCPPARMS(TCPDATA) -----------------------------------------------
   +026C
   +02A8
   +02E4
   +0320
   +0330 CRTVCFG.... -----------------------------------------------
   +036C
   +039A
   +03E4
   +0420
   +0430 CRTVCFG.... -----------------------------------------------
   +046C
```

Chapter 6. IPCS subcommands  301
Resolver Address Space is RESOLVER (ASID 01F8)

Global TCPIP.DATA file: SYS1.TCPPARMS(TCPDATA)

Default TCPIP.DATA file: None

Global IPNODES file: None

Default IPNODES file: None

CTCA: 7F6C3C68
+0000 CTCA_CBID...... CTCA
+0004 CTCA_CBSIZE..... 0398
+0006 CTCA_CBVER....... 0001
+0008 CTCA_CNAME..... C5E9C2C3 C3E3D9C3 00F89D00 00000000
+0010 CTCA_CCATTOKEN... B79291E8 7F6C3C68 0A448190 00F89D00
+0018 CTCA_CURCTBS.... 7E674000
+001C CTCA_CURRCD..... 7E67CE0A
+0030 CTCA_TRACE...... 8A76CD98
+0034 CTCA_OPTWORD.... 09E93018
...

Resolver Task Data for USER557 Asid=0027 Tcb@=007F6AB0:

RES_TASK: 7F6D6678
+0000 RES_IDENITIFIER.... RTSK
+0004 RES_LENGTH......... 0BB8
+0006 RES_SUBPOOL........ F9
+0007 RES_USERKEY........ 06
+0008 RES_SEQUENCE#....... 00000004
+0010 RESMGRTOKEN........ 00000000
+0014 RESMGRTDATA....... 00000000
+0018 RES_RTSEX........... 7F6D6678
+001C RES_RPID............ 7F6D7230
RES_STATE...........

...
Chapter 6. IPCS subcommands
SETPRINT

Use the SETPRINT command to change the destination of subsequent IPCS command output. If the IPCSPRNT data set is allocated and being sent to a node, the output of future IPCS commands accumulates (but not displayed at the terminal) until you exit IPCS. When you exit IPCS, the IPCSPRNT data set is sent to the specified node.

Syntax

```plaintext
SETPRINT [ON | OFF [node_name [.user_id]]]
```

Parameters

**ON**
Allocates the IPCSPRNT data set and issues the IPCS command SETDEF PRINT.

**OFF**
Frees the IPCSPRNT data set and issues the IPCS command SETDEF NOPRINT.

`node_name`
The name of a TSO or VM system to which the output is sent.

`user_id`
The user ID on the TSO or VM system to which the output is sent.

**Note**: If `user_id` is specified, there must be a period but no space between `node_name` and `user_id`.

Sample output

If the command completes successfully, there is no output for the SETPRINT command. The following examples are invalid invocations of the SETPRINT command.

Allocating IPCSPRNT when it is already allocated:

```plaintext
setprint on ralvms.testid
IKJ56861I FILE IPCSPRNT NOT UNALLOCATED, DATA SET IS OPEN
```

Freeing IPCSPRNT when it is already freed:

```plaintext
setprint off
BLS21060I PRINT file not open
IKJ56247I FILE IPCSPRNT NOT FREED, IS NOT ALLOCATED
```

SKMSG

This section describes the SKMSG command.

Use the SKMSG command to display the SKMSG fields.
Syntax

Syntax

```
SKMSG -skmb_address
-skdb_address
-skbd_address
-skqu_address
-raw_control_block_address
-tcb_control_block_address
-udp_control_block_address
HELP ( )
```

Parameters

* To omit this positional parameter when using the HELP keyword.

* `skmb_address`
  The address of an SKMB control block or the symbol for the address.

* `skdb_address`
  The address of an SKDB control block or the symbol for the address.

* `skbd_address`
  The address of an SKBD control block or the symbol for the address.

* `skqu_address`
  The address of an SKQU control block or the symbol for the address.

* `raw_control_block_address`
  The address of a RAW control block or the symbol for the address.

* `tcb_control_block_address`
  The address of a TCB control block or the symbol for the address.

* `udp_control_block_address`
  The address of a UDP control block or the symbol for the address.

**HELP**

Display SKMSG usage and syntax information.

**ALL**

Displays help about the function, operands, and syntax information for the SKMSG command. ALL is the default.

**FUNCTION**

Display only function help information.

**OPERANDS**

Display only operands help information.

**SYNTAX**

Display only syntax help information.

**Restriction:** If you specify multiple keywords from the set [ALL, FUNCTION, OPERANDS, SYNTAX], all of them are used.

**Sample output of the SKMSG command**

The following is a sample output of the SKMSG command:

```
SKMSG 15D4D5B8
SKDB at 15D4D5B8
```
TCPHDR

Use the TCPHDR command to display the TCP header fields.

Syntax

```
TCPHDR   tcp_header_address    tcp_control_block_address
         skdb_address        skmb_address
         size
HELP    (    ALL    )
        FUNCTION
        OPERANDS
        SYNTAX
```

Parameters

* To omit this positional parameter when using the HELP keyword.
tcp_header_address
The address of the TCP header or an IPCS symbol.

tcp_control_block_address
The address of a TCP/IP TCP control block or an IPCS symbol.

skdb_address
The address of an SKDB control block or an IPCS symbol.

skmb_address
The address of an SKMB control block or an IPCS symbol.

size
The amount of data to display. If the size is greater than the size of the header, the variable portion of the header (if it exists) is displayed. Must be one to three hexadecimal digits.

HELP
Display TCPHDR usage and syntax information.

ALL
Displays help about the function, operands, and syntax information for the TCPHDR command. ALL is the default.

FUNCTION
Display only function help information.

OPERANDS
Display only operands help information.

SYNTAX
Display only syntax help information.

Restriction: If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them are used.

Sample output of the TCPHDR command
The following is sample output of the TCPHDR command:
TCPHDR 7F522108

TCB at 7F522108

TCP Header at 7F5222D8

7F5222D8  04010402  7228DD16  7228DB82  50107FD8 | ...........b&."Q |
+0010  00000000 | .... |

| Source Port : 1025 |
| Destination Port : 1026 |
| Sequence Number : 1,915,280,662 |
| Ack Number : 1,915,280,258 |
| Header Length : 20 |
| Flags : Ack |
| Window Size : 32728 |
| Checksum : 0000 |
| Urgent Data Pointer : 0000 |

TOD

Use the TOD command to format a hexadecimal time-of-day value into a readable date and time.
Syntax

```plaintext
TOD time_value [ , time_zone ]
```

Parameters

- **time_value**
  The time to be converted. The `time_value` can be specified as either 16 hexadecimal digits or as an address in a dump of an eight-byte STCK value. If less than 16 digits are specified, the value is padded on the right with zeros. If an address is specified, it must be followed by a period. If an address is less than eight hexadecimal digits, it is padded on the left with zeros.

- **time_zone**
  An offset for the time (the difference between local time and GMT). The `time_zone` can be specified either as a word or as a positive or negative decimal value. The recognized words are:
  - LOCAL: Time zone value of zero is used. This is the default.
  - GMT: Greenwich Mean Time
  - EDT: U.S. Eastern Daylight Time zone
  - EST: U.S. Eastern Standard Time zone
  - CDT: U.S. Central Daylight Time zone
  - CST: U.S. Central Standard Time zone
  - MDT: U.S. Mountain Daylight Time zone
  - MST: U.S. Mountain Standard Time zone
  - PDT: U.S. Pacific Daylight Time zone
  - PST: U.S. Pacific Standard Time zone

Sample output of the TOD command

The following are sample outputs of the TOD command.

Sample output for STCK time-of-day with a time zone word:

```
Command ==> ip tod b214030791f3a92c,est
B2140307 91F3A92C : 1999/04/10 20:51:58.684986 TIMEZONE: 0000430E23400000
```

Sample output for an address in the dump where an STCK time-of-day value is located with a negative time zone offset:

```
Command ==> ip tod 11275d4.,-4
B24000E0 51900000 : 1999/05/16 05:36:37.632256 TIMEZONE: FFFCA5817000000
```

UDPHDR

Use the UDPHDR command to display the UDP header fields.
Syntax

```
UDPHDR  UDP_header_address
   skdb_address
   skmb_address
   size
   HELP ( ALL
      FUNCTION
      OPERANDS
      SYNTAX

Parameters

* To omit this positional parameter when using the HELP keyword.

**UDP_header_address**
The address of a UDP header or the symbol for the address.

*Note:* The UDP header has no version or identifier, so it is not possible to
definitively recognize a UDP header given an address in storage.
Therefore, this command formats the storage assuming it is a UDP
header.

**skdb_address**
The address of an SKDB control block or the symbol for the address.

**skmb_address**
The address of an SKMB control block or the symbol for the address.

**HELP**
Display UDPHDR usage and syntax information.

**ALL**
Displays help about the function, operands, and syntax information for the
UDPHDR command. ALL is the default.

**FUNCTION**
Display only function help information.

**OPERANDS**
Display only operands help information.

**SYNTAX**
Display only syntax help information.

*Restriction:* If you specify multiple keywords from the set [ALL, FUNCTION,
OPERANDS, SYNTAX], all of them are used.

Sample output of the UDPHDR command

The following is a sample output of the UDPHDR command:

```
UDPHDR 08D0A0DB

UDP Header at 08D0A0EC
08D0A0EC 040700A1 0033CD23 | ...~.... |

Source port : 1031
Destination port : 161
Datagram Length : 51
Checksum : CD23
```

Chapter 6. IPCS subcommands  309
Installing TCP/IP IPCS subcommands by using the panel interface

To use the panel interface to the TCP/IP IPCS subcommands, you can either display the panels using an IPCS command or connect the TCP/IP ISPF panels to an existing ISPF panel. No additional installation steps are required to display the panels using an IPCS command. To connect the TCP/IP ISPF panels to an existing panel, find an existing panel where you wish to add TCP/IP as an option and modify the panel. Modify the panel by adding the TCP/IP option, which processes the following command:

```plaintext
PGM(BLSGSCMD) PARM(%EZBTCPEX) NEWAPPL(EZBD)
```

where BLSGSCMD is the IPCS command, EZBTCPEX is the TCP/IP REXX exec, and EZBD is the TCP/IP key list prefix.

You can also start the TCPIP panel interface by performing the following steps:
1. Log on to TSO.
2. Access IPCS to display the IPCS Primary Option menu. Figure 26 shows an example of an IPCS Primary Option Menu.
3. Select option 2, “ANALYSIS - Analyze dump contents”.
4. Select option 6, “COMPONENT - MVS component data”.
5. Scroll down to “TCPIP TCP/IP Dump Analysis” and select it.

Entering TCP/IP IPCS subcommands

You can enter the TCP/IP IPCS subcommands as an IPCS command, either by using panels provided by TCP/IP or by using the IPCS batch facility.

Steps for entering a TCP/IP IPCS subcommand

Follow these steps to enter a TCP/IP IPCS subcommand (you can use the IPCS Subcommand Entry panel).

1. Log on to TSO.
2. Access IPCS to display the IPCS Primary Option Menu. Figure 26 shows an example of an IPCS Primary Option Menu.
3. Select option 4, COMMAND.
4. Type the TCP/IP IPCS subcommand. Figure 27 shows the IPCS Subcommand Entry panel with a subcommand entered.

---

You can invoke the TCP/IP IPCS panels in one of the following ways:

- Invoke the panel REXX exec as an IPCS Subcommand. Follow the steps above for entering a TCP/IP IPCS subcommand using the IPCS Subcommand Entry panel and enter the command:
  
  EZBTCPEX

- Invoke the TCP/IP IPCS panels by selecting the option provided in the installation section above.

For either method, you should see the main menu for the TCP/IP IPCS commands shown in Figure 28 on page 312.

Select an option, and the panels prompt you for additional menu choices or input for the specific TCP/IP IPCS subcommand you select. After all input has been selected, the TCP/IP IPCS subcommand is invoked using the current default dump data set. If the dump data set is for Telnet, only the commands indicated "Available for Telnet" in the IPCS command list provide data. The commands not supported by Telnet return no data.
Step for using the batch option

Perform this step to access IPCS commands using the batch processing interface.

- Prepare the JCL data set. Refer to the \textit{z/OS MVS IPCS User's Guide} and \textit{z/OS MVS IPCS Commands}.

The following is a sample command (single command):

\texttt{\%TCP/IPCS TELNET (* DETAIL)}
Part 3. Diagnosing z/OS Communications Server components
Chapter 7. Diagnosing problems with the z/OS Load Balancing Advisor

The z/OS Load Balancing Advisor is a system that comprises outboard load balancers (LBs), an Advisor, and one or more Agents.

This topic discusses problem diagnosis of the Advisor and Agents and includes the following sections:
- "Diagnostic data"
- "Diagnosing Advisor and Agent problems" on page 316
- "Debug settings and corresponding syslogd priority levels" on page 320

Tip: For diagnosing problems with the load balancer, refer to the appropriate load balancer documentation.

Diagnostic data

You might need to collect multiple pieces of data in order to accurately diagnose problems. For example, the following might be useful:
- Console messages for Advisor and Agents
- Output from the MODIFY command for the Advisor and Agents
- syslogd log messages for Advisor and Agents (possibly including debug level trace)
- Advisor and Agent address space dumps and snap output
- Packet traces of Load Balancer data
- TCP/IP CTRACE of Agents and possibly the Advisor
- Netstat displays on TCP/IP stacks managed by Agents
- SNMP information

Guideline: syslogd does not have to be running in order to run the Advisor or Agents; however, syslogd is the only logging facility that either the Advisor or its Agents is capable of using. Useful diagnostic information might be lost if syslogd is not running before the Advisor or Agents are run.

The Advisor and Agent trigger address space dumps when certain unexpected error conditions are encountered. Both a CEEDUMP and address space snap output are produced and written to the data sets or files that are specified by the start procedure CEEDUMP and CEESNAP DD statements, respectively.

If the Advisor or Agent abnormally terminate (for example a 0Cx abend occurs), an unformatted SYSMDUMP is produced and written to the data set that is specified by the start procedure SYSMDUMP DD statement. If you override the Language Environment run-time option TERMTHDACT during the installation or start procedure, the SYSMDUMP might not be produced, or a CEEDUMP might be produced instead. Therefore, you should not override the TERMTHDACT run-time option. Refer to z/OS Language Environment Programming Guide for more information about run-time options.

In other situations, the z/OS operator might need to dump the address space manually.
Packet trace data of Server/Application State Protocol (SASP) protocol messages that are sent between the Advisor and LBs might be needed. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47 for details about how to use the IP packet trace facility.

**Restriction:** When encrypting data, the packet trace data will be encrypted. Use MESSAGE level log messages instead.

The TCP/IP CTRACE trace of the Agents provides some information about data that has been collected from the TCP/IP stack for determining availability and desirability metrics. If the Agent is managing a CINET environment, a TCP/IP CTRACE might be needed in each TCP/IP stack. A TCP/IP CTRACE on the Advisor or Agent TCP/IP stack might also show data that is flowing between the Agents and Advisor. On the Agent TCP/IP stack, the SOCKET, INTERNET, and IOCTL CTRACE options are useful. On the Advisor TCP/IP stack, the INTERNET and SOCKET options are useful. See “Component trace” on page 47 for more information.

The following Netstat displays on stacks that are managed by Agents might be useful:

**HOME**
- Indicates which interfaces exist and which stack owns them

**ALLCONN**
- Indicates the listening TCP sockets and UDP end-points

SNMP information gives information similar to Netstat displays.

---

**Diagnosing Advisor and Agent problems**

This section includes diagnostic information about Advisor and Agent problems.

**Abends**

Messages and error-related information are usually sent to the MVS system console when an abend on the Advisor or Agent occurs. Perform a dump of the error unless the symptoms already match a known problem.

**Workload not distributed to a particular application**

Use the following checklist to determine why workload is not being distributed to an application:

1. Verify that the Advisor is running and that an Agent is running on the MVS system that contains the application. If they are not running, start the Advisor or Agent.
2. If you are using sysplex subplexing, verify that the Advisor and Agents are in the same subplex. If there are multiple TCP/IP stacks in a subplex, ensure the IP addresses used by the Advisor and Agents are DVIPAs defined within a VIPARANGE statement on each of the stacks in the subplex. Review the syslog for the Advisor and Agents for messages indicating what subplex had been used. Each subplex must have an active Advisor associated with it, and each subplex in a z/OS system must have an Agent associated with it.
3. Issue display commands on the Advisor to determine whether any LBs have registered the application. Verify that the LB is connected to the Advisor. If you are using sysplex subplexing, ensure that the load balancers have connectivity to the Advisor’s subplex.
• Verify that the Advisor’s lb_id_list statement includes the IP address of the LB in question if not using AT-TLS.

• Verify that the IP address and protocol of the member on the LB match the IP address and protocol of the application. If the IP addresses or protocols do not match, correct the definition at the LB.

• Verify that the Advisor’s agent_id_list statement contains the IP address and port that the Agent is bound to on the system where the application exists, if not using AT-TLS. If it does not match and you are not using AT-TLS, correct the agent_id_list statement on the Advisor or the advisor_id statement on the Agent.

• Verify that network connectivity exists between the Advisor and the Agent in question. Unexpected loss of network connectivity between the two should result in an immediate action console message and related messages in the Agent and Advisor log. Issue NETSTAT CONN or NETSTAT ALLCONN commands on the Advisor system to see which Agents have connections to the Advisor, and by omission, which do not. Issue the MODIFY DISPLAY command on the Agents in question to verify that the connection to the Advisor is still active. The DISC flag is shown on the MODIFY DISPLAY command when the Agent is not connected to an Advisor. Correct the underlying network connectivity problem. For more information, see Chapter 4, “Diagnosing network connectivity problems,” on page 29.

• If using AT-TLS with SERVAUTH access control checks to validate connections between the Advisor, Agents, and external load balancers, see Chapter 29, “Diagnosing Application Transparent Transport Layer Security (AT-TLS),” on page 701. In addition, ensure that the SERVAUTH class is active. Ensure that the EZB.LBA.LBACCESS.sysname.tcpsysplexgroupname resource profile is defined and that the user ID associated with the external load balancer has READ access to it. Ensure that the EZB.LBA.AGENTACCESS.sysname.tcpsysplexgroupname resource profile is defined and that the Agents have READ access to it. On the system console where the Advisor is running, look for message EZD1280I which indicates that a connection attempt using AT-TLS failed. This message has specific reason codes which indicate the reason for the failure.

• Issue display commands on the Advisor and Agent in question to verify that the application is available and enabled (not quiesced). Start the application or enable the application using the Agent MODIFY ENABLE command.

• Check the log file for ERROR or WARNING messages and take the appropriate corrective action. If ERROR and WARNING level log messages are not enabled, enable them and recheck the log file later.

• Verify that the LB has connectivity to the IP address of the member in question.

Workload not distributed as expected

Use the following checklist to diagnose workload distribution problems:

• Verify that the Advisor’s update interval value is not inordinately large. The Advisor must wait at least two update intervals before beginning to receive enough data to properly calculate weights when an application becomes available or when an Agent is started.

Allow at least three update intervals to expire after an application is started before re-examining the distribution of workload. If workload is occasionally being sent to overloaded applications or systems, adjust the update_interval downward so workload distribution can react more quickly to the pace of new workload requests.
• Periodically issue display commands on the Advisor to check the weights of members within the group in question. Determine whether the weights are consistent with the expected behavior. If the weights are not consistent with expected behavior, see z/OS Communications Server: IP System Administrator’s Commands for more information on how to analyze the member weights; if all releases in the sysplex are not V1R9 or above, note the restrictions and limitations described in this section. If these are consistent with the expected behavior, investigate the problem at the LB. For more information about groups, refer to z/OS Communications Server: New Function Summary.

• Verify that the Advisor’s agent_id_list value contains the IP addresses and ports that each Agent is bound to on the MVS systems where the application exists, if not using AT-TLS. If it does not match and you are not using AT-TLS, correct the agent_id_list statement on the Advisor or the advisor_id statement on the Agent.

• If using AT-TLS with SERVAUTH access control checks to validate connections between the Advisor, Agents, and external load balancers, see Chapter 29, “Diagnosing Application Transparent Transport Layer Security (AT-TLS),” on page 701. In addition, ensure that the SERVAUTH class is active. Ensure that the EZB.LBA.LBACCESS.sysname.tcpysplexgroupname resource profile is defined and that the user ID associated with the external load balancer has READ access to it. Ensure that the EZB.LBA.AGENTACCESS.sysname.tcpysplexgroupname resource profile is defined and that the Agents have READ access to it. On the system console where the Advisor is running, look for message EZD1280I which indicates that a connection attempt using AT-TLS failed. This message has specific reason codes which indicate the reason for the failure.

• Issue display commands at the Advisor to make sure that members of the group in question are not unexpectedly quiesced or unexpectedly unavailable (AVAIL status is NO).

• Issue display commands at the Advisor for all system-level members in the sysplex to verify that the MVS systems have the expected residual capacity.

• Check the log file for ERROR or WARNING messages and take the appropriate corrective action. If ERROR and WARNING level log messages are not enabled, enable them and recheck the log file later.

**Advisor or Agent appears to be hung**

Verify that the Agent or Advisor is actually hung by issuing a MODIFY procname,DISPLAY,DEBUG command. If no response is received, then attempts to stop (not cancel) the application. If the application does not terminate, the application is hung. If the hang occurred while DEBUG-level Advisor or Agent trace was in effect, collect the following problem documentation and call IBM Service.

- Take an SVC dump of the Agent or Advisor address space (depending on which application is hung) and of the OMVS address space including its data spaces.
- Capture the MVS console messages.
- Capture the application (Agent or Advisor) log messages written to syslogd.

If DEBUG-level trace was not in effect at the time, turn on DEBUG-level Advisor or Agent trace, reproduce the problem, collect the problem documentation previously mentioned, and call IBM Service.
Group names in displays are indecipherable

When LBs define group names, the names are coded in UTF-8 format. This character set is a superset of the EBCDIC character set; therefore, not all characters are translatable to EBCDIC. Rename the group names in the LBs to use characters limited to the ASCII character set.

Load balancer connection terminates unexpectedly

Check the following:
- Verify the load balancer administrator has not shut down the load balancer.
- Verify that TCP/IP connectivity still exists between the load balancer and the Advisor (for example, from the Advisor host, ping the address of the load balancer).
- Check the Advisor’s log file for ERROR or WARNING messages and take the appropriate corrective action. If you see an ERROR message indicating a send() operation failed with "errno = EDC8102I Operation would block" you might have too many groups or members registering from the load balancer. Increase the TCPSENDDBFRSIZE parameter of the TCPCONFIG PROFILE.TCPIP statement, or register fewer groups and members from the load balancer, and then try the operation again. If ERROR and WARNING level log messages are not enabled, enable them, repeat the operation, and recheck the log file again.
Tip: Keep in mind that the Advisor has an internal maximum message size of 128K bytes. If this limit is exceeded, the connection is closed and an error message is logged stating that the message is too large and was not received.
- Check the load balancer for errors.

Agent-Advisor connection terminates unexpectedly

Check the following:
- Verify that the Agent’s MVS operator has not shut down the Agent.
- Verify that TCP/IP connectivity still exists between the Agent and the Advisor.
- Check the Advisor’s log file for ERROR or WARNING messages and take the appropriate corrective action. If you see an ERROR message indicating a send() operation failed with errno = EDC8102I Operation would block, you might have too many groups or members registered that belong to the same Agent. Increase the TCPSENDDBFRSIZE parameter of the TCPCONFIG PROFILE.TCPIP statement, or register fewer groups and members belonging to the Agent. Then try the operation again.
Tip: Keep in mind that the Advisor and Agent have an internal maximum message size of 128KB. If this limit is exceeded, the connection is closed and an error message is logged, which states that the message is too big and was not received.

If ERROR and WARNING level log messages are not enabled, enable them, repeat the operation, and recheck the log file again.
- Check the Agent’s log for errors.
- Connectivity can be dropped between the Advisor and Agents if processing slows down too much (due to lots of registered members and/or high debug levels) and the update_interval is configured too low. Using AT-TLS could increase the possibility of this happening. Try increasing the update_interval.

Automatic restart manager (ARM) registration failure

Failure of the Advisor or Agent to properly register with ARM is indicated by a warning-level message written to the log file. This log message is a result of the
IXCARM call failing with the return code and reason codes indicated in the log message. Refer to [z/OS MVS Programming: Sysplex Services Reference](#) for information about interpreting the IXCARM return code and reason code.

One of the common causes of failure is the lack of a security profile. Refer to the EZARACF sample for instructions on how to add an ARM security profile for the application.

When you are using sysplex subplexes, do the following:

- Define an ARM policy with the TARGET_SYSTEM keyword to indicate which systems the element can be restarted on to ensure that the application is restarted only on a system that is in the same subplex.
- Restart the Advisor and Agent on a VTAM system that has been started with an XCFGRPID start option that corresponds with the vv portion of the `sysplex_group_name` in the Advisor and Agent configuration files, and has an available TCP/IP stack with a GLOBALCONFIG XCFGRPID parameter that corresponds with the tt portion of the `sysplex_group_name` in the Advisor and Agent configuration files.
- If there are multiple TCP/IP stacks in a subplex, ensure the IP addresses used by the Advisor and Agents are DVIPAs defined within a VIPARANGE statement on each of the stacks in the subplex.

### Debug settings and corresponding syslogd priority levels

Table 16 summarizes the available debug levels and their associated syslogd priority levels.

<table>
<thead>
<tr>
<th>Logging category/Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None — 0</td>
<td>No messages of any kind are sent to the logging file after initialization is complete.</td>
</tr>
<tr>
<td>ERROR — 1</td>
<td>Error messages indicate something that requires attention. Messages at this level could be fatal (terminating) or could indicate that an important part of the workload advising system is not working properly. This information is logged at the syslogd ERROR priority level.</td>
</tr>
<tr>
<td>WARNING — 2</td>
<td>Warning messages indicate that an error has occurred, but it is not severe enough to warrant an ERROR. Corrective action might be necessary because the Advisor or Agent might not be behaving as intended. This information is logged at the syslogd WARNING priority level.</td>
</tr>
<tr>
<td>EVENT — 4</td>
<td>Event messages are logged for things that happen periodically, like operator commands, UNIX signals, timer pops, receipt of a network message, and so on. This information is logged at the syslogd NOTICE priority level.</td>
</tr>
<tr>
<td>Logging category/Level</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>INFO — 8</strong></td>
<td>Informational messages are sent to the logging file. These messages do not require corrective action. This information is logged at the syslogd INFO priority level.</td>
</tr>
<tr>
<td><strong>MESSAGE — 16</strong></td>
<td>Message messages concern the detailed contents of message packets that are sent between the Advisor and LB, or between the Advisor and Agent. These can be used to assist debugging Advisor/LB and Advisor/Agent communications. This information is logged at the syslogd DEBUG priority level. This level is intended for IBM service use only.</td>
</tr>
<tr>
<td><strong>COLLECTION — 32</strong></td>
<td>Collection messages concern the details of collecting and manipulating the data that forms the basis of weight calculations. This information is logged at the syslogd DEBUG priority level. <strong>Restriction:</strong> COLLECTION is only used by the Agent. This level is intended for IBM service use only.</td>
</tr>
<tr>
<td><strong>DEBUG — 64</strong></td>
<td>Debug messages are intended for Development or Service and give detail that customers would not normally want. The intention of this level of message is to provide information that is useful in debugging code, logic, or timing errors. This information is logged at the syslogd DEBUG priority level. This level is intended for IBM service use only.</td>
</tr>
<tr>
<td><strong>TRACE — 128</strong></td>
<td>Trace messages are intended for Development or Service to track code processing (footprints). This information is logged at the syslogd DEBUG priority level. This level is intended for IBM service use only.</td>
</tr>
</tbody>
</table>
Chapter 8. Diagnosing problems with the automated domain name registration application (ADNR)

The automated domain name registration (ADNR) application is a function that dynamically updates name servers with information about sysplex resources in near real time. The DNS names managed by ADNR can be names that represent all instances of an application within the sysplex, names that represent a specific instance of an application within the sysplex, names that represent the entire sysplex, or names that represent individual systems within the sysplex. ADNR communicates with the z/OS Load Balancing Advisor, (specifically the Advisor application), which architecturally is a Global Workload Manager (GWM) according to the Server/Application State Protocol (SASP). The Advisor application from the z/OS Load Balancing Advisor is the only GWM with which ADNR is designed to interact. All references to a GWM in this topic refer to the Advisor application of the z/OS Load Balancing Advisor.

This topic contains information about problem diagnosis for ADNR and includes the following sections:

- Diagnostic data
- Diagnosing ADNR problems
- Debug settings

Diagnostic data

You might need to collect multiple pieces of data to accurately diagnose problems, such as the following:

- Console messages for ADNR
- syslogd log messages for ADNR (possibly including DEBUG - 64 level trace)
- Name server log data for the name servers managed by ADNR. If the managed name server resides on z/OS, this includes syslogd log messages for DNS BIND 9 server and any additional log data to individual log files. See Chapter 21, “Diagnosing name server problems,” on page 579 for information on DNS BIND 9 name server logging.
- ADNR address space dump and snap output
- SYSTCPIP CTRACE for the TCP/IP stack where ADNR and the GWM are running
- Packet traces of GWM data
- Netstat displays for the connection between ADNR and the GWM
- A listing of the zone data from the managed name server or name servers
- z/OS Load Balancing Advisor log data and displays. See Chapter 7, “Diagnosing problems with the z/OS Load Balancing Advisor,” on page 315 for information on the Load Balancing Advisor.

Tip: syslogd is the only logging facility that ADNR uses. Useful diagnostic information might be lost if syslogd is not running before you run ADNR.

ADNR triggers address space dumps when certain unexpected error conditions are encountered while communicating with a GWM. Just after connecting to the GWM, ADNR enters a negotiation phase. During negotiation, a series of
architected SASP requests are sent to the GWM; for each request, an architected SASP reply is received from the GWM. If the negotiation does not successfully complete, ADNR closes the connection, increases the logging level, establishes a new connection to the GWM, and retries the negotiation.

If the negotiation fails a second time, ADNR dumps its address space; the dump title header is ADNR Dump - Neg Failed and the logging level is restored to its original configured value and the connection is closed. Retries continue at one minute intervals using the configured logging level.

After completing the negotiation phase, the GWM might send an unsolicited SendWeights message to ADNR; the message contains information about the changed state of resources that the ADNR application registered to the GWM during negotiation. If the SendWeights message contains an architectural error, ADNR closes the connection, increases the logging level, establishes a new connection to the GWM, and completes negotiation with the GWM. If the next SendWeights message received from the GWM contains an error, ADNR dumps its address space; the dump title header is as follows:

ADNR Dump - Rcv Failed

The logging level is restored to its original configured value and the connection is closed. Retries continue at one minute intervals using the configured logging level.

These types of errors generally occur because incompatible levels of maintenance are applied to the GWM and ADNR. After being started, ADNR dumps its address space only once when these types of errors are detected. For further diagnosis, collect the ADNR log and address space dump. Review the log to determine the type of error that occurred. Review the PTF requirements of any recently installed PTFs. If you cannot correct the problem with additional maintenance or a configuration change, then contact IBM Service.

For other types of errors, both a CEEDUMP and address space snap output might be produced and written to the data sets or files that are specified by the start procedure CEEDUMP and CEESNAP DD statements, respectively.

If ADNR abnormally terminates (for example an 0Cx abend occurs), then an unformatted SYSMDUMP is written to the data set that is specified by the start procedure SYSMDUMP DD statement. If you override the Language Environment run-time option TERMTHDACT during the installation or start procedure, then the SYSMDUMP may not be produced, or a CEEDUMP may be produced instead. Therefore, you should not override the TERMTHDACT run-time option. Refer to z/OS Language Environment Programming Guide for more information about Language Environment runtime options.

In other situations, the z/OS operator might need to dump the ADNR address space manually. Refer to z/OS MVS Diagnosis: Tools and Service Aids for more information about obtaining a dump.

SYSTCPDA CTRACE (packet trace) data of the SASP protocol messages sent between ADNR and GWM may be needed. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47 for details about how to use the IP packet trace facility.

Restriction: When encrypting data, the packet trace data will be encrypted. Use MESSAGE level log messages instead.
A SYSTCPIP component trace on the TCP/IP stacks used by ADNR and its associated GWM shows data that is flowing between them. Start the trace by specifying OPTIONS=(PFS,TCP,UDP,INTERNET),JOBNAME=(server) on both stacks, where the server value is the ADNR or GWM address space (or both names separated by a comma if they are using the same stack). See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47 for more information.

You can dump the contents of the DNS zones managed by ADNR by issuing the domain information grouper (dig) command from z/OS UNIX:

dig @server zone axfr -p port -k key_file >zone_xfer.err 1>zone_xfer.out

where

server
The server IP address or host name which contains the zone managed by ADNR

zone
The zone being managed by ADNR whose contents are being dumped

port
Optionally specify the port number which the DNS server is listening on for queries

key_file
Optionally specify the key file is used to sign transactions for this zone

zone_xfer
Redirect the stdout and stderr file streams of the dig command to two distinct z/OS UNIX files.

Refer to z/OS Communications Server: IP System Administrator’s Commands for more information on the dig command.

The following Netstat command displays stacks that have affinity with ADNR:

ALLConn/-a, CONn/-c

This command is used to determine if there is an active connection between ADNR and the GWM. ALLConn/-a displays information for all TCP connections and UDP sockets, including some recently closed ones. CONn/-c displays information about each active TCP connection and UDP socket. Refer to z/OS Communications Server: IP System Administrator’s Commands, Monitoring the TCP/IP network, Netstat section for guidance using Netstat commands.

---

**Diagnosing ADNR problems**

This section includes diagnostic information about ADNR problems.

**Abends**

Messages and error-related information are usually sent to the MVS system console when an abend on ADNR occurs. Perform a dump of the error unless the symptoms already match a known problem.

**ADNR fails to initialize**

Problems with the configuration file are the most common cause for ADNR failure during initialization. This class of problems is identified by a console message. However, failure to give the ADNR load module proper APF authorization will not
result in an ADNR termination message on the console or in the syslog. In this particular case, the failure message is sent to the SYSOUT dataset. If the sample ADNR started procedure is used, then this output appears in the ADNR job log.

**ADNR not communicating with the Global Workload Manager**

ADNR communicates with only one GWM. Use the following information to diagnose why ADNR fails to communicate with the GWM.

**Restriction:** ADNR supports only the z/OS Load Balancing Advisor application as the GWM.

- Verify that ADNR is running. If it is not running, then start ADNR.
- Verify that the GWM is running. If it is not running, then start the GWM. See Chapter 7, “Diagnosing problems with the z/OS Load Balancing Advisor,” on page 315 for more on z/OS Load Balancing Advisor problems.
- Verify network connectivity exists between the GWM and ADNR.
  - Issue display commands on the GWM (by using the MODIFY command) to determine whether ADNR is connected to the GWM and has registered its group and member data. For more information on the MODIFY command see z/OS Communications Server: IP System Administrator’s Commands. If not using AT-TLS, verify that the z/OS Load Balancing Advisor’s lb_id_list statement includes the IP address on the host_connection_addr parameter in the gwm statement in the ADNR configuration file. The eventual action message, EZD1272E will persist on the console if communication with a GWM does not exist.
  - If using AT-TLS with SERVAUTH access control checks to validate connections between the Advisor and ADNR, see Chapter 29, “Diagnosing Application Transparent Transport Layer Security (AT-TLS),” on page 701. In addition, ensure that the SERVAUTH class is active. Ensure that the EZB.LBA.LBACCESS.sysname.tcpysplexgroupname resource profile is defined and that the load balancer and ADNR have READ access to it. On the system console where the Advisor is running, look for message EZD1280I which indicates that a connection attempt using AT-TLS failed. This message has specific reason codes which indicate the reason for the failure.
  - If you are using sysplex subplexing, ensure that the ADNR application does the following:
    - Has connectivity to the Advisor's subplex.
    - Is using one of the TCP/IP stacks in this subplex; when in a common INET (CINET) environment with multiple TCP/IP stacks on one MVS system, either by establishing affinity to one of the stacks in the subplex or by binding to a VIPA that is only defined on stacks that are in that subplex. See the adnrproc.sample for an example of the JCL to establish affinity.
  - Verify that network connectivity exists between ADNR and the GWM in question. Issue Netstat COnn/-c or Netstat ALLConn/-a commands on the ADNR system to see whether a connection exists between ADNR and the GWM. Correct the underlying network connectivity problem. For guidance on using Netstat commands see z/OS Communications Server: IP System Administrator’s Commands.
  - Issue a display command on ADNR to determine if there are indications that any groups and members are known to exist within the sysplex. For more information on the MODIFY command see z/OS Communications Server: IP System Administrator’s Commands.
• Check the syslogd output file where ADNR writes its log data for ERROR or WARNING messages and take the appropriate corrective action. If ERROR and WARNING level log message are not enabled, then enable them and recheck the log file later. The current ADNR debug level can be displayed by issuing the MODIFY procname,DISPLAY,DEBUG command at the MVS console. The debug level can be dynamically changed by issuing the MODIFY procname,DEBUG,LEVEL=n command at the MVS console. procname is the JCL procedure name for ADNR and n is the new debug level. For more information on the MODIFY command see z/OS Communications Server: IP System Administrator’s Commands.

Automatic restart manager (ARM) registration failure

Failure of ADNR to properly register with ARM is indicated by a warning-level message written to the log file. This log message is a result of the IXCARM call failing with the return code and reason codes indicated in the log message. Refer to z/OS MVS Programming: Sysplex Services Reference for information about interpreting the IXCARM return code and reason code.

One of the common causes of failure is the lack of a security profile. Refer to the EZARACF sample for instructions on how to add an ARM security profile for the application. Ensure that each instance of ADNR is configured to use an ARM element name that is unique within the sysplex. Use the arm_element_suffix configuration statement to specify a unique suffix for the element name.

ADNR not updating zones in a DNS server

Use the following information to determine why changes to host names are not being updated in the DNS server zone being managed by ADNR:

• Verify that ADNR is running. If it is not running then start ADNR.
• If message EZD1278E or EZD1257I has been issued, see “Diagnosing unresponsive zones” on page 328.
• Issue display commands with the ADNR MODIFY command to determine whether it is connected to the GWM and has registered its group and member data. If not using AT-TLS, verify that the z/OS Load Balancing Advisor’s lb_id_list value includes the IP address of ADNR specified with the host_connection_addr keyword of the gwm statement.
  If using AT-TLS with SERVAUTH access control checks to validate connections between the Advisor and ADNR, see Chapter 29, “Diagnosing Application Transparent Transport Layer Security (AT-TLS),” on page 701. In addition, ensure that the SERVAUTH class is active. Ensure that the EJB.LBA.LBACCESS.sysname.tcpsysplexgrouppname resource profile is defined and that the load balancer and ADNR have READ access to it. On the system console where the Advisor is running, look for message EZD1280I which indicates that a connection attempt using AT-TLS failed. This message has specific reason codes which indicate the reason for the failure.
• Check the log file for any ADNR ERROR or WARNING messages and take the appropriate corrective action. If ERROR and WARNING level log messages are not enabled, then enable them and recheck the log file later. For more information on the ADNR display commands see z/OS Communications Server: IP System Administrator’s Commands.
• Ensure that the ADNR configuration is not changed when ADNR is not active. Removing a dns statement or zone parameter while ADNR is not active causes ADNR to lose control of the information in that name server’s zones. The information in this case is considered to be orphaned.
DNS name servers managed by ADNR contain incorrect or outdated data

Use the following information to determine why zones being managed by ADNR contain incorrect or outdated information:

- Verify that ADNR is communicating with the GWM and its managed name servers.
- Verify that ADNR is able to communicate with the managed DNS name server’s zones.
- Ensure that the ADNR configuration file is not changed when ADNR is not active. Removing a dns statement or zone parameter while ADNR is not active causes ADNR to lose control of the information in that name server’s zones. The information in this case is considered orphaned and goes stale. These types of configuration file changes should be made while ADNR is active and applied by using the MODIFY procname,REFRESH command to avoid orphaned data in the name server; ADNR deletes the information in the name server that is associated with the removed dns statement or zone parameter.
- Verify that the zones in the name servers managed by ADNR have not been updated by any entity other than ADNR. This includes manual updates to the zone data files, updates from DHCP servers, or other nsupdate clients. Failure to abide by this restriction can result in lost DNS records and ADNR zone update failures.
- Verify that the update interval of the GWM is not longer than you expect. The GWM update interval dictates how frequently ADNR receives data from the GWM and consequently, how frequently the managed name servers are updated with that data. Lower the update interval on the GWM if you need the managed name servers to have data that more closely follows the actual availability status of the sysplex resources they represent. ADNR waits a certain period of time after ADNR initialization and after a dynamic update to ensure that all of the syslog data has been reported before attempting to update its managed name servers. When the GWM is the z/OS Load Balancing Advisor (Advisor) application, the Advisor’s update_interval statement determines the period of time that ADNR waits; specifically two times the update_interval received from the GWM. If the ttl keyword under a zone parameter of the ADNR DNS statement is defaulted to use the value from the GWM’s update_interval statement, then that value is used as the time-to-live value for the DNS resource records for that zone.
- If the ttl value for a zone is defaulted as described in the previous bullet, ensure the resource records in the name server for that zone reflect this value.

Diagnosing unresponsive zones

Messages EZD1278E and EZD1257I indicate when a zone is unresponsive and identify an unresponsive zone. An unresponsive zone does not accept updates or queries for information from ADNR. Unresponsive zones cause other symptoms, such as zones in a name server not being updated at all, or failure of the zone to contain up-to-date information regarding the status of resources in the sysplex. Use the following information to determine why a zone is not responsive.

- Issue display commands through the ADNR MODIFY command to determine whether the name server is responding. A name server managed by ADNR can be comprised of one or more zones. A MODIFY procname,DISPLAY,DNS,DETAIL command shows a count of the number of zones defined under a dns statement and a count of the number of zones under that dns statement that are active. When all of the zones managed by a DNS server controlled by ADNR are not responding then the DNS server is considered dead. ADNR makes periodic
probes to determine whether the zones for a dead server respond positively. See Chapter 21, “Diagnosing name server problems,” on page 579 for more on DNS BIND9 server problems. Take the appropriate corrective action.

- Verify that the DNS server being used supports RFC 2136, Dynamic Updates in the Domain Name System (DNS UPDATE). If you are using DNS BIND9 on z/OS then DNS UPDATE is supported, otherwise, review your DNS server’s documentation.

- Verify the DNS name server is running and responsive by issuing the `dig` or `nsupdate` command for the zone. If it is not running then start the DNS server. See z/OS Communications Server: IP System Administrator’s Commands Querying and administering a Domain Name System (DNS), for guidance on using the nslookup, dig and nsupdate commands. See Chapter 21, “Diagnosing name server problems,” on page 579 for more on DNS BIND9 server problems.

- Verify that network connectivity exists for the DNS server as it must be listening on the IP address and port number specified by the server parameter value (IP address) of the `dns` statement in the ADNR configuration. Unexpected loss of network connectivity for the DNS server will result in a console message and related messages in the DNS log. If the name server resides on z/OS issue Netstat Conn/-c or Netstat ALLConn/-a commands on the DNS system to see whether a listening socket exists for the DNS server in question. Correct the underlying network connectivity problem. For guidance on using Netstat commands see z/OS Communications Server: IP System Administrator’s Commands.

- Review your firewall’s log files to verify a firewall is not blocking communications between the system where ADNR resides and the name server on the port where the name server is listening for queries.

- Verify the name server being used is listening at the IP address and port that is specified by the `dns_id` parameter of the `dns` statement in the ADNR configuration file. For DNS BIND9, the IP addresses and ports the DNS server will listen on may be specified by the listen-on and listen-on-v6 DNS option statements.

- Verify that the name server IP address, optional port, zone domain suffix names, and optional Transaction Signature (TSIG) keys are correctly specified in the ADNR configuration file.

- Verify that the DNS name server specified in the ADNR `dns` statement actually manages the zone specified by the `domain_suffix` parameter of the ADNR `dns` statement and is the authoritative, primary master name server for the zone. For DNS BIND9 on the name server’s `zone` configuration statement, the `type master` option is used to specify that the server is an authoritative master. See BIND 9-based domain name system (DNS) z/OS Communications Server: IP Configuration Reference, configuration file statements, for guidance on coding the name server’s configuration. The name server managing this zone must be configured for the specified zone before ADNR can add DNS records to it. ADNR cannot dynamically create a zone in a name server. It can only add records to a zone that already exists.

- Verify that ADNR has the authority to manage the DNS resource records contained in this zone including the authority to request and receive zone transfers and perform dynamic updates. See Automated Domain Name Registration in z/OS Communications Server: IP Configuration Guide, for guidance on authorizing ADNR.

- Verify that the transaction security (TSIG) keys represented in the update and transfer keys (if specified in the ADNR configuration file) match those specified in the DNS name server for the zones ADNR is managing.
• Verify that the name server is configured with the same key names that ADNR is configured to use for the zone. Even if the name server configuration does not require a key to update or transfer an ADNR managed zone, the keys must at least be configured to the name server if ADNR is configured to use a key for that zone. If your security policies do not require you to use an update or a transfer key, they should be removed from the ADNR configuration, otherwise, the keys should be configured to the name server and used to restrict which entities are allowed to update the zone and request zone transfers.

• Verify that the name server’s working directory did not run out of disk space. ADNR makes dynamic updates to name servers. Many name server implementations require that dynamic updates be written to disk. If a name server is unable to do this, the dynamic updates from ADNR will fail causing the zone to go unresponsive. In this case, the zone emerges from the unresponsive state spontaneously, but again returns to the unresponsive state. This cycle will repeat until the storage problem on the name server host is corrected.

• Verify that the zone specified on the zone_label keyword of the ADNR dns configuration statement is not a DNSSEC signed zone. ADNR does not support the use of zones signed by DNSSEC.

• Verify that OMVS has not run out of file descriptors. See the DISPLAY OMVS command in [z/OS MVS System Commands] for information on how to make this determination.

ADNR appears to be hung

Verify that ADNR is actually hung by first issuing a MODIFY proclname,DISPLAY,GWM command. If no response is received then attempt to stop (not cancel) the application. See [z/OS MVS System Commands] STOP command subsection for more information on stopping an address space. If the application does not terminate, then the application is hung. If the hang occurred while the debug-level ADNR trace was in effect, then collect the following problem documentation and call IBM Service:

• Take an SVC dump of the ADNR address space
• Take an SVC dump of TCP/IP address space including its data spaces
• Take an SVC dump of the OMVS address space including its data spaces
• Capture the MVS console messages
• Capture the ADNR log messages written to syslogd

If the ADNR debug-level trace was not in effect at the time, then turn on the debug-level ADNR trace, reproduce the problem, collect the problem documentation, and call IBM Service.

ADNR connection to the GWM terminates unexpectedly

Check the following:

• Verify that the load balancing administrator has not shut down the GWM advising ADNR.

• Verify that TCP/IP connectivity still exists between ADNR and the GWM (for example, from the ADNR host, ping the address of the GWM). See Monitoring the TCP/IP network, Ping subsection, in [z/OS Communications Server: IP System Administrator's Commands] for further information on ping.

• Check ADNR’s log file for ERROR or WARNING messages and take the appropriate corrective action. If ERROR and WARNING debug level log messages are not enabled, then enable them, repeat the operation, and recheck to
log file again. See Modify command -- Automated Domain Name Registration in z/OS Communications Server: IP System Administrator’s Commands for further information on enabling ADNR’s debug levels.

- Check the GWM for errors.

### Debug settings

The value specified by the ADNR **debug_level** configuration option determines the ADNR logging levels. See z/OS Communications Server: IP Configuration Reference, Automated Domain Name Registration, Automated Domain Name Registration configuration file section for more on ADNR logging levels. The values may be added together to trace multiple logging categories. See z/OS Communications Server: IP System Administrator’s Commands Operator commands and system administration, Modify command, Modify command -- Automated Domain Name Registration subsection for further information on displaying and changing ADNR’s debug levels.

- When a problem occurs communicating with a DNS zone, then specifying a debug level of COLLECTION -32 causes ADNR to log the Nsupdate and Dig commands, and responses against the DNS zone. This data is the exact Nsupdate and Dig client commands and any associated responses.
- When ADNR is not able to communicate with a GWM, then specifying a debug level of MESSAGE record (16) causes ADNR to log the SASP flows.
Chapter 9. Diagnosing IKE daemon problems

This topic describes how to diagnose IKE daemon problems, and contains the following sections:

- "Overview of diagnosing IKE daemon problems"
- "Diagnosing IKE daemon problems" on page 334
- "IKE daemon debug information" on page 349
- "TCP/IP services component trace for the IKE daemon" on page 350
- "Steps for enabling the CTRACE at IKE daemon startup" on page 353

Overview of diagnosing IKE daemon problems

This section provides overview information about the z/OS Internet Key Exchange (IKE) daemon and its functions.

The IKE daemon manages dynamic IPSec tunnels. The IKE daemon is not involved in the filtering, encapsulation, or decapsulation of packets. The IKE daemon is not required for the configuration or use of IP filters.

The critical elements of IP security are security associations (SAs); specifically the information that they provide about the partners of a secure communications channel, and the cryptographic algorithms and keys to be used. The Internet Security Association Key Management protocol (ISAKMP) provides a framework for exchanging messages to automate the negotiation of security associations. The IKE protocol is a hybrid protocol that conforms to the ISAKMP framework and implements a subset of the Oakley and SKEME protocols to negotiate SAs and provide authenticated keying material for SAs in a protected manner.

The z/OS IKE daemon implements the IKE protocol to dynamically establish SAs with peer daemons that also support these protocols. In the sections that follow, a peer daemon might be referred to as an ISAKMP server or ISAKMP peer. Also, the z/OS IKE daemon might be referred to as the IKE daemon or IKED.

The IKE daemon establishes SAs within the guidelines of internet protocol security (IP security) policy. IP security policies are defined in one or more local files that are read by the Policy Agent. The IKE daemon obtains IP security policies from the Policy Agent using the Policy API (PAPI). Refer to the z/OS Communications Server IP Configuration Guide for more information about configuring and starting Policy Agent, as well as defining policies.

The IKE daemon establishes and installs the following types of SAs:

- An ISAKMP SA, or phase 1 SA; its purpose is to protect communications between ISAKMP peers
- An IPSec SA, or phase 2 SA; its purpose is to protect internet protocol (IP) traffic originating from, destined to, or routed by the z/OS TCP/IP stack

The IKE daemon installs three primary types of information in the TCP/IP stack:
IPSec (phase 2) SAs
The IKE daemon installs established IPSec SAs in the TCP/IP stack. On z/OS, the IPSec SA information that is installed in the TCP/IP stack is referred to as a dynamic tunnel.

Dynamic IP filters
When the IKE daemon installs a dynamic tunnel in the TCP/IP stack, it also installs dynamic IP filters that define what IP traffic can be sent or received through the tunnel. The IKE daemon installs one inbound and one outbound dynamic IP filter with each dynamic tunnel.

ISAKMP (phase 1) SAs
For Sysplex-Wide Security Association (SWSA) support, the IKE daemon also installs ISAKMP SA information in the TCP/IP stack. The IKE daemon only installs ISAKMP SAs in a stack that is configured for SWSA support using the DVIPSEC keyword. Refer to the z/OS Communications Server: IP Configuration Guide for more information about SWSA support. For information about diagnosing SWSA problems, see "Steps for diagnosing sysplex-wide security association (SWSA) problems" on page 393.

Diagnosing IKE daemon problems
This section contains information helpful in diagnosing IKE daemon problems.

Initialization problems
When IKE successfully initializes, message EZD1046I is issued. If the IKE daemon fails to initialize, message EZD1045I or EZD1049I is issued. Common initialization problems include:

- IKE started from a user ID without superuser authority. IKE must be started from a superuser. The symptom for this problem is the following message:
  EZD1045I IKE initialization error : IKE is not running in superuser state

  To correct this problem, restart the IKE daemon from a user ID that has superuser authority.

- The IKE daemon load module is not APF-authorized. The IKE daemon load module must be APF-authorized. The symptom for this problem is the following message: EZD0986I IKE is not APF authorized.

  To correct this problem, ensure that the IKE daemon load module resides in an APF-authorized library, and then restart the IKE daemon.

- IKE cannot create the /var/ike or /var/sock directories. The IKE daemon attempts to create the /var/ike and /var/sock directories at initialization. If IKE cannot create either of these directories, then initialization fails. If this problem has occurred, one of the following messages is issued:
  EZD1045I IKE initialization error : mkdir /var/ike failed
  EZD1045I IKE initialization error : mkdir /var/sock failed

  To correct this problem, ensure that the /var directory is mounted as read/write. If the /var directory is mounted as read/write and the problem still occurs, contact IBM for additional assistance.

Problems establishing security associations
This section describes problems in establishing security associations and offers guidance on what steps to take to overcome these problems.
Table 17. Establishing security associations problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cannot send or receive packets on UDP ports 500 or 4500</td>
<td>Message EZD1065I was issued. When filter logging is active, message EZD0815I is issued, showing packets to UDP port 500 or UDP port 4500 that were denied.</td>
<td>The IKE daemon communicates using UDP ports 500 and 4500 for IPv4. The IKE daemon communicates using UDP port 500 for IPv6. See [Steps for verifying IP routing to a destination when not using policy-based routing (PBR)](z/OS Communications Server: IP System Administrator's Commands) on page 32 to verify that the IKE daemon is running and bound to ports 500 and 4500. A filter rule must be configured to permit inbound UDP traffic from any source port to destination ports 500 and 4500. A filter rule must be configured to permit outbound UDP traffic from source ports 500 and 4500 to any destination port. Use the <code>ipsec -f display</code> command to confirm there is a filter rule installed in the stack that permits receiving traffic from any source UDP port to destination UDP ports 500 and 4500. Also confirm that there is a filter rule that permits receiving traffic from source UDP ports 500 and 4500 to any destination port. Activate filter logging for these rules so that you can observe packets sent on source ports 500 and 4500 and received on destination ports 500 and 4500 in the syslog. For information about the <code>ipsec</code> command, refer to [z/OS Communications Server: IP Configuration Guide](z/OS Communications Server: IP Configuration Guide) for general information about configuring IP filters.</td>
</tr>
<tr>
<td>Problem</td>
<td>Symptom</td>
<td>Cause/response</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Pre-shared key mismatch</td>
<td>Message EZD0965I was issued.</td>
<td>If IKE is using pre-shared key mode authentication and it cannot interpret a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>decrypted message that it has received, then message EZD0965I is issued,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>indicating a likely pre-shared key mismatch. In main mode, the responder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gets the message upon receipt of message 5. In aggressive mode, the initiator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gets the message upon receipt of message 2. EZD0965I can also be issued if</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IKE receives a corrupted message even though the pre-shared keys match. If the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>remote peer cannot decrypt the message that was sent by IKE because of a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pre-shared key mismatch, the local symptom is that IKE retransmits the first</td>
</tr>
<tr>
<td></td>
<td></td>
<td>encrypted message of the exchange. Review the pre-shared key configuration on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the local and remote system and ensure that the keys match.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tip:</strong> The keys might be represented differently (for example, ASCII or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EBCDIC) on the local and remote system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failure accessing local certificate</td>
<td>One of the following messages was issued:</td>
<td>For the IKE daemon to support RSA signature mode authentication using a local</td>
</tr>
<tr>
<td>repository</td>
<td>• EZD0990I</td>
<td>certificate repository, the daemon must be able to access certificates on the</td>
</tr>
<tr>
<td></td>
<td>• EZD1030I</td>
<td>SAF key ring. IKE issues message EZD0990I to indicate that RSA signature mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is supported or EZD1030I if RSA signature mode is not supported for a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>given stack using the key ring. Refer to the messages to determine the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>appropriate response. The key ring is specified on the KeyRing parameter in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the IkeConfig statement. When configuring with the IBM Configuration Assistant</td>
</tr>
<tr>
<td></td>
<td></td>
<td>for z/OS Communications Server GUI, the key ring is specified on the key ring</td>
</tr>
<tr>
<td></td>
<td></td>
<td>database field on the IPSec: IKE Daemon Settings panel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem</td>
<td>Symptom</td>
<td>Cause/response</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Failure accessing the network security services (NSS) server</td>
<td>One of the following messages was issued:</td>
<td>For the IKE daemon to support RSA signature mode authentication using IPsec certificate services, it must be able to connect to a network security server. IKE issues the following messages:</td>
</tr>
<tr>
<td></td>
<td>- EZD1136I</td>
<td>- EZD1136I to indicate that it has connected to a network security server.</td>
</tr>
<tr>
<td></td>
<td>- EZD1137I</td>
<td>- EZD1137I to indicate that it is not connected to a network security server.</td>
</tr>
<tr>
<td></td>
<td>- EZD1138I</td>
<td>- EZD1138I to indicate that it is connecting to a network security server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The network security services server is specified on the NetworkSecurityServer and NetworkSecurityServerBackup parameters on the IkeConfig statement. When configuring with the IBM Configuration Assistant for z/OS Communications Server GUI, the network security server is specified on the server setting in the NSS perspective.</td>
</tr>
<tr>
<td>RSA signature authentication failure - missing certificate in the local certificate repository</td>
<td>Message EZD1037I was issued.</td>
<td>Check the syslog to determine whether message EZD1037I was issued. If the IKE daemon cannot locate a certificate that is needed for RSA signature mode authentication, message EZD1037I is issued.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Display the certificates on the SAF key ring.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Ensure that all the certificates on the key ring that are to be used by the IKE daemon include a digital signature.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- If you are using RACF, make sure that the trust status of the certificates is TRUST or HIGHTRUST.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Use the IKE daemon IkeSyslogLevel 64 to display the contents of the IKE daemon’s certificate caches and ensure that the desired certificates are included in the caches.</td>
</tr>
<tr>
<td>RSA signature authentication failure because of identity mismatch</td>
<td>One of the following messages was issued:</td>
<td>Check the syslog to determine whether message EZD0981I or EZD1075I was issued. If the identity that is contained within a received certificate does not match the identity that is configured on the RemoteSecurityEndPoint statement, message EZD0981I is issued. If the peer detects such a mismatch, it might send an &quot;Invalid ID information&quot; notification. If IKE receives such a notification, message EZD1075I is issued. Refer to the messages to determine the appropriate response.</td>
</tr>
</tbody>
</table>
### Table 17. Establishing security associations problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA signature authentication failure</td>
<td>One of the following messages was issued:</td>
<td>Check the syslog to determine whether message EZD0902I or EZD0903I was issued. If the certificate that is received from a peer cannot be verified, message EZD0902I is issued. If the certificate that is received from the peer cannot be authenticated, message EZD0903I is issued. Refer to the messages to determine the appropriate response. Activate IkeSyslogLevel 64 to get additional diagnostic information that relates to RSA signature mode authentication. The IKE daemon syslog level is set in the IkeSyslogLevel parameter in the IkeConfig statement. When configuring with the IBM Configuration Assistant for z/OS Communications Server GUI, the IKE Daemon Syslog settings are accessed from the IPSec: IKE Daemon Settings panel. IKE maintains a separate cache for Certificate Authority (CA) certificates and security endpoint certificates. When IkeSyslogLevel 64 is active, the contents of the certificate caches are displayed when they are built or rebuilt. Refer to <a href="https://www.ibm.com/support/documentation/en/zos-ws10">z/OS Communications Server: IP Configuration Reference</a> for information about setting the IkeSyslogLevel, or see the online help in the IBM Configuration Assistant for z/OS Communications Server. <strong>Tip:</strong> The name of the key ring is case sensitive.</td>
</tr>
<tr>
<td>RSA signature authentication failure -</td>
<td>Message EZD1139I was issued.</td>
<td>Check the syslog to determine whether message EZD1139I was issued. The EZD1139I message will be issued if the network security server failed to locate a certificate, could not verify a digital signature, or could not create a digital signature for RSA signature mode authentication. Refer to the messages to determine the appropriate response.</td>
</tr>
<tr>
<td>IPsec certificate services failure</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

338  
z/OS V1R11.0 Comm Svr: IP Diagnosis Guide
Table 17. Establishing security associations problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
</table>
| Failure to locate phase 1 policy | Message EZD0917I was issued.         | In order for IKE to establish a phase 1 SA, it must first locate an applicable phase 1 policy. KeyExchangeRules encapsulate phase 1 policy for IKE. KeyExchangeRules are classified according to a 4-tuple that is comprised of LocalSecurityEndpoint Location, LocalSecurityEndpoint Identity, RemoteSecurityEndpoint Location, and RemoteSecurityEndpoint Identity. When IKE needs to locate a KeyExchangeRule statement, it performs a search of the configured KeyExchangeRules statements, supplying specific values or Any for each parameter of the classification 4-tuple. When configuring with the IBM Configuration Assistant for z/OS Communications Server the following are configured in each Connectivity Rule:
   • Local Security End Point Location
   • Local Security End Point Identity
   • Remote Security End Point Location
   • Remote Security End Point Identity
   • Key Exchange Settings

   It is also possible in the GUI to configure a single Local Security End Point Location and Identity for an entire TCP/IP stack.

   If IKE fails to locate an applicable KeyExchangeRule statement, message EZD0917I is issued that lists the classification 4-tuple. Use the `pasearch -v k -r` command to review the configured KeyExchangeRules statement. If there is no KeyExchangeRule statement that corresponds to the classification 4-tuple that is given on the EZD0917I message, configure a new KeyExchangeRule statement as needed. Refer to the messages for EZD0917I for more information. |
<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1 policy mismatch</td>
<td>Message EZD1093I or EZD1075I was issued.</td>
<td>The ISAKMP initiator and responder must agree on phase 1 policy in order to successfully complete negotiation of a phase 1 security association. If the IKE daemon rejects the phase 1 policy that is proposed by an ISAKMP peer, it issues message EZD1021I, which indicates the KeyExchangeRule and KeyExchangeAction statements that were in effect when the mismatch occurred. Message EZD1093I is issued, which indicates why the mismatch occurred. When configuring with the IBM Configuration Assistant for z/OS Communications Server, the Key Exchange Settings are set in each Connectivity Rule. If the IKE daemon proposes phase 1 policy that the ISAKMP peer rejects, the ISAKMP peer should send a notification message to the IKE daemon. If the IKE daemon receives such a notification, it issues message EZD1075I. For more information, see the EZD1075I message documentation in z/OS Communications Server: IP Messages Volume 2 (EZB, EZD). If the peer is a z/OS IKE daemon, it issues the EZD1021I and EZD1093I messages as described above. If the peer is not a z/OS IKE daemon, consult the documentation for the ISAKMP peer product to determine why it rejected the proposal. In the case of a mismatch, a No proposal chosen notification is expected from the peer.</td>
</tr>
<tr>
<td>Problem</td>
<td>Symptom</td>
<td>Cause/response</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Failure to locate phase 2 policy</td>
<td>Message EZD1024I was issued</td>
<td>In order for IKE to establish a phase 2 SA, it must first locate an applicable phase 2 policy. Phase 2 policy for the IKE daemon is comprised of IpFilterRule and IpDynVpnAction statements. The first step in locating a phase 2 policy for the IKE daemon is to locate an IpFilterRule statement that matches the traffic to be protected and includes a reference to an IpDynVpnAction statement. If IKE cannot find an applicable IpFilterRule statement, message EZD1024I is issued, which indicates the traffic that was to be protected. When configuring with the IBM Configuration Assistant for z/OS Communications Server, the Connectivity Rules are used to locate the phase 2 policies. The Connectivity Rules contain the local and remote data end points, and which type of traffic is protected by Security Levels implementing dynamic tunnels. Refer to the messages to determine the appropriate response. See “Steps for verifying IP security and defensive filter operation” on page 732, supplying the IP traffic characteristics identified on the EZD1024I message.</td>
</tr>
</tbody>
</table>
Table 17. Establishing security associations problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 2 policy mismatch</td>
<td>Message EZD1022I, EZD1093I, or EZD1075I was issued.</td>
</tr>
</tbody>
</table>

The ISAKMP initiator and responder must agree on phase 2 policy in order to successfully complete negotiation of a phase 2 security association. If the IKE daemon rejects the phase 2 policy that is proposed by an ISAKMP peer, it issues message EZD1022I, which indicates the IpFilterRule and IpDynVpnAction statements that were applied. When configuring with the IBM Configuration Assistant for z/OS Communications Server, the Connectivity Rules are used to locate the phase 2 policies. The Connectivity Rules contain the local and remote data end points, and which type of traffic is protected by Security Levels implementing dynamic tunnels. Message EZD1093I is issued indicating why the mismatch occurred.

Check the syslog to determine whether message EZD1075I was issued. If the IKE daemon proposes a phase 2 policy that the ISAKMP peer rejects, the ISAKMP peer should send a notification message to the IKE daemon. If the IKE daemon receives such a notification, it issues message EZD1075I. Review the diagnostic data at the ISAKMP peer to determine why the peer rejected the proposal. See the EZD1075I message documentation for more information.

In the case of a mismatch, a No proposal chosen notification is expected from the peer.
## Table 17. Establishing security associations problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES encryption/decryption failure</td>
<td>Message EZD0918 or EZD1109 was issued.</td>
<td>If IKE is using AES for phase1 or phase2 encryption and decryption, it calls Integrated Cryptographic Service Facility (ICSF) to do the actual cryptography. If ICSF has not been started the cryptography cannot be performed, and IKE cannot encrypt or decrypt messages using AES. This can happen any time during an informational exchange, any time during a phase 2 exchange, in message 5 of main mode if acting as the initiator, or in message 6 of main mode if acting as the responder. The return and reason codes for an ICSF failure are output in message EZD0918 (encryption) or message EZD1109 (decryption). If the return code is C(12) and the reason code is 0, this normally means that ICSF has not been started and therefore cannot perform the necessary cryptography. Ensure that ICSF is started so that the IKE daemon can perform AES cryptography. If the return code is C(12) and the reason code is 8, this normally means that the installed version of ICSF does not support AES. The Security Level Feature of ICSF is required (FMID HCR7706 or higher) for AES support.</td>
</tr>
</tbody>
</table>

### Network security services client problems:

IKED can be configured to request network security services (NSS) from an NSS server. The following table lists common problems when IKED, running as an NSS client, is unable to obtain services from the NSS server.

## Table 18. Common problems when IKED, running as an NSS client, is unable to obtain services from the NSS server

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL is not properly configured for IKED, running as an NSS client, to connect to the NSS server.</td>
<td>When AT-TLS is not enabled or is misconfigured on the TCP/IP stack used by IKED or the NSS server, IKED issues message EZD1149I indicating that the connection is not secure.</td>
<td>AT-TLS must be enabled on both the client and server stacks with the TCPCONFIG TTLS statement in the TCP/IP profile. AT-TLS policies must be defined for both the client and the server to secure the connection. Refer to &quot;Define AT-TLS policy to protect communication with an NSS server&quot; in z/OS Communications Server: IP Configuration Guide. If AT-TLS is enabled and the definitions are configured on the client and server stacks but EZD1149I is still displayed then refer to Chapter 29, &quot;Diagnosing Application Transparent Transport Layer Security (AT-TLS),&quot; on page 701.</td>
</tr>
</tbody>
</table>
Table 18. Common problems when IKED, running as an NSS client, is unable to obtain services from the NSS server (continued)

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
<tbody>
<tr>
<td>The userid used for the IKED connection to the NSS server has insufficient authority to connect.</td>
<td>IKED issues message EZD1139I with reason code NSSRsnUserAuthentication. For example: EZD1139I Request type NSS_ConnectClientReqToSrv with correlator ID 00000000000000040000000000000000 for stack TCPSCS2 failed - return code EACCESS reason code NSSRsnUserAuthentication</td>
<td>The IKED connection to the NSS server requires configuration of a valid userid and password or passticket on the NssStackConfig statement in the IKED configuration file.</td>
</tr>
<tr>
<td>The userid used for the IKED connection to the NSS server has insufficient authority to access services requested.</td>
<td>IKED issues messages indicating which requested services are not available. For example: • EZD1145I The network security certificate service is not available for stack TCPSCS2 • EZD1147I The network security remote management service is not available for stack TCPSCS2</td>
<td>SAF resource permissions are required to access network security services: • EZB.NSS.sysname.clientname.IPSEC.CERT • EZB.NSS.sysname.clientname.IPSEC.NETMGMT These resources must be defined on the NSS server system and the userid configured on the NssStackConfig statement in the IKED configuration file must be permitted read access to them.</td>
</tr>
<tr>
<td>IKED fails to retrieve certificates from the NSS server.</td>
<td>IKED syslog daemon traces may show that no cache entries were received from the NSS server. For example: IKE: Initializing CA Cache with 0 entries for stack TCPSCS2 Dynamic tunnel negotiations using RSA signature mode fail.</td>
<td>SAF resource permissions are required to access certificates from the NSS server: EZB.NSSCERT.sysname.mappedlabelname.CERTAUTH EZB.NSSCERT.sysname.mappedlabelname.HOST These resources must be defined on the NSS server system and the userid configured on the NssStackConfig statement in the IKED configuration file must be permitted read access to it. Refer to “Steps for authorizing resources for NSS” in z/OS Communications Server: IP Configuration Guide.</td>
</tr>
<tr>
<td>IKED does not attempt to connect to the NSS server for a given stack.</td>
<td>IKED does not issue message EZD1138I for the given stack.</td>
<td>A valid NssStackConfig statement is required for each stack to utilize NSS. Refer to IKE daemon in z/OS Communications Server: IP Configuration Reference for information about configuring the NssStackConfig statement.</td>
</tr>
</tbody>
</table>

**NAT traversal considerations**

- **NAT traversal support must be enabled.**

  By default, NAT traversal support on z/OS is disabled. To enable NAT traversal support do one of the following:
  - Specify a value of *Yes* for the AllowNat parameter of the KeyExchangeAction statement utilized when negotiating with a remote security endpoint that you want to perform NAT traversal with.
  - Specify a value of *Yes* for the AllowNat parameter of the KeyExchangePolicy statement. Verify that the AllowNat parameter is not specified as *No* on the KeyExchangeAction statement utilized when negotiating with the remote security endpoint that you want to perform NAT traversal with.
Use care when using the latter method. The AllowNat parameter specified on the KeyExchangePolicy statement becomes the default AllowNat setting for KeyExchangeAction statements that do not specify the AllowNat parameter. Refer to [z/OS Communications Server: IP Configuration Reference] for more details concerning the AllowNat parameter. When configuring with the IBM Configuration Assistant for z/OS Communications Server, you configure whether to allow NAT traversal processing on the Stack Level Settings panel. This setting can be overridden in each Connectivity Rule.

The AllowNat field contained in the output of the `ipsec -k display` command can be utilized to determine whether NAT Traversal support was enabled for a phase 1 negotiation.

Changes made to the AllowNat parameter do not impact existing ISAKMP security associations. Existing ISAKMP security associations must be refreshed before any changes to the AllowNat are honored. There are no configuration options to enable or disable NAT traversal for an IPSec security association. The state of NAT traversal for an IPSec security association is determined by the ISAKMP security association used when negotiating the IPSec security association.

- The remote security endpoint must support an acceptable version of NAT traversal. z/OS provides limited support for the following levels of NAT Traversal:
  - draft-ietf-ipsec-nat-t-ike-02
  - draft-ietf-ipsec-nat-t-ike-03
  - RFC 3947
  - RFC 3947 with z/OS-only extensions

  The remote security endpoint must support one of these levels of NAT traversal.

You can use the NatLevel field contained in the output of the `ipsec -k display` command to determine what level of NAT traversal support was utilized during a phase 1 negotiation. If the NatLevel is None, verify that NAT traversal support is enabled when negotiating with this remote security endpoint. If NAT traversal support was enabled, then the remote security endpoint does not provide an acceptable level of NAT traversal support.

- z/OS does not support NAT Traversal for IPv6 traffic
- z/OS cannot act as a gateway when traversing a NAT.

The z/OS IKE daemon does not support acting in the gateway role when traversing a NAT. The z/OS IKE daemon is acting as a gateway when the local data endpoint of an IPSec security association is not the same as the IP address utilized as the local IP address of the protecting ISAKMP security association. Message EZD1089I is issued when z/OS is acting as a gateway while traversing a NAT.

When a NAT is detected between z/OS IKE and a remote security endpoint, all IPSec security associations negotiated with that remote security endpoint must end in the local z/OS box. Specifically, the local data endpoint of any IPSec security association negotiated when traversing a NAT must be the IP address utilized as the local IP address of the protecting ISAKMP security association. If z/OS is behind a NAT this could be its private address or the public IP address provided by the NAT.

You can use the LocalEndpoint field contained in the `ipsec -k display` command utilized to determine the local private IP address of the protecting ISAKMP security association. The local public IP address of the protecting ISAKMP security association is assigned by the NAT box in front of z/OS.
You can use the NATInFrntLclScEndPnt and NATInFrntRmtScEndPnt fields contained in the output of the `ipsec -k display` command to determine whether a NAT was detected between the IKE daemon and the remote security gateway.

- **z/OS cannot act as an initiator to a security gateway.**
  The z/OS IKE daemon cannot act as the initiator of a phase 2 negotiation for a new IPSec security association when traversing a NAT and the remote security endpoint is acting as security gateway. Messages EZD1090I or EZD1057I are issued in this case. The z/OS IKE daemon can act as the initiator of subsequent refreshes of an existing IPSec security association with a remote security endpoint that is acting as a security gateway.

- **z/OS cannot act as an initiator to a remote security endpoint located behind a NAT device performing network address port translation (NAPT).**
  The z/OS IKE daemon does not support initiating the first security association to a remote security endpoint when a NAT has translated the remote security endpoint’s port (that is, when IKE detects the existence of a NAPT in front of the remote security endpoint). If this condition is detected during a phase 1 or phase 2 negotiation, the negotiation is terminated.

- **z/OS utilizes only IPv4 identities during phase 2**
  During a phase 2 negotiation for a new IPSec security association, the z/OS IKE daemon uses IPv4 ID types to identify the traffic pattern to be protected by the new IPSec security association. When traversing a NAT, other IKE implementations might require the traffic pattern to be specified using a non-IPv4 ID type. The z/OS IKE daemon is not able to act as the initiator of a phase 2 negotiation with such an implementation when creating a new IPSec security association. The z/OS IKE daemon can act as the initiator of subsequent refreshes of an existing IPSec security association with a remote security endpoint utilizing such an IKE implementation.

  When the z/OS IKE daemon acts as the initiator of a phase 2 negotiation to create a new IPSec security association and the remote security endpoint is using an implementation that requires a non-IPv4 ID type, the remote security endpoint rejects the proposal. Some implementations might send an informational notification in this case. The informational notification indicates that the proposal was rejected and why. If an informational notification is received, the z/OS daemon issues message EZD1075I.

- **Interoperability considerations when z/OS initiates a phase 2 negotiation to a non-z/OS peer for a host-to-host tunnel that traverses a NAT**
  - **Host-to-host dynamic tunnel protecting all ports, tunnel mode**
    When z/OS initiates a phase 2 negotiation for a new host-to-host dynamic tunnel that protects all ports and all protocols, z/OS allows the default traffic pattern which is the IP addresses of the local and remote security endpoints. This means that IKE and its peer view the traffic pattern differently, z/OS views the traffic pattern as its private IP address and the peer’s public IP address. The peer views the traffic pattern as z/OS’s public address and the peer’s private address.

    A host-to-host dynamic tunnel uses either the transport or tunnel mode of encapsulation. When z/OS initiates a phase 2 negotiation to a non-z/OS peer for a new host-to-host dynamic tunnel that protects all ports and all protocols it is possible that the negotiation succeeds, but it produces an SA that cannot
be used to send traffic. This is partially because data protected using tunnel mode SAs have two IP headers. Because both peers have a different view of the traffic pattern, they might not agree on the contents of the inner-most IP header. When both the local and remote peer are z/OS, the SA negotiation should be successful and produce an SA that can be used to send traffic.

- Host-to-host dynamic tunnel protecting specific ports or protocols
The traffic endpoints cannot use the default pattern when negotiating a new host-to-host dynamic tunnel that protects a specific port or protocol. RFC 3947 does not discuss how traffic patterns should be defined when one or more NAI’s are being traversed. When z/OS initiates a phase 2 negotiation for a new host-to-host dynamic tunnel that protects a specific port or protocol, it defines the traffic pattern using z/OS private addresses as the local endpoint, if z/OS is behind a NAT, and the peer’s public address as the remote endpoint. In this case, the negotiation might fail with a non-z/OS peer, depending on the NAT traversal support of the non-z/OS peer. The negotiation should be successful with a z/OS peer.

To help identify configurations where there are potential interoperability concerns, three informational messages have been defined. When z/OS initiates a phase 2 negotiation for a UDP encapsulated tunnel mode SA with a non-z/OS peer, message EZD1104I or EZD1105I is issued. When z/OS initiates a phase 2 negotiation for a UDP encapsulated tunnel or transport mode SA for a specific port, protocol, or both, message EZD1107I is issued. In all cases, the negotiation continues.

- SWSA implications
During VIPA takeover or giveback processing, the IKE daemon attempts to create security associations that existed on the stack that owned the security association prior to the takeover or giveback. These security associations appear as new security associations on the new owning stack.

The z/OS IKE daemon cannot initiate the creation of new IPSec security associations when the peer is acting as a gateway, or when the peer is behind a NAPT, or when the peer expects a non-IPv4 identity during a quick mode exchange; however, it can act as a responder in these cases. When a VIPA takeover or giveback occurs the IKE daemon does not attempt to re-establish such phase 2 security associations.

There are also cases when the results might be unpredictable when the z/OS IKE daemon initiates a new host-to-host SA negotiation to a non-z/OS peer. These cases include:

- z/OS IKE initiates a new host-to-host UDP encapsulated tunnel mode SA to a non-z/OS peer
- z/OS IKE initiates a new host-to-host UDP encapsulated SA for a specific port, protocol, or both to a non-z/OS peer

It is expected that IKE can always act as a responder in these cases. If such SAs exist when a VIPA takeover or giveback occurs, the IKE daemon attempts to re-establish these security associations. The results of these attempts are unpredictable. This can result in a disruption of traffic until new SAs are created by the remote security endpoint. The IKE daemon still sends delete notifications informing the remote security endpoint that the security associations are no longer valid.

- Remapping of a remote security endpoint’s address
When a remote security endpoint is behind a NAT, the NAT maps the private IP address of the remote security endpoint to a public IP address. This mapping
could expire as a result of inactivity or a new mapping could be created due to a reboot of the NAT device. In such cases, the public IP address of the remote security endpoint might change.

In the cases where NAT performs port translation (NAPT), the IP address or port or both might change.

If the IKE daemon or stack detects such a change while there are one or more security associations with that remote security endpoint, the IKE daemon attempts to verify the new IP address and port pair. It does this by initiating the creation of a new ISAKMP security association using the remote security endpoint’s new IP address and port pair. Message EZD1086I is issued when this negotiation starts. If this negotiation is successful, the IKE daemon issues message EZD1087I and all ISAKMP and IPSec security associations with that remote security endpoint using the old address are deleted.

- **NAT keepalive timer**
  
  When a z/OS is behind a NAT, the NAT maps its private IP addresses to public IP addresses. A static NAT mapping does not expire. A dynamic NAT mapping can expire as a result of inactivity. In order to prevent the expiration of this mapping, the stack occasionally sends messages known as NAT keepalive messages. If these messages are not sent frequently enough, the NAT device could expire the mapping of any z/OS private IP addresses to public IP addresses. Such a remapping could be disruptive to existing IPSec traffic.

  The frequency of message transmission is defined by the NatKeepAliveInterval value on the KeyExchangePolicy statement. Refer to [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com/support/docview.wss?uid=swg27005085) for more details. When configuring with the IBM Configuration Assistant for z/OS Communications Server, the NAT keep alive interval is specified on the Stack Level Settings panel.

  A NAT keep alive is a 1-byte UDP message sent to a remote security endpoint using the UDP encapsulation ports. The sent byte is set to x'FF'. Figure 29 shows a NAT keep alive message:

  ![NAT keep alive message](image)

  **Figure 29. NAT keep alive message**

- **Multiple remote security endpoints sharing the same ISAKMP identity**
  
  During a phase 1 negotiation the remote security endpoint sends its identity in an ID payload. The IKE daemon can manage multiple remote security endpoints using the same ID when those endpoints are not behind a NAT. However, when a remote security endpoint is behind a NAT it must use a unique ISAKMP identity. If a second remote security endpoint behind a NAT attempts to use an ISAKMP identity already in use by another remote security endpoint behind a NAT, the IKE daemon detects this as a remapping of the first remote security endpoint’s IP address.

  When multiple remote security endpoints behind a NAT share the same ISAKMP identity, messages EZD1086I and EZD1087I might be repeatedly issued.

- **Responding to phase 1 main mode SA negotiations with multiple remote security endpoints behind a NAPT**
  
  When acting as a responder in main mode SA negotiations, the z/OS IKE daemon must agree to key exchange parameters before the remote security endpoint identity is known. The key exchange policy is searched to match on a
KeyExchangeRule based upon the IP addresses of the local and remote security endpoints. Different remote security endpoints located behind an NAPT might use the same public IP address. This can cause a policy mismatch if the KeyExchangeRule settings for those remote security endpoints do not match. A policy mismatch will cause EZD1093I to be issued to syslog.

To prevent this situation, do the following:

Configure a single KeyExchangeRule to represent all the remote security endpoints behind the NAPT device. The remote security endpoints, represented by the same public address, must use the same security parameters. The remote security endpoints’ policy must be configured with the same security parameters as well.

**Abends**

Messages and error-related information should be sent to the system console when an abend occurs during IKE daemon processing. A dump of the error is needed unless the symptoms match a known problem. System dumps of IKE include Language Environment data. The Language Environment IPCS verbexit LEDATA can be used to format this information. See [z/OS Language Environment Debugging Guide](https://www.ibm.com/support/docview#!/en/IT660-122_1.2.0/IT660-122/IT660-122.doc) for more information. The following is a sample IPCS verbexit LEDATA command:

```
verbx ledata 'asid(68) tcb(007E5E88) cedump nthreads(*)'
```

**Tip:** In this example, the IKE asid is 0x68 and the address of the abended IKE TCB is 0x007E5E88.

---

**IKE daemon debug information**

Additional IKE daemon debug information can be sent to the syslog using the IkeSyslogLevel and PagentSyslogLevel parameters in the IKE configuration file.

**Obtaining syslog debug information for the IKE daemon**

The IkeSyslogLevel parameter in the IKE configuration file controls the level of IKE internal debug information that is sent to syslog. When configuring with the IBM Configuration Assistant for z/OS Communications Server, use the IKE Daemon Settings panel to configure the level of IKE internal debug information that is sent to syslog.

The IKE syslog level value should be set above 1 only when diagnosing a problem; levels above 1 impact IKE performance. Level 8 and Level 16 have the greatest performance impact because they affect processing on each UDP datagram IKE sends and receives.

IKE Syslog level values can be combined. Refer to the [z/OS Communications Server IP Configuration Reference](https://www.ibm.com/support/docview/doc/scga00625) or the IBM Configuration Assistant for z/OS Communications Server’s online help for more information.

**Obtaining debug information using PagentSyslogLevel**

IKE uses the Policy API (PAPI) to communicate with the Policy Agent and manipulate policy information it has obtained from the Policy Agent. The PagentSyslogLevel parameter in the IKE configuration file controls the level of debug information that is sent to syslog when IKE uses PAPI. When configuring with the IBM Configuration Assistant for z/OS Communications Server, the Policy Agent Syslog events are configured from the IKE Daemon Settings panel. The Policy Agent Syslog level value should be set above 0 only at the direction of IBM.
TCP/IP services component trace for the IKE daemon

z/OS CS provides component trace support for the IKE daemon. This section describes how to specify IKE daemon trace and formatting options. For short descriptions of other tracing procedures, such as displaying trace status, see Chapter 5, “TCP/IP services traces and IPCS support,” on page 47.

For detailed information, refer to the following information:

- z/OS MVS Diagnosis: Tools and Service Aids for information about component trace procedures.
- z/OS MVS Initialization and Tuning Reference for information about the component trace SYS1.PARMLIB member.
- z/OS MVS System Commands for information about commands.
- z/OS MVS Programming: Authorized Assembler Services Guide for procedures and return codes for component trace macros.

Using CTRACE

You can specify component trace options at TCP/IP initialization or after TCP/IP has initialized.

Table 19 lists the IKE daemon trace options.

<table>
<thead>
<tr>
<th>Trace Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Trace all types of records. This option slows performance.</td>
</tr>
<tr>
<td>MINIMUM</td>
<td>Trace the IKE daemon’s minimum level of tracing.</td>
</tr>
<tr>
<td>INIT</td>
<td>Trace IKE daemon initialization information.</td>
</tr>
<tr>
<td>TERM</td>
<td>Trace IKE daemon termination information.</td>
</tr>
<tr>
<td>EXCEPT</td>
<td>Trace IKE daemon exception information.</td>
</tr>
<tr>
<td>CONFIG</td>
<td>Trace IKE daemon configuration information.</td>
</tr>
<tr>
<td>WORKUNIT</td>
<td>Trace IKE workunit information.</td>
</tr>
<tr>
<td>SERIAL</td>
<td>Trace IKE serialization information.</td>
</tr>
<tr>
<td>IKE</td>
<td>Trace IKE protocol information.</td>
</tr>
<tr>
<td>CRYPTO</td>
<td>Trace IKE cryptographic information.</td>
</tr>
<tr>
<td>OPMSCGS</td>
<td>Trace IKE operator messages.</td>
</tr>
<tr>
<td>LOGMSGS</td>
<td>Trace IKE syslog messages.</td>
</tr>
<tr>
<td>MSGQ</td>
<td>Trace IKE message queue information.</td>
</tr>
<tr>
<td>TIMER</td>
<td>Trace IKE timer information.</td>
</tr>
<tr>
<td>SOCKETS</td>
<td>Trace IKE socket information.</td>
</tr>
<tr>
<td>IOCTL</td>
<td>Trace IKE IOCTL call information.</td>
</tr>
<tr>
<td>REQUESTS</td>
<td>Trace IKE request information.</td>
</tr>
</tbody>
</table>
Table 19. IKE daemon trace options (continued)

<table>
<thead>
<tr>
<th>Trace Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOW</td>
<td>Trace IKE code flow information.</td>
</tr>
<tr>
<td>STORAGE</td>
<td>Trace IKE storage information.</td>
</tr>
<tr>
<td>EVENT</td>
<td>Trace IKE event information.</td>
</tr>
<tr>
<td>POLICY</td>
<td>Trace IKE policy information.</td>
</tr>
<tr>
<td>CONTROL</td>
<td>Trace IKE daemon control information.</td>
</tr>
<tr>
<td>MISC</td>
<td>Trace IKE miscellaneous information.</td>
</tr>
<tr>
<td>DEBUG</td>
<td>Trace IKE debugging information.</td>
</tr>
</tbody>
</table>

Enabling CTRACE at IKE daemon startup

A default minimum component trace is always started during IKE daemon initialization. Use a parmlib member to customize the parameters that are used to initialize the trace. The default IKE daemon component trace parmlib member is the SYS1.PARMLIB member CTIIKE00. The parmlib member name can be changed using the IKED_CTRACE_MEMBER environment variable.

**Tip:** The IKE daemon reads the IKED_CTRACE_MEMBER environment variable only during initialization. Changes to IKED_CTRACE_MEMBER after daemon initialization have no affect.

For a description of trace options, see Table 19 on page 350.

**Restriction:** In addition to specifying the trace options, you can also change the IKE daemon trace buffer size. The buffer size can be changed only at IKE initialization and has a maximum of 256 MB.

If the CTIIKE00 member or the member that is specified in IKED_CTRACE_MEMBER is not found when starting the IKE daemon, the following message is issued:

IEE5381 memberName MEMBER NOT FOUND IN PARMLIB

When this occurs, the IKE daemon component trace is started with a buffer size of 1 MB and the MINIMUM tracing option.
DESCRIPTION = This parmlib member causes component trace for
the TCP/IP IKE application to be initialized
with a trace buffer size of 1M
This parmlib members only lists those TRACEOPTS
values specific to IKE. For a complete list
of TRACEOPTS keywords and their values see
z/OS MVS Initialization and Tuning Reference.

TRACEOPTS

TRANSACTION

ON = OFF: PICK 1
ON
OFF
BUFSIZE: A VALUE IN RANGE 128K TO 256M
CTRACE buffers reside in IKE daemon Private storage
which is in the regions address space.
BUFSIZE(IM)

OPTIONS: NAMES OF FUNCTIONS TO BE TRACED, OR "ALL"

OPTIONS(    */
  'ALL ' */
  'MINIMUM ' */
  'INIT ' */
  'TERM ' */
  'EXCEPT ' */

Figure 30. SYS1.PARMLIB member CTIIKE00 (Part 1 of 2)
Steps for enabling the CTRACE at IKE daemon startup
Perform the following steps to enable the CTRACE at IKE daemon startup.

1. Edit the CTIIKE00 parmlib member and specify TRACEOPTS ON, the desired buffer size with the BUFSIZE() parameter and the desired CTRACE options. To direct the CTRACE to an external writer, also specify the name of the writer JCL procedure in the WTR() parameter. Refer to the example CTIIKE00 parmlib member.

2. Start the IKE daemon.

Steps for disabling the CTRACE at IKE daemon startup
Perform the following steps to disable the CTRACE at IKE daemon startup.

1. To disable the CTRACE at IKE daemon startup, edit the CTIIKE00 parmlib member and specify TRACEOPTS OFF.

2. Start the IKE daemon.

Step for enabling the CTRACE after the IKE daemon has started
Perform the following steps to enable the CTRACE after the IKE daemon has started.

• Issue the following console commands to enable the CTRACE to an internal buffer:

   TRACE CT,ON,COMP=SYSTCPIK,SUB={iked_jobname}
   R xx,OPTIONS=(option[,option2...]),END

   or

   Issue the following console commands to enable the CTRACE to an external writer:
Step for disabling the CTRACE after the IKE daemon has started

Perform the following steps to disable the CTRACE after the IKE daemon has started.

• Issue the following console commands to disable the CTRACE to an internal buffer:
  
  TRACE CT,OFF,COMP=SYSTCPIK,SUB=(iked_jobname)

  or

  TRACE CT,OFF,COMP=SYSTCPIK,SUB=(iked_jobname)

  TRACE CT,WTRSTOP=writer_proc

Step for displaying the CTRACE status

Perform the following step to display the CTRACE status.

• To display the CTRACE status, issue the following console command:
  
  D TRACE,COMP=SYSTCPIK,SUB=(iked_jobname)

Enabling CTRACE after IKE daemon initialization

After IKE daemon initialization, you must use the TRACE CT command to change the component trace options. Each time a new Component Trace is initiated, all prior trace options are turned OFF and the new options are put into effect. You can specify the trace options with or without the parmlib member. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47.

Formatting IKE daemon trace records

You can format component trace records using IPCS panels or a combination of the IPCS panels and the CTRACE command, either from a dump or from external-writer files. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47 for details.

Enter any combination of the following values as options to filter the CTRACE entries. The options must be entered using the following format:

  TYPE(option[,option]...)

You can use any of the options listed in Table 19 on page 350 except ALL and MINIMUM.
Chapter 10. Diagnosing network security services (NSS) server problems

This section describes how to diagnose network security services (NSS) server problems, and contains the following information:

- “Overview of diagnosing NSS server problems”
- “Network security services server debug information” on page 362
- “TCP/IP services component trace for the network security services (NSS) server” on page 368
- “Steps for enabling the CTRACE at network security service (NSS) server startup” on page 369

Overview of diagnosing NSS server problems

The NSS server provides network security services for one or more network security enforcement points. A component that requests network security services from the network security services server is called a network security client or NSS client. Problems with the network security services server may be categorized as follows:

- Network security services server configuration problems
- Network security services server internal problems
- Network security services server problems interacting with an external component such as a network security client or the Secure Access Facility (SAF).

The NSS server provides log output using syslogd and internal trace information using component trace (CTRACE). The log output is sufficient for diagnosing most network security services server problems and is the first place to look if you suspect a problem.

Common NSS server initialization problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
<tbody>
<tr>
<td>The NSS load module is not APF-authorized.</td>
<td>The NSS load module abends. The following message will be logged to the console: IEF4501 NSSD STEP1 - ABEND=S000 U4087 REASON=00000000</td>
<td>The NSS load module must be APF-authorized.</td>
</tr>
<tr>
<td>The NSS socket directory does not exist or else it cannot be created by the NSS server.</td>
<td>When NSS server syslog level 2 is set (NSS_SYSLOG_LEVEL_VERBOSE), debug message DBG0040I is generated. For example: DBG0040I NSS_VERBOSE Cannot create socket directory /var/sock - rc -1 errno 135 EDC5135I Not a directory. The NSS server will immediately shutdown.</td>
<td>1. The /var directory must already exist. 2. The /var/sock subdirectory must already exist, or else the userid that the NSS server is running under must have authority to create the /var/sock subdirectory.</td>
</tr>
</tbody>
</table>
## NSS client connection problems

The following table lists common problems when a network security services (NSS) client is unable to obtain services from the NSS server.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
</table>
| SSL is not properly configured for the NSS client connection to the NSS server. NSS client fails to connect. | When NSS server syslog level 8 is set (NSS_SYSLOG_LEVEL_CLIENTLIFECYCLE), debug message DBG0104I is generated: DBG0104I NSS_LIFECYCLE NSS connID 1 - the connection is not secure - the connection will be closed | For NSS IPSec client connections:  
  - AT-TLS must be enabled on both the client and server stacks with the TCPCONFIG TTLS statement in the TCP/IP profile.  
  - AT-TLS policies must be defined to secure the connection.  
  - If AT-TLS is enabled and the definitions are configured on the client and server stacks but DBG0104I is still displayed then refer to Chapter 29, "Diagnosing Application Transparent Transport Layer Security (AT-TLS)," on page 701.  
For NSS XMLAppliance client connections:  
  - The NSS XMLAppliance client must support an SSL/TLS negotiation protocol that is compatible with that which is configured for the NSS server.  
  - The NSS server stack must have AT-TLS enabled with the TCPCONFIG TTLS statement in the TCP/IP profile.  
  - AT-TLS policies must be defined for the NSS server stack to secure the connection. See AT-TLS policy in [z/OS Communications Server: IP Configuration Guide](https://www.ibm.com/support/docviewialias/5784).  
  - If AT-TLS is enabled on the server stack, and the definition is configured on the server stack but DBG0104I is still displayed then see Chapter 29, "Diagnosing Application Transparent Transport Layer Security (AT-TLS)," on page 701.  
Configuration of the client’s TLS settings are left up to the client application’s implementation. |
| The userid used for the NSS client connection to the NSS server has insufficient authority to access services requested. | When NSS server syslog level 2 is set (NSS_SYSLOG_LEVEL_VERBOSE), debug message DBG0032I is generated. For example: DBG0032I NSS_VERBOSE ServauthCheck (USER2 , EZB.NSS.MVS093.CLIENT2.IPSEC.CERT) rc 4 (DENY) racfRC 4 racfrsn 0 | SAF resource permissions are required to access NSS IPSec services:  
  - EZB.NSS.sysname.clientname.IPSEC.CERT  
  - EZB.NSS.sysname.clientname.IPSEC.NETMGMT  
SAF resource permissions are required to access the NSS XMLAppliance services:  
  - EZB.NSS.sysname.clientname.XMLAPPLIANCE.SAFACCESS  
  - EZB.NSS.sysname.clientname.XMLAPPLIANCE.CERT  
  - EZB.NSS.sysname.clientname.XMLAPPLIANCE.PRIVKEY  
These resources must be defined on the NSS server system and the client userid must be permitted read access to them. |
**Table 21. Common NSS client connection problems (continued)**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
<tbody>
<tr>
<td>An NSS client appears to be connected to two instances of the NSS server.</td>
<td>For an NSS IPSec client, the ipsec -x display for both NSS servers shows the same client connected. For an NSS client, the nssctl -d for both NSS servers shows the same client connected.</td>
<td>Under normal termination, an NSS client will issue a disconnect to close its connection with the NSS server. In some rare recovery situations, the NSS server may not be aware that a connection with a NSS client has ended. When the client restarts or attempts to reconnect, it is possible it may connect to a different NSS server instance, such as the backup server or a NSS server on another system when the client is connecting on a distributed dynamic VIPA. Use the ipsec -w display on the system running the affected NSS IPSec client to determine which NSS server the IPSec client is actually connected. Optionally, use the Netstat DRop/-D command to close out the old connection on the other NSS server.</td>
</tr>
</tbody>
</table>

| NSS clients are failing to connect to the NSS server. | The NSS server issues the EZD1371I console message to indicate the disabled discipline and closes the connection. | The NSS server has been configured to disable the specified discipline. Modify the NSS server configuration to enable the specified discipline. See z/OS Communications Server: IP Configuration Reference for more information about the NSS server configuration. |

The following table lists common problems when requests from a network security services (NSS) client fails.

**Table 22. Common NSS client request failures**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
<tbody>
<tr>
<td>The userid used for the NSS client connection has insufficient authority to access client certificates.</td>
<td>When NSS server syslog level 4 is set (NSS_SYSLOG_LEVEL_CERTINFO), debug message DBC0004I is generated: DBG0004I NSS_CERTINFO Client MVS093 TCP653 connected as userid USER1 is not authorized to profile EZB.NSSCERT.VIC012.NSCLIENT3.HOST associated with matching certificate ( NSCLIENT3 ) for request 00000000000000150000000000000000</td>
<td>SAF resource permissions are required to access certificates from the NSS server: • EZB.NSSCERT.sysname.mappedlabelname.HOST • EZB.NSSCERT.sysname.mappedlabelname.CERTAUTH These resources must be defined on the NSS server system and the client userid must be permitted read access to them.</td>
</tr>
</tbody>
</table>

| The userid used for the NSS client connection has insufficient authority to access the private keys associated with client certificates. | When NSS server syslog level 4 is set (NSS_SYSLOG_LEVEL_CERTINFO), debug message DBC0004I is generated: Jun 25 14:54:43 MVS093 NSSD: DBG0004I NSS_CERTINFO Client XML_ClientB8 connected as userid USER198 is not authorized to profile EZB.NSSCERT.MVS093.KEY1024ICSF,PRIVKEY associated with matching certificate ( Key1024ICSF ) for request 9987A24B8786648B24BF0F10000 | SAF resource permissions are required to access private keys associated with certificates from the NSS server: EZB.NSSCERT.sysname.mappedlabelname.PRIVKEY These resources must be defined on the NSS server system and the client userid must be permitted read access to them. |

**NSS XMLAppliance client API return codes and reason codes**

The following table lists and describes the possible return codes and reason codes returned by the NSS server to NSS XMLAppliance clients.
<table>
<thead>
<tr>
<th>Return code (NMsMrC)</th>
<th>Reason code (NMsMrSn)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>No error</td>
</tr>
</tbody>
</table>
| EINVAL(121)          | NMsRsnBadIdent (1)    | Invalid message or record identifier supplied in message.  
                      | System Action: Request is failed but connection remains open.  
                      | Response: Re-issue the request and send a correctly formatted message |
| EINVAL(121)          | NMsRsnBadVersion (2)  | Invalid version supplied in message header.  
                      | System Action: Request fails but connection remains open.  
                      | Response: Send a correctly formatted message. |
| EINVAL(121)          | NMsRsnBadType (3)     | Unsupported or unknown message type supplied in message header.  
                      | System Action: Request is failed but connection remains open.  
                      | Response: Send a supported message type. |
| EINVAL(121)          | NMsRsnExcessiveSize (4) | Excessive message size.  
                      | System Action: Connection is closed.  
                      | Response: Re-issue the connection and send a correctly formatted message. |
| EINVAL(121)          | NMsRsnHdrSize (5)     | Message header size is not valid.  
                      | System Action: Request is failed but connection remains open.  
                      | Response: Send a message with the header size field set to the correct value. |
| EINVAL(121)          | NMsRsnMsgSize (6)     | Message size is not valid. For example, the message may be too short, or the message size may be greater than the sum of its parts.  
                      | System Action: Request is failed but connection remains open.  
                      | Response: Send a correctly formatted message. |
| EINVAL(121)          | NMsRsnReservedNonzero (7) | Reserved data in message header, record header, or record data is non-zero value. Reserved fields must be set to 0 for compatibility with any future versions of the interface.  
                      | System Action: Request is failed but connection remains open.  
                      | Response: Send a message with reserved fields set to 0. |
| EINVAL(121)          | NMsRsnRecordLength (8) | Unrecognized record length supplied in message. Length does not correspond to known record data.  
                      | System Action: Request is failed but connection remains open.  
                      | Response: Send a message with input filters of the correct length. |
| EINVAL(121)          | NMsRsnRecordCount (9) | Unsupported record count supplied in message. NMI requests currently support a maximum of twenty input records.  
                      | System Action: Request is failed but connection remains open.  
                      | Response: Send a message with the correct number of input filters. |
| EINVAL(121)          | NMsRsnSectionLength (10) | Unrecognized section length supplied in record. Length does not correspond to known section data.  
                      | System Action: Request is failed but connection remains open.  
<pre><code>                  | Response: Send a message with correct input filters. |
</code></pre>
<table>
<thead>
<tr>
<th>Return code (NMsmRc)</th>
<th>Reason code (NMsmRsn)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EINVAL(121)</td>
<td>NMsRsRsnSectionCount (11)</td>
<td>Unrecognized section count supplied in record. NMI requests currently allow one section in an input record. System Action: Request is failed but connection remains open. Response: Send a message with correct input filters.</td>
</tr>
<tr>
<td>EINVAL(121)</td>
<td>NMsRsRsnClientAlreadyConnected (10002)</td>
<td>The remote client name is already registered with the NSS server. NSS client names must be unique. System Action: Connection is closed. Response: Re-issue the connection request using a unique client name.</td>
</tr>
<tr>
<td>EINVAL(121)</td>
<td>NMsRsRsnNoMatchingCert (10004)</td>
<td>The NSS server could not find a matching certificate. The certificate does not exist on the NSS servers configured keyring, it is marked untrusted, or the NSS client does not have the authority to use the certificate. System Action: Request is failed but connection remains open. Response: Re-issue the request with an existing, trusted, and authorized certificate label.</td>
</tr>
<tr>
<td>EINVAL(121)</td>
<td>NMsRsRsnNoCertRep (10014)</td>
<td>The NSS server does not have a certificate repository available to process the request. System Action: Request is failed but connection remains open. Response: Start the appropriate application logic.</td>
</tr>
<tr>
<td>EINVAL(121)</td>
<td>NMsRsRsnInvalidService (10021)</td>
<td>A service has been requested that is not affiliated with the requested discipline. System Action: Connection is closed. Response: Re-attempt the connection and request only the services affiliated with the requested discipline.</td>
</tr>
<tr>
<td>EINVAL(121)</td>
<td>NMsRsRsnInvalidIdentity (10022)</td>
<td>The SAF user access check request did not contain a valid user identity. The SAF ID is not recognized or, if a certificate was provided as input to the NSS, CheckUserAccessReqToSrv, no valid certificate name filter mapping is defined in RACF. System Action: NSS server processing continues. Response: Ensure that the user identity is entered correctly and reissue the request.</td>
</tr>
<tr>
<td>EINVAL(121)</td>
<td>NMsRsRsnInvalidSAFClass (10023)</td>
<td>The SAF class specified in the NSS_SAFCheckUserAccessReqToSrv message was unsupported. System Action: None Response: Contact the NSS client vendor. The SERVAUTH class is the only currently-supported SAF class.</td>
</tr>
<tr>
<td>EINVAL(121)</td>
<td>NMsRsRsnInvalidProfileLength (10024)</td>
<td>The SAF user access check request contained an invalid profile length. System Action: NSS server processing continues. Response: Contact the NSS client vendor. The maximum profile length for the SERVAUTH class is 64 bytes.</td>
</tr>
<tr>
<td>EINVAL(121)</td>
<td>NMsRsRsnInvalidDiscipline (10025)</td>
<td>The discipline specified in the connection request contains an invalid value. System Action: Connection is closed. Response: Re-attempt the connection and pass in a valid discipline.</td>
</tr>
</tbody>
</table>
### Table 23. NSS XMLAppliance client API return codes and reason codes (continued)

<table>
<thead>
<tr>
<th>Return code (NMsMRc)</th>
<th>Reason code (NMsMRsn)</th>
<th>Description</th>
</tr>
</thead>
</table>
| EINVAL(121)          | NMsRsnBadUpdate (10026) | The client has attempted to update its client information using values that cannot be changed after the initial connection has succeeded.  
**System Action:** Request is failed but connection remains open.  
**Response:** Re-attempt the update by changing only those fields which are acceptable under and update. |
| EINVAL(121)          | NMsRsnInvalidAPIVersion (10027) | An NSS client has attempted to connect to the NSS server and has specified adherence to an API version that is insufficient for the requested discipline.  
**System Action:** Connection is closed.  
**Response:** Contact the NSS client vendor. NSS XMLAppliance clients must adhere to NMsec_NSS_API_VERSION2 (2) or higher. |
| EINVAL(121)          | NMsRsnInvalidAccessLevel (10028) | The SAF user access check request contained an invalid value for the requested access level.  
**System Action:** NSS server processing continues.  
**Response:** Contact the NSS client vendor. Supported access levels are:  
<table>
<thead>
<tr>
<th>Requested access</th>
<th>Hexadecimal value</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td>0x02</td>
</tr>
<tr>
<td>UPDATE</td>
<td>0x04</td>
</tr>
<tr>
<td>CONTROL</td>
<td>0x08</td>
</tr>
<tr>
<td>ALTER</td>
<td>0x80</td>
</tr>
</tbody>
</table>
| EINVAL(121)        | NMsRsnInvalidClientName (10029) | NSS_ConnectClientReqToSrv or NSS_UpdateClientInfoReqToSrv request is invalid.  
**System Action:** If the client name is invalid on the connect, the request is failed and the connection is closed. If the client name is invalid on the update, the request is failed, the connection remains open, but the client remains in the update pending state until a valid update is provided.  
**Response:** Re-attempt the connect or update by providing a valid NSS client name. Valid characters are [a-zA-Z0-9_] The client name must be left-justified and blank-padded. Embedded spaces are invalid. |
| EINVAL(121)        | NMsRsnInvalidCertLabelName (10030) | The NSS XMLAppliance client request contained an invalid value for the requested certificate label name.  
**System Action:** NSS server processing continues.  
**Response:** The NSS client should re-issue the request with a valid certificate label name. The values accepted in this field are documented in the request input for the NSS_GetCertificateReqToSrv call. |
| EINVAL(121)        | NMsRsnNoPrivateKey (10031) | The certificate does not contain the private key.  
**System Action:** Request is failed but connection remains open.  
**Response:** If the certificate is intended to have a private key, then contact the NSSD administrator to determine what action to take. |
| EACCESS(111)        | NMsRsnDisconnectPending (1) | A client disconnect operation is pending, so no new request messages are being accepted.  
**System Action:** Request is rejected and connection is eventually closed.  
**Response:** Stop sending requests for this connection. Reconnect to the server and re-issue the request. |
Table 23. NSS XMLAppliance client API return codes and reason codes (continued)

<table>
<thead>
<tr>
<th>Return code (NMsMRc)</th>
<th>Reason code (NMsMRsn)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EACCES(111)</td>
<td>NMsRsnUpdatePending (2)</td>
<td>A client update operation is pending, so no new request messages are being accepted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Re-issue the request.</td>
</tr>
<tr>
<td>EACCES(111)</td>
<td>NMsRsnNoAuthForService (4)</td>
<td>Userid is not authorized to use the NSS server for the requested service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Connection is closed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Ensure that the clients access to the requested service is defined in the servers SERVAUTH profiles (EZB.NSS.sysname.clientname.discipline.service).</td>
</tr>
<tr>
<td>EACCES(111)</td>
<td>NMsRsnUserAuthentication (10001)</td>
<td>User authentication failed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> NSS server processing continues.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Ensure that the userid is defined in the server’s security manager and that the password or passticket is formed correctly.</td>
</tr>
<tr>
<td>EACCES(111)</td>
<td>NMsRsnSAFUserNotAuthenticated (10002)</td>
<td>The SAF user identity specified in the request failed authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> The NSS server successfully completed the authentication check. Processing continues.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> NSS client independent. The NSS client should react accordingly.</td>
</tr>
<tr>
<td>EACCES(111)</td>
<td>NMsRsnSAFUserAccessDenied (10003)</td>
<td>The SAF user access check indicates access denied from the security server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> The NSS server successfully completed the access check. Processing continues.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> NSS client independent. The NSS client should react accordingly.</td>
</tr>
<tr>
<td>EACCES(111)</td>
<td>NMsRsnSAFResourceError (10004)</td>
<td>The SAF user access check indicates that a SAF server is not installed, has not been started, or the specified class is not active, is not defined, or that no profile exists for the specified resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> The NSS server failed to complete the access check. Processing continues.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> NSS client independent. The NSS client should react accordingly. Most commonly this reason code indicates that the profile that was queried does not exist. It would then be up to the NSS client to decide whether this implies access denied or access granted.</td>
</tr>
<tr>
<td>EACCES(111)</td>
<td>NMsRsnUnsupportedDiscipline (10005)</td>
<td>The discipline specified in the connection request is currently disabled in the NSS server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Connection is closed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Modify the NSS server configuration to enable the specified discipline.</td>
</tr>
<tr>
<td>EACCES(111)</td>
<td>NMsRsnNoAuthForPrivKey (10006)</td>
<td>Userid is not permitted to use the requested certificate’s private key.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Request is failed but connection remains open.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Permit user to security resource SERVAUTH profile EZB.NSSCERT.sysname.mappedlabelname.PRIVKEY</td>
</tr>
</tbody>
</table>
### Table 23. NSS XMLAppliance client API return codes and reason codes (continued)

<table>
<thead>
<tr>
<th>Return code (NMsmRc)</th>
<th>Reason code (NMsmRsn)</th>
<th>Description</th>
</tr>
</thead>
</table>
| EACCES(111)          | NMsmRsnPrivKeyProtected (10007) | The certificate’s private key is protected from being retrieved.  
  **System Action**: Request is failed but connection remains open.  
  **Response**: If the private key is intended to be retrieved, then contact the NSSD administrator to determine what action to take. |
| EACCES(111)          | NMsmRsnPrivKeyNotProtected (10008) | The certificate’s private key is not stored in the ICSF PKA key data set (PKDS). Signature generation and decryption require use of certificates for which the private keys are stored in the ICSF PKDS.  
  **System Action**: Request is failed but connection remains open.  
  **Response**: If the private key is intended to be used for signature generation or decryption, then contact the NSSD administrator to determine what action to take. |
| EACCES(111)          | NMsmRsnNoAuthForCert (10009) | Userid is not permitted to retrieve the requested certificate.  
  **System Action**: Request is failed but connection remains open.  
  **Response**: Permit user to proper security resource SERVAUTH profiles: Read access to either of the following profiles will authorize the NSS server to retrieve the requested certificate:  
  - EZB.NSSCERT.sysname.mappedlabelname.CERTAUTH  
  - EZB.NSSCERT.sysname.mappedlabelname.HOST |
| ENOMEM(132)          | 0 | Insufficient storage available in the server to process the request.  
  **System Action**: Request is failed but connection remains open.  
  **Response**: Increase the REGION size for the NSS server, or send a message with a narrower set of input filters to limit the response. |
| ENOLCK(131)          | 0 | Failed to obtain an internal lock.  
  **System Action**: Request fails but connection remains open. A message will appear in the MVS system log with additional diagnostic information.  
  **Response**: Contact IBM service. |
| EGSKCMS(10004)       | See gskcms.h. The Reason code represents the GSK (system SSL) return code provided on the failed call. | A System SSL error was encountered when issuing a System SSL library call. The NSS reason code will contain the System SSL CMS Status Code.  
  **System Action**: Request is failed but connection remains open.  
  **Response**: Review the system SSL CMS status codes from [z/OS Cryptographic Services System SSL Programming](https://www.ibm.com/docs/en/zos-cryptographic-services) |
| ECSFBEXT(10005)      | The high-order 16 bits of the reason code represent the ICSF return Code. The low-order 16 bits of the reason code represent the ICFS reason code. | An Integrated Cryptographic Service Facility (ICSF) error was encountered while performing an RSA operation. The reason code will contain the ICSF return code(high order 16 bits) and reason code(low order 16 bits).  
  **System Action**: Request is failed but connection remains open.  

---

### Network security services server debug information

**Obtaining syslog debug information for the network security service server**

The SyslogLevel parameter in the network security services server configuration file controls the level of NSS server internal debug information that is sent to
syslog. When configuring with the IBM Configuration Assistant for z/OS Communications Server, use the NSS Daemon Syslog Trace panel to configure the level of network security services server internal debug information that is sent to syslog. Refer to z/OS Communications Server: IP Configuration Reference or the IBM Configuration Assistant for z/OS Communications Server online help for more information.

**Abends**

Messages and error-related information should be sent to the system console when an abend occurs during NSS server processing. NSSD will initiate a system dump for the abend condition. System dumps of the NSS server include Language Environment data. The Language Environment IPCS verbexit LEDATA can be used to format this information. Refer to z/OS Language Environment Debugging Guide for more information. The following is a sample IPCS verbexit LEDATA command:

```
verbx ledata 'asid(68) tcb(007E5E88) ceedump nthreads(*)' 
```

In this example, the network security services server asid is 0x68 and the address of the abended NSS TCB is 0x007E5E88.

**Error codes**

Several messages display a return code and reason generated by the NSS server. Most of these return codes and reasons are generated in support of the application interface for managing IP filtering and IPSec on remote network security clients. These return codes and reasons are documented in z/OS Communications Server: IP Programmer’s Guide and Reference.

Additional return codes and reasons may be generated by the NSS server. These return codes and reasons are generated in support of remote management services offered to remote network security clients and are explained in the following error codes table.

**Table 24. NSS IPSec client API return codes and reason codes**

<table>
<thead>
<tr>
<th>Return code (NMsMrC)</th>
<th>Reason code (NMsMRsn)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGSKSIGN (10001)</td>
<td>gsk_status code generated during the failure. Common codes are:</td>
<td>A System SSL CMS error was encountered while attempting to create a signature. The reason code will contain the System SSL return code.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_ALG_NOT_SUPPORTED]</td>
<td>System Action: Request fails but connection remains open.</td>
</tr>
<tr>
<td></td>
<td>The signature algorithm is not supported.</td>
<td>Response: Examine gsk_status code (returned as the reason code), which are documented in z/OS Cryptographic Services System SSL Programming. Verify the failed message contained correct data. If it did not then take action to correct the message content. If it did then contact the NSSD administrator to determine what action to take.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_BAD_DIGEST_SIZE]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The digest size is not correct.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[CMSERR_KEY_MISMATCH]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The supplied key does not match the signature algorithm.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[CMSERR_NO_MEMORY]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Insufficient storage is available.</td>
<td></td>
</tr>
</tbody>
</table>
Table 24. NSS IPSec client API return codes and reason codes (continued)

<table>
<thead>
<tr>
<th>Return code (NMsMRC)</th>
<th>Reason code (NMsMRsn)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGSKVAL (10002)</td>
<td>gsk_status code generated during the failure. Common codes are:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[CMSERR_BAD_HANDLE]</td>
<td>The database handle is not valid.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_BAD_ISSUER_NAME]</td>
<td>The certificate issuer name is not valid.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_BAD_SIGNATURE]</td>
<td>The signature is not correct.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_CERT_CHAIN_NOT_TRUST]</td>
<td>The certification chain is not trusted.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_CERTIFICATE_REVOKED]</td>
<td>The certificate is revoked.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_EXPIRED]</td>
<td>The certificate is expired.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_INCORRECT_DTYPE]</td>
<td>The database type does not support certificates.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_INCORRECT_KEY_USAGE]</td>
<td>The issuer certificate does not allow signing certificates.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_ISSUER_NOT_CA]</td>
<td>The certificate issuer is not a certification authority.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_ISSUER_NOT_FOUND]</td>
<td>The issuer certificate is not found in one of the data sources.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_NAME_CONSTRAINTS_VIOLATED]</td>
<td>The certificate name is not consistent with the name constraints.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_NAME_NOT_SUPPORTED]</td>
<td>The AuthorityKeyIdentifier extension name is not a directory name.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_NOT_YET_VALID]</td>
<td>The certificate is not yet valid.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_PATH_TOO_LONG]</td>
<td>The certification chain exceeds the maximum allowed by the CA.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_SELF_SIGNED_NOT_FOUND]</td>
<td>A self-signed certificate is not found in a trusted data source.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return code (NMsMRC)</th>
<th>Reason code (NMsMRsn)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGSKVER (10003)</td>
<td>gsk_status code generated during the failure. Common codes are:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[CMSERR_ALG_NOT_SUPPORTED]</td>
<td>The signature algorithm is not supported.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_BAD_DIGEST_SIZE]</td>
<td>The digest size is not correct.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_BAD_SIGNATURE]</td>
<td>The signature is not correct.</td>
</tr>
<tr>
<td></td>
<td>[CMSERR_KEY_MISMATCH]</td>
<td>The supplied key does not match the signature algorithm.</td>
</tr>
</tbody>
</table>

A System SSL CMS error was encountered while attempting to verify a signature. The reason code will contain the System SSL return code.

**System Action:** Request fails but connection remains open.

**Response:** Examine gsk_status code (returned as the reason code), which are documented in z/OS Cryptographic Services System SSL Programming. Verify the failed message contained correct data. If it did not then take action to correct the message content. If it did and the reason code is one of the following contact the certificate owner and inform them of the problem encountered with the certificate:

- CMSERR_BAD_ISSUER_NAME
- CMSERR_BAD_SIGNATURE
- CMSERR_CERTIFICATE_REVOKED
- CMSERR_EXPIRED
- CMSERR_INCORRECT_KEY_USAGE
- CMSERR_ISSUER_NOT_CA
- CMSERR_NAME_CONSTRAINTS_VIOLATED
- CMSERR_NAME_NOT_SUPPORTED
- CMSERR_NOT_YET_VALID
- CMSERR_PATH_TOO_LONG

If the reason code is anything other than the codes above, contact the NSSD administrator to determine what action to take. Other common reason codes include:

- CMSERR_BAD_HANDLE
- CMSERR_CERT_CHAIN_NOT_TRUST
- CMSERR_INCORRECT_DTYPE
- CMSERR_ISSUER_NOT_FOUND
- CMSERR_SELF_SIGNED_NOT_FOUND

---

364  
z/OS V1R11.0 Comm Svr: IP Diagnosis Guide
Table 24. NSS IPSec client API return codes and reason codes (continued)

<table>
<thead>
<tr>
<th>Return code (NMsmRc)</th>
<th>Reason code (NMsmRsn)</th>
<th>Description</th>
</tr>
</thead>
</table>
| EACCES (111)         | NMsmRsnUserAuthentication (10001) | User authentication failed  
  **System Action:** Request fails and the connection is closed.  
  **Response:** Verify the following: The user ID under which the NSS client connects to the NSS server is correct. The password used to authenticated that user ID is valid, or the application key used to generate the passticket is correct (this key is stored in the SAF-enabled security manager). |
| EACCES (111)         | NMsmRsnNoAuthForService (4) | The NSS client does not have access to the requested service through the governing SERVAUTH profile.  
  **System Action:** Request fails but connection remains open.  
  **Response:** If appropriate, define a SERVAUTH profile that will allow the requested access. |
| EACCES (111)         | NMsmRsnNoAuthForClientname (3) | The user ID in the connection request is not authorized to act on behalf of the NSS clientName  
  **System Action:** Request fails and the connection is closed.  
  **Response:** Ensure that all of the following are correct: The user ID (and password, if necessary) as configured at the client. The client name as configured at the client. Also ensure that the appropriate SERVAUTH profiles are defined at the server system for the client. |
| EACCES (111)         | NMsmRsnDisconnectPending (1) | A disconnect operation is pending.  
  **System Action:** Request fails but connection remains open for a very short time.  
  **Response:** The client must reconnect the server before any more NSS services can be requested. |
| EACCESS (111)        | NMsmRsnUnsupportedDiscipline (10005) | The discipline specified in the connection request is currently disabled in the NSS server.  
  **System Action:** Connection is closed.  
  **Response:** Modify the NSS server configuration to enable the specified discipline. |
| EINVAL (121)         | NMsmRsnClientAlreadyConnected (10002) | Client is already connected to this server.  
  **System Action:** Request fails and the connection is closed.  
  **Response:** If appropriate, disconnect the active client and reattempt the connection request. |
| EINVAL (121)         | NSSRsnRIDNotInCert (10003) | The certificate used to sign does not contain remote ID specified.  
  **System Action:** Request fails but connection remains open.  
  **Response:** None - this is an informational code only. |
| EINVAL (121)         | NSSRsnBadCert (10005) | Certificate not valid.  
  **System Action:** Request fails but connection remains open.  
  **Response:** If the failing certificate is one that is stored on the local system, it should be refreshed or replaced. If that certificate comes from a remote system, then this is an informational code only. |
<table>
<thead>
<tr>
<th>Return code (NFsMRC)</th>
<th>Reason code (NFsMRsn)</th>
<th>Description</th>
</tr>
</thead>
</table>
| EINVAL (121)         | NSSRsnUnsupportedCert (10006) | Unsupported certificate encoding.  
**System Action:** Request fails but connection remains open.  
**Response:** Contact IBM service. |
| EINVAL (121)         | NSSRsnBadLIDType (10007) | Unrecognized LID type.  
**System Action:** Request fails but connection remains open.  
**Response:** Contact IBM service. |
| EINVAL (121)         | NSSRsnBadLIDValue (10008) | LID value not valid.  
**System Action:** Request fails but connection remains open.  
**Response:** Contact IBM service. |
| EINVAL (121)         | NSSRsnBadRIDType (10009) | Unrecognized LID type.  
**System Action:** Request fails but connection remains open.  
**Response:** Contact IBM service. |
| EINVAL (121)         | NSSRsnBadRIDValue (10010) | LID value not valid.  
**System Action:** Request fails but connection remains open.  
**Response:** Contact IBM service. |
| EINVAL (121)         | NSSRsnBadLocalIPaddr (10011) | Local IPaddr not valid.  
**System Action:** Request fails but connection remains open.  
**Response:** Contact IBM service. |
| EINVAL (121)         | NSSRsnBadRemoteIPaddr (10012) | Remote IPaddr not valid.  
**System Action:** Request fails but connection remains open.  
**Response:** Contact IBM service. |
| EINVAL (121)         | NSSRsnAddrVersionMismatch (10013) | Local and remote IP address versions don’t match.  
**System Action:** Request fails but connection remains open.  
**Response:** Contact IBM service. |
| EINVAL (121)         | NSSRsnNoCertRep (10014) | Certificate repository not available.  
**System Action:** Request fails but connection remains open.  
**Response:** Create or restore the certificate repository and then retry the request. |
| EINVAL (121)         | NSSRsnBadHashSize (10016) | Hash size not valid for specified hash algorithm.  
**System Action:** Request fails but connection remains open.  
**Response:** Contact IBM service. |
| EINVAL (121)         | NSSRsnBadHashAlg (10017) | Hash algorithm not supported.  
**System Action:** Request fails but connection remains open.  
**Response:** Contact IBM service. |
| EINVAL (121)         | NSSRsnSaNotInCertLife (10018) | SA lifetime not in certificate lifetime.  
**System Action:** Request fails but connection remains open.  
**Response:** None - this is an informational code only. |
<table>
<thead>
<tr>
<th>Return code (NMsMRC)</th>
<th>Reason code (NMsMRsn)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EINVAL (121)</td>
<td>NSSRsnBadCa (10019)</td>
<td>The DER encoding type specified for the Certificate Authority name is unrecognized.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Request fails but connection remains open.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Contact IBM service.</td>
</tr>
<tr>
<td>EINVAL (121)</td>
<td>NSSRsnUnsupportedCaType (10020)</td>
<td>Unsupported CA encoding.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Request fails but connection remains open.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Contact IBM service.</td>
</tr>
<tr>
<td>EINVAL (121)</td>
<td>NMsrnInvalidService (10021)</td>
<td>A service has been requested that is not affiliated with the requested discipline.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Connection is closed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Re-attempt the connection and request only the services affiliated with the requested discipline.</td>
</tr>
<tr>
<td>EINVAL (121)</td>
<td>NMsrnInvalidDiscipline (10025)</td>
<td>The discipline specified in the connection request contains an invalid value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Connection is closed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Re-attempt the connection and pass in a valid discipline.</td>
</tr>
<tr>
<td>EINVAL (121)</td>
<td>NMsrnBadUpdate (10026)</td>
<td>The client has attempted to update its client information using values that cannot be changed after the initial connection has succeeded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Request is failed but connection remains open.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Re-attempt the update by changing only those fields which are acceptable under an update.</td>
</tr>
<tr>
<td>EINVAL (121)</td>
<td>NMsrnInvalidAPIVersion (10027)</td>
<td>An NSS client has attempted to connect to the NSS server and has specified adherence to an API version that is insufficient for the requested discipline.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Connection is closed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Re-attempt the connection using an accepted API version. NSS IPSec clients must adhere to NMsec_NSS_API_VERSION1 (1) or higher. NSS XMLAppliance clients must adhere to NMsec_NSS_API_VERSION2 (2) or higher.</td>
</tr>
<tr>
<td>ENOLCK (131)</td>
<td>0</td>
<td>Failed to obtain an internal lock.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Request fails but connection remains open. A message will appear in the MVS system log with additional diagnostic information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Contact IBM service.</td>
</tr>
<tr>
<td>ENOMEM (132)</td>
<td>NMsrnTooManyConns (1)</td>
<td>The NSS server is already using its maximum number of 500 connections and cannot accept any more.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Connection is not opened and the request is failed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Try the request again later.</td>
</tr>
</tbody>
</table>
Table 24. NSS IPSec client API return codes and reason codes (continued)

<table>
<thead>
<tr>
<th>Return code (NMsMrC)</th>
<th>Reason code (NMsMrRsN)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENXIO (138)</td>
<td>NSSRsNUnknownClientName (10001)</td>
<td>The specified client name not recognized.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>System Action:</strong> Request fails and the connection is closed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Response:</strong> Verify that the client name was specified correctly and that the NSS client is connected to the NSS server. Note, however, that this error code often occurs when directing a request to an NSS client that is not currently connected to the NSS server.</td>
</tr>
</tbody>
</table>

### TCP/IP services component trace for the network security services (NSS) server

The network security services (NSS) server uses component trace support to trace internal operations. This section describes how to specify NSS server trace and formatting options. For short descriptions of other tracing procedures, such as displaying trace status, see Chapter 5, “TCP/IP services traces and IPCS support,” on page 47.

For detailed information, refer to the following information:

- z/OS MVS Diagnosis: Tools and Service Aids for information about component trace procedures.
- z/OS MVS Initialization and Tuning Reference for information about the component trace SYS1.PARMLIB member.
- z/OS MVS System Commands for information about commands.
- z/OS MVS Programming: Authorized Assembler Services Guide for procedures and return codes for component trace macros.
- z/OS MVS IPCS Commands for information about IPCS commands.
- z/OS MVS IPCS User's Guide for information about using IPCS.

### Using CTRACE

You can specify component trace options at NSS server initialization or after the NSS server has initialized.

Table 25 lists the network security services server trace options.

Table 25. NSS server trace options

<table>
<thead>
<tr>
<th>Trace event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Select all types of records. Note: This option may have an impact on performance.</td>
</tr>
<tr>
<td>MINIMUM</td>
<td>Select the network security services server’s minimum level of tracing. This level includes the INIT, EXCEPT, and TERM categories.</td>
</tr>
<tr>
<td>INIT</td>
<td>Select NSS server initialization information.</td>
</tr>
<tr>
<td>TERM</td>
<td>Select NSS server termination information.</td>
</tr>
<tr>
<td>EXCEPT</td>
<td>Select NSS server exception information.</td>
</tr>
<tr>
<td>CONFIG</td>
<td>Select NSS server configuration information.</td>
</tr>
</tbody>
</table>
Table 25. NSS server trace options (continued)

<table>
<thead>
<tr>
<th>Trace event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMANDS</td>
<td>Select processing of NSS server commands from the console or command line.</td>
</tr>
<tr>
<td>LOGMSGS</td>
<td>Select NSS server syslog messages. These entries can be used to easily correlate system log messages to a specific point in the CTRACE log.</td>
</tr>
<tr>
<td>ROUTING</td>
<td>Select NSS server threading and request dispatching information</td>
</tr>
<tr>
<td>SERIAL</td>
<td>Select NSS server serialization information</td>
</tr>
<tr>
<td>EVENT</td>
<td>Select NSS server event information</td>
</tr>
<tr>
<td>SOCKETS</td>
<td>Select NSS server socket information</td>
</tr>
<tr>
<td>PERFORM</td>
<td>Select NSS server performance information</td>
</tr>
<tr>
<td>REQUESTS</td>
<td>Select NSS server request/response information</td>
</tr>
<tr>
<td>FLOW</td>
<td>Select NSS server code flow information</td>
</tr>
<tr>
<td>STORAGE</td>
<td>Select NSS server storage information</td>
</tr>
<tr>
<td>CERTOPS</td>
<td>Select NSS server certificate operations information (cert cache ops and signature verify/create calls).</td>
</tr>
<tr>
<td>CONTROL</td>
<td>Select NSS server control information</td>
</tr>
<tr>
<td>DEBUG</td>
<td>Select NSS server debugging information</td>
</tr>
<tr>
<td>VERBOSE</td>
<td>Select NSS server verbose debugging information</td>
</tr>
</tbody>
</table>

Steps for enabling the CTRACE at network security service (NSS) server startup

A default minimum component trace is always started during NSS server initialization. Use a parmlib member to customize the parameters that are used to initialize the trace. The default NSS server component trace parmlib member is the SYSPARMLIB member CTINSS00. You can change the parmlib member name using the NSSD_CTRACE_MEMBER environment variable.

**Rule:** The NSS server reads the NSSD_CTRACE_MEMBER environment variable only during initialization. Changes to NSSD_CTRACE_MEMBER after server initialization have no effect.

For a description of trace options, see Table 25 on page 368

**Restriction:** In addition to specifying the trace options, you can also change the NSS trace buffer size. The buffer size can be changed only at NSS initialization and has a maximum of 256 MB.

If the CTINSS00 member or the member that is specified in NSSD_CTRACE_MEMBER is not found when starting the network security services server, the following message is issued:

IEE5381 memberName MEMBER NOT FOUND IN PARMLIB
When this message is issued, the NSS component trace is started with a buffer size of 1 MB and the minimum tracing option.

INSERT XMP HERE SYS1.PARMLIB member CTINSS00

Steps for enabling the CTRACE at network security services server startup

1. Edit the CTINSS00 parmlib member and specify the following:
   • TRACEOPTS ON
   • The desired buffer size with the BUFSIZE() parameter
   • The desired CTRACE options. To direct the CTRACE to an external writer, also specify the name of the writer JCL procedure in the WTR() parameter.

   Refer to the example CTINSS00 parmlib member.

2. Start the network security services (NSS) server.

Steps for disabling the CTRACE at network security services server startup

1. To disable the CTRACE at NSS startup, edit the CTINSS00 parmlib member and specify TRACEOPTS OFF.

2. Start the network security services (NSS) server.

Step for enabling the CTRACE after the network security services server has started

Perform one of the following steps to enable the CTRACE after NSS has started.

• Issue the following console commands to enable the CTRACE to an internal buffer:

  TRACE CT,ON,COMP=SYSTCPNS,SUB=(nss_jobname)
  R xx,OPTIONS=(option[,option2...]),END

• Issue the following console commands to enable the CTRACE to an external writer:

  TRACE CT,WTRSTART=writer_proc
  TRACE CT,ON,COMP=SYSTCPNS,SUB=(nss_jobname)
  R xx,OPTIONS=(option[,option2...]),WTR=writer_proc,END

Step for disabling the CTRACE after the network security services server has started

Perform one of the following steps to disable the CTRACE after NSS has started.

• Issue the following console commands to disable the CTRACE to an internal buffer:

  TRACE CT,OFF,COMP=SYSTCPNS,SUB=(nss_jobname)

• Issue the following console commands to disable a CTRACE to an external writer:

  TRACE CT,OFF,COMP=SYSTCPNS,SUB=(nss_jobname)
  TRACE CT,WTRSTOP=writer_proc

Step for displaying the CTRACE status

To display the CTRACE status, issue the following console command:

D TRACE,COMP=SYSTCPNS,SUB=(nss_jobname)
Enabling CTRACE after network security services server initialization

After NSS initialization, you must use the TRACE CT command to change the component trace options. Each time a new component trace is initiated, all prior trace options are turned off and the new options are put into effect. You can specify the trace options with or without the parmlib member. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47.

Formatting network security services server trace records

You can format component trace records using IPCS panels or a combination of the IPCS panels and the CTRACE command, either from a dump or from external-writer files. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47.

Enter any combination of values as options to filter the CTRACE entries. The options must be entered using the following format:

`TYPE(option[,option]...)`

You can use any of the options listed in Table 25 on page 368 except ALL and MINIMUM.
Chapter 11. Diagnosing dynamic VIPA and sysplex problems

This topic presents diagnostic information for dynamic virtual IP address (DVIPA) and sysplex problems, and contains the following sections:

- “Overview of diagnosing sysplex distributor problems” on page 374
- “Steps for diagnosing sysplex problems” on page 374
- “Steps for diagnosing problems using DVIPAs in source IP address selection for TCP connections problems” on page 385
- “Steps for diagnosing problems with the SYSPLEX-wide ephemeral port assignment for distributed DVIPAs” on page 387
- “Diagnosing problems with the SYSPLEX-wide ephemeral port assignment for EXPLICITBINDPORTRANGE processing” on page 389
- “Diagnosing SYSPLEX-wide security association (SWSA) problems” on page 393
- “Steps for diagnosing sysplex routing problems” on page 397
- “Steps for diagnosing Tier 1 z/OS sysplex distribution problems” on page 400
- “Steps for diagnosing Tier 1 non-z/OS sysplex distribution problems” on page 401

Overview of diagnosing sysplex distributor problems

Diagnosing sysplex distributor problems presents some unique challenges. Because a DVIPA can be associated with multiple stacks in a sysplex, determining where a problem is can be more difficult. You can use a combination of the Netstat command from the system console and display sysplex commands to provide a clear picture of the sysplex. Refer to the z/OS Communications Server: IP Configuration Guide for an introduction to sysplex distribution with virtual addressing.

You can collect Netstat information in the following ways:

- You can issue the z/OS UNIX netstat command from the z/OS UNIX shell.
- You can issue the NETSTAT command from TSO.
- You can issue the DISPLAY TCPIP,,NETSTAT command from the system console.

In the following list of activities, you can find steps to perform them in "Steps for diagnosing sysplex problems" on page 374:

- First, determine that all the stacks that you expect to be communicating are in the same subplex, if subplexing is being used. See step 1 on page 374.
- For problems where the actual DVIPAs defined on a stack are not what you expected, confirm the current definitions on a stack. See step 2.
- For Sysplex Distributor workload monitoring, use steps 7 and 10. If the output from these commands is not what you expected, see step 6 for an overall picture of all DVIPA activity in your sysplex.
- If the output from step 6 reveals an expected target stack not listed for a distributed DVIPA, perform step 3 on the target stack in question. This helps to identify configuration problems on that stack. Note what is required of target stacks. Also use step 11 to verify that a server application has indeed been activated and bound to the correct port.
To help follow the flow of packets into and throughout the sysplex, a CTRACE with options XCF, TCP, and SYSTCPDA on participating stacks is useful. Use these to:

- Identify the connection being received by the distributing stack
- Determine the stack to which the connection is forwarded
- Verify the connection being forwarded
- Determine the expected target stack receiving and processing the connection

After the connection has been established, subsequent packets can be followed in the same manner. When the connection is terminated, CTRACE records record target stacks, cleans up the connection, and notifies the distributing stack.

**Steps for diagnosing sysplex problems**

Perform the following steps to diagnose sysplex problems. The output is shown in the long, or IPv6-enabled, format.

1. If subplexing is being used, run the D XCF,GROUP MVS command to determine what groups are being used in the sysplex. Find all the group names with the format EZBTvtt.

   ```
   D XCF,GROUP
   IXC33II 10.11.09 DISPLAY XCF 637
   GROUPS(SIZE): COFVLFNO(2) EZBT1122(2) ISTXCF11(2) MVSIC94(1) MVS165(1) SYSGRS(2)
   EZBT1123(1) ISTCFS11(2) MVSIVC02(1) MVSIVC96(1) SYSJES(2) SYSIGW01(9)
   EZBT1121(2) MVSIVC11(1) MVS031(1) MVSJES(2) SYSIGW00(9)
   EZBT1122(2) MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1)
   EZBT1123(1) MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1)
   ISTXCF11(2) MVS165(1) SYSGRS(2) SYSENF(2) SYSJES(2) SYSIGW00(9)
   ISTCFS11(2) MVSIVC02(1) MVSIVC96(1) MVSJES(2) SYSIGW00(9)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SVSTTRC(2)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSENF(2)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSJES(2)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSIGW00(9)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSIGW01(9)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSJES(2)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSIGW00(9)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSJES(2)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSIGW00(9)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSJES(2)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSIGW00(9)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSJES(2)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSIGW00(9)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSJES(2)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSIGW00(9)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSJES(2)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1) SYSIGW00(9)
   MVSVIC02(1) MVSVIC11(1) MVSVIC94(1) MVSVIC96(1)
   ```

Run the D XCF,GROUP,groupid MVS command for each of the EZBTvtt format groups to find all the member stacks participating in each subplex group.

```
D XCF,GROUP,EZBT11CS
IXC332I 10.11.18 DISPLAY XCF 640
GROUP EZBT11CS: MVS165TCPCS1 MVS165TCPCS3 MVS165TCPCS11 MVS165TCPCS12
D XCF,GROUP,EZBT1121
IXC332I 10.11.18 DISPLAY XCF 640
GROUP EZBT1121: MVS165TCPCS1 MVS165TCPCS3 MVS165TCPCS11 MVS165TCPCS12
D XCF,GROUP,EZBT1122
IXC332I 10.11.18 DISPLAY XCF 640
GROUP EZBT1122: MVS165TCPCS1 MVS165TCPCS3 MVS165TCPCS11 MVS165TCPCS12
D XCF,GROUP,EZBT1123
IXC332I 10.11.18 DISPLAY XCF 640
GROUP EZBT1123: MVS165TCPCS1 MVS165TCPCS3 MVS165TCPCS11 MVS165TCPCS12
```

The member names listed are the MVS name concatenated with the TCP/IP stack name. Verify that all the TCP/IP stacks you expect to be communicating are all within the same subplex group.

If the TCP/IP stacks that you expect to be communicating use the same HiperSockets CHPID, verify that they have all specified the same IQDVLANID value. Issue the Netstat CONFIG/-f command for each stack and verify that the IQDVLANID value displayed in the Global Configuration Information section of the output is the same for each stack.

```
NETSTAT CONFIG
MVS TCP/IP NETSTAT CS V1R8 TCPIP Name: TCPCS 12:55:20
TCP Configuration Table:
DefaultRcvBufSize: 00016384 DefaultSndBufSize: 00016384
DefltMaxRcvBufSize: 00262144
```
MaxReTransmitTime: 120.000 MinReTransmitTime: 0.500
RoundTripGain: 0.125 VarianceGain: 0.250
VarianceMultiplier: 2.000 MaxSegLifeTime: 30.000
DefaultKeepAlive: 00000120 DelayAck: Yes
RestrictLowPort: Yes SendGarbage: No
TcpTimeStamp: Yes FinWait2Time: 600
TLS Yes
UDP Configuration Table:
DefaultRcvBufSize: 00065535 DefaultSndBufSize: 00065535
CheckSum: Yes
RestrictLowPort: Yes UdpQueueLimit: No

IP Configuration Table:
Forwarding: Yes TimeToLive: 000064 RsmTimeOut: 000060
IpSecurity: Yes
ArpTimeout: 01200 MaxRsmSize: 65535 Format: Short
IgRedirect: Yes SysplexRoute: No DoubleNop: No
StopClawEr: No SourceVipa: No
MultiPath: Conn PathMtuDesc: No DevRtryDur: 0000000090
DynamicXCF: Yes
   IpAddr/PrefFixLen: 193.9.200.3/28 Metric: 01
SecClass: 0
IqDioroute: Yes QDiOPriority: 1
TcpStackSrcVipa: 201.1.10.10

SMF Parameters:
Type 118:
   TcpInit: 01 TcpTerm: 02 FTPClient: 03
   TN3270Client: 00 TcpIpStats: 05
Type 119:
   TcpInit: No TcpTerm: No FTPClient: Yes
   TcpIpStats: Yes IfStats: Yes PortStats: Yes
   Stack: Yes UdpTerm: Yes TN3270Client: Yes

Global Configuration Information:
TcpIpStats: Yes ECSALimit: 0002047K PoolLimit: 2096128K
MlsChkTerm: No XCFGRPID: 11 IqDVLANID: 27
Sysplex Monitor:
   TimerSecs: 60 Recovery: Yes DelayJoin: No AutoRejoin: Yes
   MonIntf: No Dynroute: NO
Network Monitor Configuration Information:
PktTrcSrv: Yes TcpCnnSrv: Yes MinLifTim: 3 SmfSrv: Yes

Data Trace Setting:
   JobName: * TrRecCnt: 00000009 Length: FULL
   IpAddr: * SubNet: *

2. Run the Netstat VIPADCFG/-F display command on the distributing stack to confirm that it is configured to distribute the DVIPA and how it is to be distributed. If the DVIPA has been deactivated, the deactivated configuration definitions are displayed under the heading DEACTIVATED DYNAMIC VIPA INFORMATION.

   - Figure 31 on page 376 shows that the TCP/IP identified by TCPCS was configured to distribute DVIPAs. Workload for the first DVIPA, 201.2.10.11 ports 20 and 21, is being distributed to all stacks in the sysplex including TCPCS itself; the configured distribution method is SERVERWLM.
   - Workload for 201.2.10.12, ports 20 and 21, is being distributed only to the TCP/IP with dynamic XCF address 193.9.200.2.
   - Workload for 201.2.10.13 port 5000 is being distributed to all stacks using the TIMEDAFFfinity function.
   - Workload for IPv6 DVIPA 2001:0DB8:1::1, port 6000 is being distributed to all stacks; the configured distribution method is SERVERWLM.
- The DVIPA, 201.2.10.23, port 4000, was configured to be distributed to all stacks in the sysplex. Because the DVIPA has been deactivated on this stack, it is not currently being distributed by this stack.

D TCPIP,TCPCS,NET,VIPADCFG
EZD0101I NETSTAT CS V1R7 TCPCS 876
DYNAMIC VIPA INFORMATION:
VIPA BACKUP:
  IPADDR/PREFIXLEN: 201.2.10.21
    RANK: 000080 MOVEABLE: SRVMGR:
  IPADDR/PREFIXLEN: 201.2.10.22
    RANK: 000080 MOVEABLE: SRVMGR:
VIPA DEFINE:
  IPADDR/PREFIXLEN: 201.2.10.11/28
    MOVEABLE: IMMEDIATE SRVMGR: NO
  IPADDR/PREFIXLEN: 201.2.10.12/28
    MOVEABLE: IMMEDIATE SRVMGR: NO
  IPADDR/PREFIXLEN: 201.2.10.13/28
    MOVEABLE: IMMEDIATE SRVMGR: NO
INTFNAME: DVIPA1
  IPADDR: 2001:0DB8:1::1
    MOVEABLE: IMMEDIATE SRVMGR: N/A
VIPA DISTRIBUTE:
  DEST: 201.2.10.11..20
    DESTXCF: ALL
    SYSPT: NO TIMAFF: NO FLG: SERVERWLM
  DEST: 201.2.10.11..21
    DESTXCF: ALL
    SYSPT: NO TIMAFF: NO FLG: SERVERWLM
  DEST: 201.2.10.12..20
    DESTXCF: 193.9.200.2
    SYSPT: NO TIMAFF: NO FLG: BASEWLM
  DEST: 201.2.10.12..21
    DESTXCF: 193.9.200.2
    SYSPT: NO TIMAFF: NO FLG: BASEWLM
  DEST: 201.2.10.13..5000
    DESTXCF: ALL
    SYSPT: NO TIMAFF: NO FLG: BASEWLM
DESTINF: DVIPA1
  DEST: 2001:0DB8:1::1..6000
    DESTXCF: ALL
    SYSPT: NO TIMAFF: NO FLG: SERVERWLM
Deactivated Dynamic VIPA Information:
VIPA Define:
  IpAddr/PrefixLen: 201.2.10.23/28
    Moveable: Immediate SrvMgr: No
VIPA Distribute:
  Dest: 201.2.10.23..4000
    DestXCF: ALL
    SysPtt: No TimAff: No Flg: BaseWLM

Figure 31. Netstat VIPADCFG/-F example

3. Run the display Netstat CONFIG/-f command on the distributing stack and all target stacks to confirm that the correct IPCONFIG and IPCONFIG6 options have been specified.

Specify SYSPLEXROUTING on the distributor and all target stacks in order to get WLM-based distribution. Verify that DYNAMICXCF was specified on the distributor and all target stacks.

Figure 32 on page 377 shows the output of this command for the distributing TCP/IP:
Run the display command `D WLM,SYSTEMS` on the distributing stack and all targets stack to confirm that WLM is active. For more information about the DISPLAY command, refer to [z/OS MVS System Commands](#) on page 378 shows an example:
4. Run the display Netstat VIPADYN/-v command on the distributing stack to verify that the DVIPA status is ACTIVE and the distribution status is DIST or DIST/DEST. The deactivated DVIPA 203.2.10.23 do not appear in this display. Figure 34 shows an example:

5. Run display command Netstat VIPADYN/-v on the target stacks to verify that they have activated the distributed DVIPA and have it designated as a DEST. In this case, TCPCS2 has designated the distributed DVIPAs as DEST and TCPCS2 is a backup stack for several DVIPAs (status and origin show backup). Figure 35 on page 379 shows an example:
6. Run the Sysplex VIPADyn command from any stack in the sysplex to get a
global view of how and where DVIPAs are defined within the sysplex and
what their status is on each stack. Deactivated DVIPA configurations do not
appear in this display. Figure 36 on page 380 shows the following:

- Which TCP/IPs own distributed DVIPAs, DIST field=BOTH or DIST
- Which TCP/IPs have been made targets, DIST field = DEST
- The status of all other DVIPAs in this sysplex
7. Run the Netstat VDPT/-O command on the distributing stack to confirm that there are target stacks available with server applications ready.

With the keyword DETAIL, you can also see the following fields. For a complete DETAIL display example, see z/OS Communications Server: IP System Administrator's Commands.

- Raw (before normalization) WLM composite weight
- The original CP, zAAP, and zIIP weights, and the proportioned CP, zAAP, and zIIP weights that were used to determine the raw WLM composite weight. The original CP, zAAP, and zIIP weights, and the proportioned CP, zAAP, and zIIP weights are for only SERVERWLM and BASEWLM distribution algorithms.
- The target server connection responsiveness factors that make up the TSR and the current WLM or QOS weight for each service level mapping to a

Figure 36. Sysplex VIPADyn example
DVIPA or port entry for each target stack (each DESTXCF ADDR). Refer to the
z/OS Communications Server: IP User’s Guide and Commands
for more information.

This display shows only target stacks that are currently up and have joined
the sysplex. If there are fewer entries than what resulted from the display
command d tcpip,net,tipadcfg, the missing entries might be for target
stacks that are not yet up, or for stacks that are already up now, but that do
not specify the expected dynamic XCF address. Figure 37 on page 383 shows
an example:

D TCPIP,TCPCS,NET,VDPT,DETAIL

EZD0101I NETSTAT CS V1R7 TCPCS 010
DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:201.2.10.11..20
DESTXCF:193.9.200.2
TOTALCONN:0000000959 RDY:001 WLM:11 TSR: 79
FLG:SERVERWLM
  TCSR: 100 CER: 75 SEF: 79
  Weight 44
    Raw  CP: 44 zAAP: 00 zIIP: 00
    Proportional CP: 44 zAAP: 00 zIIP: 00
    QosPlcAct:*DEFAULT*
    W/Q:01

DEST:201.2.10.11..20
DESTXCF:193.15.1.1
TOTALCONN:0000000330 RDY:001 WLM:9 TSR: 68
FLG:SERVERWLM
  TCSR: 87 CER: 75 SEF: 79
  Weight 36
    Raw  CP: 50 zAAP: 00 zIIP: 34
    Proportional CP: 05 zAAP: 00 zIIP: 31
    QosPlcAct:*DEFAULT*
    W/Q:01

DEST:201.2.10.11..20
DESTXCF:193.15.3.1
TOTALCONN:0000000315 RDY:001 WLM:15 TSR: 100
FLG:SERVERWLM
  TCSR: 100 CER: 100 SEF: 100
  Weight 60
    Raw  CP: 60 zAAP: 00 zIIP: 00
    Proportional CP: 60 zAAP: 00 zIIP: 00
    QosPlcAct:*DEFAULT*
    W/Q:01

DEST:201.2.10.11..21
DESTXCF:193.9.200.2
TOTALCONN:0000000021 RDY:001 WLM:15 TSR: 100
FLG:SERVERWLM
  TCSR: 100 CER: 100 SEF: 100
  Weight 60
    Raw  CP: 60 zAAP: 00 zIIP: 00
    Proportional CP: 60 zAAP: 00 zIIP: 00
    QosPlcAct:*DEFAULT*
    W/Q:01

DEST:201.2.10.11..21
DESTXCF:193.15.1.1
TOTALCONN:0000000008 RDY:001 WLM:11 TSR: 78
FLG:SERVERWLM
  TCSR: 99 CER: 99 SEF: 80
  Weight 44
    Raw  CP: 44 zAAP: 00 zIIP: 00
    Proportional CP: 44 zAAP: 00 zIIP: 00
    QosPlcAct:*DEFAULT*
    W/Q:01

DEST:201.2.10.11..21
DESTXCF: 193.15.3.1
TOTALCONN: 0000000007 RDY: 001 WLM: 10 TSR: 94
FLG: SERVERWLM
  TCSR: 97 CER: 98 SEF: 97
  Weight 40
    Raw CP: 40 zAAP: 00 zIIP: 00
    Proportional CP: 40 zAAP: 00 zIIP: 00
    QosPlcAct: *DEFAULT*
    W/Q: 01
DEST: 201.2.10.12..20
DESTXCF: 193.9.200.2
TOTALCONN: 0000000000 RDY: 001 WLM: 03 TSR: 99
FLG: BASEWLM
  TCSR: 100 CER: 99 SEF: 99
  Weight 12
    Raw CP: 20 zAAP: 11 zIIP: 00
    Proportional CP: 02 zAAP: 10 zIIP: 00
    QosPlcAct: *DEFAULT*
    W/Q: 01
DEST: 201.2.10.12..21
DESTXCF: 193.9.200.2
TOTALCONN: 0000000000 RDY: 001 WLM: 03 TSR: 100
FLG: BASEWLM
  TCSR: 100 CER: 100 SEF: 100
  Weight 12
    Raw CP: 12 zAAP: 00 zIIP: 00
    Proportional CP: 12 zAAP: 00 zIIP: 00
    QosPlcAct: *DEFAULT*
    W/Q: 01
DEST: 201.2.10.13..5000
DESTXCF: 193.9.200.2
TOTALCONN: 0000000000 RDY: 001 WLM: 03 TSR: 0
FLG: BASEWLM
  TCSR: 90 CER: 75 SEF: 0
  Weight 12
    Raw CP: 12 zAAP: 00 zIIP: 00
    Proportional CP: 12 zAAP: 00 zIIP: 00
    QosPlcAct: *DEFAULT*
    W/Q: 01
DEST: 201.2.10.13..5000
DESTXCF: 193.15.1.1
TOTALCONN: 0000000000 RDY: 001 WLM: 01 TSR: 27
FLG: BASEWLM
  TCSR: 100 CER: 27 SEF: 27
  Weight 04
    Raw CP: 04 zAAP: 00 zIIP: 00
    Proportional CP: 04 zAAP: 00 zIIP: 00
    QosPlcAct: *DEFAULT*
    W/Q: 01
DEST: 201.2.10.13..5000
DESTXCF: 193.15.3.1
TOTALCONN: 0000000000 RDY: 001 WLM: 01 TSR: 48
FLG: BASEWLM
  TCSR: 75 CER: 64 SEF: 64
  Weight 04
    Raw CP: 04 zAAP: 00 zIIP: 00
    Proportional CP: 04 zAAP: 00 zIIP: 00
    QosPlcAct: *DEFAULT*
    W/Q: 01
DESTXCF: DVIPA1
DEST: 201.2.10.5000
TOTALCONN: 0000000000 RDY: 001 WLM: 11 TSR: 79
FLG: SERVERWLM
  TCSR: 99 CER: 98 SEF: 80
  Weight 44
    Raw CP: 44 zAAP: 00 zIIP: 00
8. Examine the READY (RDY) count fields. The READY (RDY) count is the number of servers that are currently listening on the DVIPA and PORT specified in the DEST: field on the target stack that was identified by the DESTXCF address.

For servers that use more than one port, the RDY value reflects the port where a LISTEN is performed. For example, for FTP, the control connection port (port 21) is where the RDY count is usually greater than 0. If the ready count is not as expected, proceed to step 11 to verify whether any non-quiesced server is listening on the DPORT on the target stack. If there is a server listening on the target stack, verify that it has not been quiesced by a VARY TCPIP,,SYSPLEX,QUIESCE command. On the target stack, run the Netstat ALL/-A command and verify that the quiesced value is NO.

9. Check the TotalConn count to see the distribution history. This is a cumulative count of the number of connections that have been forwarded by the distributing stack to each target stack.

If the connections are not being distributed to the target stacks as expected and either the WLM field or the W/Q field contains 00, then consider the following:

- If using WLM to distribute connections based upon the workload of the target stacks, verify that all participating stacks (the distributor and all targets) have SYSPLEXROUTING specified. See step 3 for instructions for verifying this. Also, verify that WLM is configured and active on all participating stacks. See step 3.

- If the WLM configuration appears correct and BASEWLM is being used as the distribution method, consider whether the unexpected distribution results might be caused by the current workload on the target stacks.

If SERVERWLM is being used, consider whether the unexpected distribution results might be the result of how well the server is meeting the goals of its service class, and the amount of workload available on the system given the importance of its service class. Refer to the z/OS Communications Server: IP Configuration Guide for an overview of how WLM determines server recommendations and how they are used by TCP/IP. For more detailed information about sysplex routing services, refer to z/OS MVS Planning: Workload Management.
If some entries have a low TSR value, consider whether network or server performance problems might be affecting distribution. Examine the TCSR, CER, and SEF values in the DETAIL output for these entries.

- If the TCSR value is low, this indicates a connectivity problem between the sysplex distributor stack and the target stack for those particular DVIPA, Port, and Destination entries.
- If the SEF value is low, but the CER is not, then this indicates that the application on this target is having problems accepting new connections.
- If the SEF and CER values are low, then the target stack is having problems establishing connections with one or more clients.
- If all entries representing distribution to the same target are very low, or 0, this might indicate that the target stack is experiencing problems.
- If you used a VIPAROUTE definition to specify the route from the distributor to the target, check the specified route to verify that it is active.

- If SERVERWLM is being used as the distribution method and a server has a WLM weight of 0, verify that the server is using the appropriate WLM Policy and that the system is not too overloaded to enable the server to meet its policy goals. Refer to z/OS Communications Server: IP Configuration Guide for an overview of how WLM determines server recommendations and how they are used by TCP/IP. For more detailed information about sysplex routing services, refer to z/OS MVS Planning: Workload Management.

- If the unexpected distribution results have not yet been explained and Sysplex Distributor Performance Policies have been defined using Policy Agent, consider whether the distribution might be caused by two network performance issues (TCP retransmissions and timeouts).

- If Sysplex Distributor Routing Policies have been defined using Policy Agent, consider whether the definition of that policy is affecting the connection distribution. After determining which connections are not being distributed correctly, run D TCP,TCPCS,NET,VCRT,DETAIL (see step 10) to determine the policy action to which each connection maps. Look at the QoS weights for those policy actions in the VDPT DETAIL display to see whether they are unusually low. The Policy Agent log on the target stack can display for each DVIPA/Port the QoS service level fractions used to modify the QoS weight. It can also display the calculations that caused a QoS fraction to be set abnormally high (such as connection limit exceeded or throughput exceeded). See “Diagnosing Policy Agent problems” on page 657 for more information.

10. Run the Netstat VCRT/-V command on the distributing stack to check whether there are any active connections that are being routed by the distributor. If you run the command with the keyword DETAIL (d tcpip,tcpcs,net,vcrt,detail) you can see the policy rule and policy action that each connection maps to.

If the VCRT table shown in Figure 38 on page 385 is empty, then connection requests might not be reaching the distributor. Check for a routing problem from the client to the distributor.

If you see expected entries in the table, note the dynamic XCF address and proceed to step 11. Figure 38 on page 385 shows an example:
11. Go to the target stacks represented by the DESTXCF ADDR field in the VCRT or VDPT display and run the Netstat ALLCONN(/-a),IPA=201.2.10.12 display command to see the connections on the target stack. Figure 39 shows an example:

```
D TCPIP,TCPCS,NET,VCRT,DETAIL
EZD0101I NETSTAT CS V1R7 TCPCS 758
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST: 201.2.10.11..21
SOURCE: 203.110.1.1..1031
DESTXCF: 193.15.1.1
  POLICYRULE: *NONE*
  POLICYACTION: *NONE*
DEST: 201.2.10.12..21
SOURCE: 203.110.1.1..1033
DESTXCF: 193.9.200.2
DEST: 201.2.10.13..5000
SOURCE: 203.110.1.1..0
DESTXCF: 193.15.1.1
  CFGTIMAFF: 0045  TIMAFFCNT: 0000000002  TIMAFFLFT: 0000
DEST: 201.2.10.13..5000
SOURCE: 203.110.1.1..1029
DESTXCF: 193.15.1.1
  POLICYRULE: *NONE*
  POLICYACTION: *NONE*
DEST: 201.2.10.13..5000
SOURCE: 203.110.1.1..1030
DESTXCF: 193.15.1.1
  POLICYRULE: *NONE*
  POLICYACTION: *NONE*
```

Figure 38. d tcpip,tcpcs,net,vcrt,detail example

Tip: For a variety of reasons, the VCRT and ALLCONN displays might not match exactly. For example, with short-lived connections such as Web connections, an entry might show up in one display but be gone by the time the second display is run. Also, the distributing stack places an entry into the Dynamic VIPA Connection Routing Table when it first forwards a connection request. A busy server might reject these connection requests, and therefore cause a temporary mismatch in the two displays.

---

**Steps for diagnosing problems using DVIPAs in source IP address selection for TCP connections problems**

Investigating problems related to which source IP address was chosen for outbound TCP connections depends on which options you have configured.

1. If you are using the TCPSTACKSRCVIPA function, then run the Netstat CONFIG/-f command on the stack in question to verify that the sysplex-wide
dynamic source VIPA was configured as expected. In other words, verify that
IPCONFIG/IPCONFIG6 SOURCEVIPA is set to YES and that
IPCONFIG/IPCONFIG6 TCPSTACKSRCVIPA is specified with the correct
address or interface. If you are using TCPSTACKSOURCEVIPA with a
distributed DVIPA, run the Netstat CONFIG/-f command on the distributor
stack and on the target stacks. Figure 40 shows an example:

```
IP CONFIGURATION TABLE:
FORWARDING: YES  TIMETOLIVE: 00064  RSTM_TIMEOUT: 00060
ARPTIMEOUT: 01200  MAXRSM_SIZE: 65535  FORMAT:  LONG
IGREDIRECT: YES  SYSPLXROUT: YES  DOUBLENOP: NO
STOPCLAWER: NO  SOURCEVIPA: YES
MULTIPATH: NO  PATHMTUDSC: NO  DEVRTRYDUR: 000000090
DYNAMICXCF: YES
IPADDR/PREFIXLEN: 193.15.1.1/24  METRIC: 02
IQDITOROUTE: NO
TCPSTACKSRCVIPA: 203.15.1.1
```

Figure 40. Netstat CONFIG/-f example

```
IPV6 CONFIGURATION TABLE:
FORWARDING: YES  HOPLIMIT: 00255  IGREDIRECT: YES
SOURCEVIPA: YES  MULTIPATH: NO  ICMPERRLIM: 00003
IGRTRHOPLIMIT: NO
DYNAMICXCF: YES
IPADDR: 2001:0DB8::151:0
INTFID: 0006:0007:0008:0009
TCPSTACKSRCVIPA: DVIPA1
```

Figure 40. Netstat CONFIG/-f example

2. If you are using the SRCIP function to specify source IP addressing for
specified jobnames or destinations, then run the Netstat SRCIP/-J command to
display the SRCIP configuration. Verify that either the jobname for the
application performing the outbound CONNECT() or the destination address
for the CONNECT() matches an entry in the SRCIP configuration. See [z/OS
Communications Server: IP Configuration Reference](#) for the order of precedence
that is followed if an outbound connection matches more than one entry in the
SRCIP configuration.

If you have configured distributed DVIPAs on SRCIP rules and outbound
connections are failing with EADDRNOTAVAIL and JRSRCIPDistDVIPA
, EXPLICITBINDPORTRANGE processing is either not configured or not
working properly. See "Steps for diagnosing problems with
EXPLICITBINDPORTRANGE processing" on page 391 for more information.

D TCPIP,,N,SRCIP
EZD0101I NETSTAT CS V1R8 TCPCS 745
SOURCE IP ADDRESS BASED ON JOB NAME:
JOB NAME  TYPE  SOURCE
-------- ---- -----
USER*  IPV4  203.15.2.1
USER*  IPV6  2003::15:1:1

SOURCE IP ADDRESS BASED ON DESTINATION:
DESTINATION: 192.1.1.90
SOURCE: 203.15.2.2

DESTINATION: 2001:DB8:10::82:2:2
SOURCE: 2003::15:1:2
4 OF 4 RECORDS DISPLAYED
END OF THE REPORT

Figure 41. Netstat SRCIP/-J example
3. Create an outbound connection. Use the Netstat ALLConn/-a command to confirm that the correct source IP address was used.

**Tip:** TCPSTACKSOURCEVIPA and SRCIP specifications can be overridden. For example, a match to a SRCIP entry will override a TCPSTACKSOURCEVIPA specification. If your TCPSTACKSOURCEVIPA or SRCIP configuration is correct but you are not getting the expected source IP address, see the source IP address selection information in **z/OS Communications Server: IP Configuration Guide** for the hierarchy of the various ways that the source IP address of an outbound packet is determined.

---

### Steps for diagnosing problems with the SYSPLEX-wide ephemeral port assignment for distributed DVIPAs

Perform the following steps to diagnose problems with the SYSPLEXPORTS field setting.

1. Run the Netstat VIPADCFG/-F command on the distributor stack to confirm that SYSPLEXPORTS was specified for all distributed DVIPAs as expected.

```
D TCPIP,TCPSCS,TCPGW,TCPADCFG
EZD0101I NETSTAT CS V1R7 TCPSCS 862
DYNAMIC VIPA INFORMATION:
VIPA DEFINE:
  IPADDR/PREFIXLEN: 203.15.1.1/24
  MOVEABLE: IMMEDIATE SRVMGR: NO
  IPADDR/PREFIXLEN: 203.15.1.2/24
  MOVEABLE: IMMEDIATE SRVMGR: NO
  INTFNAME: DVIPA1
  IPADDR: 2001:0DB8:1::1
  MOVEABLE: IMMEDIATE SRVMGR: N/A
VIPA DISTRIBUTE:
  DEST: 203.15.1.1..4011
  DESTXCF: ALL
  SYSPT: YES TIMAFF: NO FLG: BASEWLM
  DEST: 203.15.1.2..245
  DESTXCF: ALL
  SYSPT: NO TIMAFF: NO FLG: BASEWLM
DESTINTF: DVIPA1
  DEST: 2001:0DB8:1::1
  DESTXCF: ALL
  SYSPT: YES TIMAFF: NO FLG: BASEWLM
```

*Figure 42. Diagnosing SYSPLEXPORTS problems*

In the preceding display, the distributed DVIPAs 203.15.1.1 and 2001:0DB8:1::1 were enabled with SYSPLEXPORTS(SYSPT is Yes), while 203.15.1.2 was not (SYSPT is NO).

2. Verify from the system log that the following message was issued:

```
IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBEPORT
```

If subplexing is being used, the message will be:

```
IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBEPORTvvtt
```

where **vv** is the VTAM subplex group ID and **tt** is the TCP/IP subplex group ID. If no VTAM subplex group ID was specified at VTAM startup, but a TCP/IP subplex group ID was specified on the GLOBALCONFIG statement in
the TCP/IP Profile, then the structure name is EZBEPORT01tt. If a VTAM subplex group ID was specified, but no TCP/IP subplex group ID was specified, then the structure name is EZBEPORTvv.

If this message was not issued and Netstat VIPADCFG/-F shows that the SYSPLEXPORTS field was specified, refer to z/OS Communications Server: SNA Network Implementation Guide for more information about defining EZBEPORTvvtt with the coupling facility.

3. Bind to an ephemeral port and then create an outbound connection with the source IP address of the SYSPLEXPORTS distributed DVIPA. Do the following to verify SYPLEXPORTS is working correctly:
   a. Issue Netstat ALLConn/-a to verify the connection on the target stack.
   b. Issue Netstat VCRT/-V to confirm that the distributing stack is aware of the connection.
   c. Issue the DISPLAY VTAM NET,STATS command, specifying the full name of the EZBEPORT structure, to confirm that the coupling facility is managing ports for this distributed DVIPA. For ephemeral ports, the coupling facility assigns a block of 64 ports to the TCP/IP stack. For example:
Diagnosing problems with the SYSPLEX-wide ephemeral port assignment for EXPLICITBINDPORTRANGE processing

This section contains two sub-sections:

D NET,STATS,TYPE=CFS,STRNAME=EZBEPORT1121,LIST=ALL,SCOPE=ALL
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = STATS,TYPE=CFS 180
IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBEPORT1121
IST1797I STRUCTURE TYPE = LIST
IST1517I LIST HEADERS = 1024 - LOCK HEADERS = 1024
IST1373I STORAGE ELEMENT SIZE = 256
IST924I -------------------------------------------------------------
IST1374I CURRENT MAXIMUM PERCENT
IST1375I STRUCTURE SIZE  8192K 15104K *NA*
IST1376I STORAGE ELEMENTS  128 22400 0
IST1377I LIST ENTRIES  5 700 0
IST924I -------------------------------------------------------------
ISTREW1I EXPLICITBINDPORTRANGE - START: 50000 END: 50255
IST1823I LIST DVIPA SYSNAME TCPNAME # ASSIGNED PORTS
IST1824I 0 EXPLICITBINDPORTRANGE 64
IST1825I VIC015 TCPCS 64
IST1826I PORTS: 50000 50001 50002 50003 50004 50005
IST1827I 50006 50007 50008 50009 50010 50011
IST1827I 50012 50013 50014 50015 50016 50017
IST1827I 50018 50019 50020 50021 50022 50023
IST1827I 50024 50025 50026 50027 50028 50029
IST1827I 50030 50031 50032 50033 50034 50035
IST1827I 50036 50037 50038 50039 50040 50041
IST1827I 50042 50043 50044 50045 50046 50047
IST1827I 50048 50049 50050 50051 50052 50053
IST1827I 50054 50055 50056 50057 50058 50059
IST1827I 50060 50061 50062 50063
IST1824I 1 203.15.1.1 64
IST1825I VIC015 TCPCS 64
IST1826I PORTS: 1024 1025 1026 1027 1028 1029
IST1827I 1030 1031 1032 1033 1034 1035
IST1827I 1036 1037 1038 1039 1040 1041
IST1827I 1042 1043 1044 1045 1046 1047
IST1827I 1048 1049 1050 1051 1052 1053
IST1827I 1054 1055 1056 1057 1058 1059
IST1827I 1060 1061 1062 1063 1064 1065
IST1827I 1066 1067 1068 1069 1070 1071
IST1827I 1072 1073 1074 1075 1076 1077
IST1827I 1078 1079 1080 1081 1082 1083
IST1827I 1084 1085 1086 1087
IST1824I 2 2001:0DB8:1::1 64
IST1825I VIC015 TCPCS 64
IST1826I PORTS: 1024 1025 1026 1027 1028 1029
IST1827I 1030 1031 1032 1033 1034 1035
IST1827I 1036 1037 1038 1039 1040 1041
IST1827I 1042 1043 1044 1045 1046 1047
IST1827I 1048 1049 1050 1051 1052 1053
IST1827I 1054 1055 1056 1057 1058 1059
IST1827I 1060 1061 1062 1063 1064 1065
IST1827I 1066 1067 1068 1069 1070 1071
IST1827I 1072 1073 1074 1075 1076 1077
IST1827I 1078 1079 1080 1081 1082 1083
IST1827I 1084 1085 1086 1087
IST314I END

Figure 43. VTAM NET,STATS example

---

**Diagnosing problems with the SYSPLEX-wide ephemeral port assignment for EXPLICITBINDPORTRANGE processing**

This section contains two sub-sections:
Steps for determining an optimal range for the EXPLICITBINDPORTRANGE parameter

1. Change your existing configuration to specify the GLOBALCONFIG EXPLICITBINDPORTRANGE parameter on all stacks that you anticipate will be participating in explicit bind port range processing. As a guideline, the port range size should be at least large enough to allow for 2 blocks of ports to be in use by each participating TCP/IP stack (128 * number of TCP/IP stacks using the explicit bind port range). If you are using a SRCIP block, do not initially make the change to use distributed DVIPAs on the DESTINATION rules.

2. Start all of the stacks, servers and clients to reach the typical steady state connection load for your sysplex environment. All connections from sockets that were explicitly bound to the IPv4 address, INADDR_ANY, or the IPv6 unspecified address (in6addr_any) and port 0 will use ports from the new range.

3. Periodically check to determine how many ports from this new range are in use by issuing a D NET,STATS,TYPE=CFS,STRNAME=EZBEPORTvvtt,LIST=0 command (See Figure 44).

4. Check for message EZD1296 which will be issued if local ephemeral ports were used for connections because no explicit bind ports were available from the active EXPLICITBINDPORTRANGE parameter.

5. If message EZD1296 is not issued and if the number of allocated ports is consistently less than the total port range, then proceed to the next step, otherwise:
   a. Change the GLOBALCONFIG EXPLICITBINDPORTRANGE parameter to use a larger explicit bind port range; as a guideline increase the size by at least 64 for each participating stack (64 multiplied by the number of TCP/IP stacks)
   b. Issue a VARY TCPIP,OBEYFILE command to change the range on each of the stacks and return to step 3 above.
6. Change your SRCIP block to use distributed DVIPAs on DESTINATION rules.

**Steps for diagnosing problems with EXPLICITBINDPORTRANGE processing**

If you have configured distributed DVIPAs on SRCIP rules and outbound connections are failing with EADDRNOTAVAIL and JRSCIPDistDVIPA, EXPLICITBINDPORTRANGE processing is not working. There are several possible reasons for this to occur:

- EXPLICITBINDPORTRANGE parameter is not configured on the stack.
- EXPLICITBINDPORTRANGE parameter is configured but:
  - The stack did not connect to the EZBEPORT structure in the coupling facility
  - The stack has lost access to the EZBEPORT structure
  - The stack is running in a CINET environment with more than 1 stack and stack affinity was not established
  - There are no avail ports in the range (range is exhausted)
  - The application bound explicitly to an ephemeral port (equal to or greater than 1024) that is not reserved for this job by the PORT or PORTRANGE profile statement

Use the following steps to determine and correct the problem:

1. Issue the D TCPIP, tcp_stackname, SYSPLEX, PORTS command to determine the configured EXPLICITBINDPORTRANGE value for this stack and the active EXPLICITBINDPORTRANGE value in the sysplex (or subplex). If the command response indicates "No EXPLICITBINDPORTRANGE is configured on this stack", see z/OS Communications Server: IP Configuration Reference under the section for the GLOBALCONFIG statement, for information on enabling the EXPLICITBINDPORTRANGE.

   **Tip:** If the active port range does not match the configured port range for this stack, it means that another stack that was started after this stack had a different range defined in the GLOBALCONFIG EXPLICITBINDPORTRANGE parameter, or a VARY OBEYFILE command was processed on another stack that specified a GLOBALCONFIG EXPLICITBINDPORTRANGE parameter with a different range. You should try to ensure that all stacks participating in EXPLICITBINDPORTRANGE parameter processing specify the same port range. This can be done by specifying the GLOBALCONFIG EXPLICITBINDPORTRANGE statement in a file that is included in each stack’s TCP/IP profile using an INCLUDE statement.

2. Verify from the system log that the following message was issued:
   - IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBEPORT
   - If subplexing is being used, the message will be:
     IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBEPORTvv
     The vv value is the VTAM subplex group ID and the tt value is the TCP/IP subplex group ID. If no VTAM subplex group ID was specified at VTAM startup, but a TCP/IP subplex group ID was specified on the GLOBALCONFIG statement in the TCP/IP profile, then the structure name is EZBEPORT01tt. If a VTAM subplex group ID was specified, but no TCP/IP subplex group ID was specified, then the structure name is EZBEPORTvv.
   - If this message was not issued and D TCPIP, SYSPLEX, PORTS shows that an EXPLICITBINDPORTRANGE parameter was configured, refer to z/OS
Communications Server: SNA Network Implementation Guide for more information about defining EZBEPORTvtt with the coupling facility.

- If the IST1370I message was issued (the stack did connect to the EZBEPORT structure), and D TCPIP,SYSPLEX,PORTS shows that an EXPLICITBINDPORTRANGE parameter was configured but that no active EXPLICITBINDPORTRANGE is available from this stack, the stack may have lost connectivity to the EZBEPORT structure either as a result of a structure rebuild or a structure disconnect. In the console log, check for any failure or rebuild messages referencing the EZBEPORT structure. If a structure rebuild was in process for the EZBEPORT structure in use by this stack, wait for the rebuild to complete. If VTAM lost connectivity to the structure, issue the VARY NET,CFS,ACTION=CONNECT,STRNAME=structure_name command to re-establish connectivity to the structure.

3. Bind to the IPv4 address, INADDR_ANY, or the IPv6 unspecified address (in6addr_any) and port 0. Issue the D NET,STATS,TYPE=CFS,STRNAME=EZBEPORTvtt,LIST=0 command to confirm that the coupling facility has ports in the EXPLICITBINDPORTRANGE parameter allocated for the stack on which you issued the bind, and issue the Netstat ALLConn/-a command to determine if the port assigned to your application for this bind was one in the explicit bind port range.

   ```
   netstat -a -p tcp/s1
   MVS TCP/IP NETSTAT CS V1R9  TCPIP Name: TCPCS1  13:21:04
   User Id Conn State
   ------- ---- ----- 
   BPX0INIT 00000017 Listen
   Local Socket: 0.0.0.0..10007
   Foreign Socket: 0.0.0.0..0
   USER11 0000002B Closed
   Local Socket: 0.0.0.0..50001
   Foreign Socket: 0.0.0.0..0
   USER11 00000021 Closed
   Local Socket: 0.0.0.0..50000
   Foreign Socket: 0.0.0.0..0
   SYSLOGD8 00000018 UDP
   Local Socket: ::..514
   Foreign Socket: *.*
   ```

   If a port from the explicit bind port range was not allocated, check if you are running in a CINET environment in which more than one TCP/IP stack is being managed by CINET and stack affinity has not been established. Explicit bind port range processing is not supported in such a configuration.

4. Check the system log for message "EZD1296I EXPLICITBINDPORTRANGE exhausted" which indicates that the number of ports in the EXPLICITBINDPORTRANGE parameter is not large enough. The coupling facility was unable to allocate a port from this range and a stack ephemeral port was allocated instead. This may be a temporary situation because EBPR ports are eventually returned to the coupling facility after sockets bound to them are closed.

   **Tip:** Message EZD1296I is not issued more than once every 5 minutes. If this message is issued multiple times, you should consider enlarging the number of ports for the EXPLICITBINDPORTRANGE parameter. See "Steps for determining an optimal range for the EXPLICITBINDPORTRANGE parameter" on page 390 for more information.

5. Issue the Netstat ALLCONN/-a command to display the local socket IP address and port the application is bound to, and Netstat PORTLIST/-o
command to display the ports that are reserved. If you want to add the applications local port to the list of reserved ports, use the PORT or PORTRANGE profile statement.

Diagnosing SYSPLEX-wide security association (SWSA) problems

This section describes methods for diagnosing SWSA problems.

Steps for diagnosing sysplex-wide security association (SWSA) problems

A stack that is configured with the IPSECURITY keyword on the IPCONFIG statement is referred to as an IPSECURITY stack. A stack that is configured with the FIREWALL keyword on the IPCONFIG statement is referred to as a FIREWALL stack. Note that a TCPIP stack configured for FIREWALL must be at a V1R7 or earlier level. The SWSA environment can be comprised of a mix of IPSECURITY and FIREWALL stacks. Refer to z/OS Communications Server: IP Configuration Guide for information about configuring IP security policy on an IPSECURITY stack. Refer to z/OS Integrated Security Services Firewall Technologies for information about configuring IP security policy on a FIREWALL stack.

Use the following information to aid with diagnosing Sysplex-wide Security Association (SWSA) specifically.

Before you begin: Ensure that you have consistent IPSec policies on all participating systems, which include the following:

- Distributing stacks, target stacks and backup stacks.
- Certificates identifying hosts must be available on all distributing and backup hosts. This is most easily accomplished by sharing the SAF certificate repository between the processors in the sysplex.

Perform the following steps to diagnose SWSA problems.

1. Code the DVIPSEC option on the owning and backup stacks to take advantage of SWSA. Do the following:
   - On IPSECURITY stacks, use the `ipsec -f` command to confirm that IPSECURITY was specified on the IPCONFIG statement and DVIPSEC was specified on the IPSEC statement.

   ```
   # ipsec -f disp
   ZCS V1R7 ipsec TCPIP Name: TCPCS1 Fri Jul 16 10:48:47 2004
   Primary: Filter Function: Display Format: Detail
   Source: Stack Profile Scope: Current TotAvail: 2
   Logging: No Predecap: No DVIPSec: Yes
   ```

   *Figure 45. ipsec -f example*

   - On FIREWALL stacks, use the `netstat config` command to confirm that FIREWALL and DVIPSEC were specified on the IPCONFIG statement.
2. Verify from the system log for the distributing and target stacks (for sysplex distribution of IPSec workload) and the primary and backup stacks (for dynamic tunnel recovery) that an IST1370I message like the following was issued:

**IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBDVIPA**

If subplexing is being used, the message is:

**IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBDVIPA vvtt**

where `vv` is the VTAM subplex group ID and `tt` is the TCP/IP subplex group ID. If no VTAM subplex group ID was specified at VTAM startup, but a TCP/IP subplex group ID was specified on the GLOBALCONFIG statement in the TCP/IP Profile, then the structure name is EZBDVIPA01tt. If a VTAM subplex group ID was specified, but no TCP/IP subplex group ID was specified, then the structure name is EZBDVIPAvv.

For SWSA functions to work correctly, the stacks involved must be connected to the EZBDVIPAvvtt coupling facility structure. If this message was not issued, verify the stack was configured with DVIPSEC using the `ipsec -f` command for an IpSecurity stack or the Netstat CONFIG/-f command for a Firewall stack on a V1R7 or earlier system. If DVIPSEC was specified, refer to [z/OS Communications Server: SNA Network Implementation Guide](http://www.ibm.com) for information about setting up the sysplex environment for VTAM function and defining EZBDVIPAvvtt with the coupling facility.

3. For sysplex distribution of IPSec traffic, the target stacks must have a copy of the dynamic tunnel, called a shadow tunnel, that matches the dynamic tunnel on the distributing stack. Do the following:
   a. Use the following command to verify that a dynamic tunnel is active on IP security distributing stacks:

```bash
D TCPIP,,NETSTAT,CONFIG
NETSTAT Config MVS TCP/IP onetstat
CS V1R5 TCPIP Name: TCPCS 18:14:48
IP Configuration Table:
  Forwarding: Yes TimeToLive: 00064 RsmTimeOut: 00060
  FireWall: Yes
  DVIPSec: Yes
```

*Figure 46. netstat,config example*
b. Use the following command to verify that a shadow tunnel is active on IP security target stacks:

```bash
# ipsec -y display
```

![Output of ipsec -y display command]

---

**Figure 47. ipsec -y example**

- Use the following command to verify that a shadow tunnel is active on IP security target stacks:
c. Use the following command to verify that a dynamic tunnel is active on firewall distributing stacks:

```
# fwdynconns cmd=listactive
2 203.15.1.1 203.110.1.1 inbound/outbound remote
```

d. Use the following command to verify that a shadow tunnel is active on firewall target stacks:

```
# fwdynconns cmd=listshadow
2 203.15.1.1 203.110.1.1 inbound only shadow
```

4. To confirm that the coupling facility has the information about the tunnels in the event a recovery is necessary, use the following VTAM command, specifying the full name of the EZBDVIPA structure:

```
d net,stats,type=cfs,strname=ezbdvipa1121,dvipa=203.15.1.1
```

The following output is displayed:

```
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = STATS,TYPE=CFS
IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBDVIPA1121
IST1371I STRUCTURE TYPE = LIST
IST1517I LIST HEADERS = 1024 - LOCK HEADERS = 0
```
Information about the dynamic tunnels that are used in SWSA is kept in the coupling facility structure in the event that a recovery of the tunnel is necessary. For example, the recovery information is used when a DVIPA is taken over by another stack in the sysplex.

For more information about DISPLAY STATS, refer to the z/OS Communications Server: SNA Operation.

For IPSec connections to continue functioning with that DVIPA, the tunnel has to be recovered by the same stack that took over the dynamic VIPA.

The list entry for the DVIPA (list 1 above) shows the system and stack for which the coupling facility is maintaining information about the tunnel.

Steps for diagnosing sysplex routing problems

Perform the following steps to diagnose sysplex routing problems:
1. Run the Netstat VIPADyn VIPAROUTE/-v VIPAROUTE command on the distributing stack to see what type of route is used for distributing packets to target stacks.

   If there is no VIPA ROUTE entry, IP packets that are distributed by Sysplex Distributor to target stacks use dynamic XCF interfaces. Use the Netstat ROUTe/-r command on the distributing stack to see other routing failure problems.

   If there is a VIPA ROUTE entry defined for a target stack and the RtStatus field shows `Active`, IP packets that are distributed by Sysplex Distributor to that target stack use the normal IP routing tables to determine the best available route.

   If there is a VIPA ROUTE entry defined for that target stack and the RtStatus field shows `Unavail`, the defined target IP address in the route entry is not available yet. This could be because the target stack is currently active, but the target IP address is not defined in that target stack. All packets to that target stack use dynamic XCF interfaces. This is likely to be a configuration error that should be investigated. EZD1173I is issued when the stack detects this problem.

   If there is a VIPA ROUTE entry defined for a target stack and the RtStatus field shows `Inactive`, no route exists to that target stack. Refer to Communications Server: IP System Administrator’s Commands for more information about the RtStatus field.

2. Run the Netstat ROUTe/-r command on the distributing stack to see details of the routing information. The following shows an example of this information.

```
   D TCPIP,TCPCS,NET,VIPADYN,VIPAROUTE
   EZD0101I NETSTAT CS V1R7 TCPCS
   VIPA ROUTE:
   DESTXCF: 193.1.3.94
   TARGETIP: 213.5.1.1
   RTSTATUS: ACTIVE
   DESTXCF: 193.1.4.94
   TARGETIP: 213.6.2.2
   RTSTATUS: INACTIVE
   DESTXCF: 2EC0::943:FF03
   TARGETIP: 1EC0::5:1:1
   RTSTATUS: ACTIVE
   DESTXCF: 2EC0::943:FF04
   TARGETIP: 1EC0::6:2:2
   RTSTATUS: INACTIVE
   4 OF 4 RECORDS DISPLAYED
```

Figure 49. Netstat VIPADyn example

<table>
<thead>
<tr>
<th>DESTINATION</th>
<th>GATEWAY</th>
<th>FLAGS</th>
<th>REFCNT</th>
<th>INTERFACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>193.1.1.94/32</td>
<td>0.0.0.0</td>
<td>H</td>
<td>000000</td>
<td>EZASAMEMVS</td>
</tr>
<tr>
<td>193.1.1.94/32</td>
<td>0.0.0.0</td>
<td>UH</td>
<td>000000</td>
<td>EZAXCFC6</td>
</tr>
<tr>
<td>193.1.1.94/32</td>
<td>0.0.0.0</td>
<td>UH</td>
<td>000000</td>
<td>EZAXCFC7</td>
</tr>
<tr>
<td>193.1.3.94/32</td>
<td>0.0.0.0</td>
<td>UHS</td>
<td>000000</td>
<td>EZAXCFC6</td>
</tr>
<tr>
<td>193.1.4.94/32</td>
<td>0.0.0.0</td>
<td>UHS</td>
<td>000000</td>
<td>EZAXCFC7</td>
</tr>
<tr>
<td>203.1.1.94/32</td>
<td>0.0.0.0</td>
<td>UH</td>
<td>000000</td>
<td>VIPLCB01015E</td>
</tr>
<tr>
<td>213.4.1.1/32</td>
<td>0.0.0.0</td>
<td>UH</td>
<td>000000</td>
<td>LTRLE1A</td>
</tr>
<tr>
<td>213.4.2.2/32</td>
<td>0.0.0.0</td>
<td>H</td>
<td>000000</td>
<td>LTRLE2A</td>
</tr>
<tr>
<td>213.5.1.1/32</td>
<td>0.0.0.0</td>
<td>UHZ</td>
<td>000001</td>
<td>LTRLE1A</td>
</tr>
<tr>
<td>213.6.2.2/32</td>
<td>0.0.0.0</td>
<td>HZ</td>
<td>000001</td>
<td>LTRLE2A</td>
</tr>
</tbody>
</table>
3. Run the Netstat VCRT/-V DETAIL command on the distributing stack to see the routing information for each connection. The following shows an example of this information.

D TCPIP,TCPCS,NET,VCRT,DETAIL
EZD0101I NETSTAT CS V1R7 TCPCS DYNAMIC VIPA CONNECTION ROUTING TABLE:
Dest: 203.38.1.1..801
    Source: 192.168.2.76..1029
    DestXCF: 193.1.3.94
    PolicyRule: *NONE*
    PolicyAction: *NONE*
    Intf: LTRLE1A
    VipaRoute: Yes    Gw: 0.0.0.0
Dest: 203.38.1.1..801
    Source: 192.168.2.76..1028
    DestXCF: 193.1.7.94
    PolicyRule: *NONE*
    PolicyAction: *NONE*
    Intf: EZASAMEMVS
    VipaRoute: No    Gw: 0.0.0.0
Dest: 203.38.1.2..9001
    Source: 192.168.2.76..1031
    DestXCF: 193.1.4.94
    PolicyRule: *NONE*
    PolicyAction: *NONE*
    Intf: LTRLE2A
    VipaRoute: Yes    Gw: 0.0.0.0
Dest: 203.38.1.2..9001
    Source: 192.168.2.76..1030
    DestXCF: 193.1.6.94
    PolicyRule: *NONE*
    PolicyAction: *NONE*
    Intf: EZASAMEMVS
    VipaRoute: No    Gw: 0.0.0.0
4 OF 4 RECORDS DISPLAYED
See "Routing failures" on page 752 for additional information about routing failures.

### Steps for diagnosing Tier 1 z/OS sysplex distribution problems

For an in depth explanation of distribution to Tier 1 z/OS targets, see the “Sysplex distribution optimizations for multi-tier z/OS workloads” section in z/OS Communications Server: IP Configuration Guide.

**For Tier 1 sysplex distribution**

On the Tier 1 distributor, run the Netstat VIPADCFG/-F DETAIL command to see the Tier 1 configuration statement information.

- In the VIPA DEFINE section of the report, verify the new flag field, 1, which indicates that this is a VIPA definition for a Tier 1 DVIPA.
- In the VIPA DISTRIBUTE section, verify the Tier 1 configuration parameters.
  - The flag fields show that the targets are Tier1, and that the distribution method uses Server-specific WLM weights (ServerWLM).
  - The group name is CICSGROUP. The group name is used when a Tier 2 sysplex distributor is being used to correlate Tier 1 targets with their Tier 2 targets. This is explained further in the following Tier 2 distribution.

**Dynamic VIPA Information:**

**VIPA Define:**
- IpAddr/PrefixLen: 203.1.181.10/24
- Moveable: Immediate
- SrvMgr: No
- Flg: 1

**VIPA Distribute:**
- Dest: 203.1.181.10..10000
- DestXCF: ALL
- SysPt: No
- TimAff: No
- Flg: ServerWLM Tier1
- OptLoc: No
- GrpName: CICSGROUP
- ProcXcost:
  - zAAP: 001
  - zIIP: 001
- ILWeighting: 0

Run the Netstat VDPT/-O report to see the current status of the tier 1 targets. Verify that weight recommendations are nonzero values and that there is a ready server for each tier 1 target. This information is updated in each polling interval. The ready count is 1 for each tier 1 target, which indicates that each has a server listening on port 10000.

**Dynamic VIPA Destination Port Table for TCP/IP stacks:**
- Dest: 203.1.181.10..10000
- DestXCF: 193.1.1.100
- TotalConn: 0000000000
- Rdy: 001
- WLM: 16
- TSR: 100
- Flg: ServerWLM, Tier1

**Tier 2 sysplex distribution**

- A Tier 1 target may distribute connections to Tier 2 targets using a Tier 2 sysplex distributor.
- In the VIPA DEFINE section of the report, the flag field, “2”, indicates that this is a VIPA definition for a Tier 2 DVIPA.
Each Tier 2 distributor will send the weight of each ready server to the Tier 1 distributor; the group name is used to correlate a group of Tier 1 targets with their Tier 2 targets.

OPTLOCAL has been configured (OptLoc: Yes). As connection requests received are received from the Tier 1 targets, Tier 2 distribution targets on the same stack as the Tier 1 target will be preferred.

On each Tier 2 distributor run the Netstat VIPADCFG/-F DETAIL report to verify the following:

- The same group name is used on the Tier 1 and corresponding Tier 2 VIPADISTRIBUTE statements. In the example display from the VIPA distribute section, the group name "CICSGROUP" matches the Tier 1 group name.
- OPTLOCAL has been configured

VIPA Define:
IpAddr/PrefixLen: 203.2.108.10/24
Moveable: Immediate SrvMgr: No Flg: 2

VIPA Distribute:
Dest: 203.2.108.10..10002
DestXCF: ALL
SysPt: No TimAff: No Flg: ServerWLM Tier2
OptLoc: Yes
GrpName: CICSGROUP
ProcXcost:
  zAAP: 001 zIIP: 001
ILWeighting: 0

On each tier 2 distributor, run the Netstat VDPT/-O report to see the status of each tier 2 target. In the example, there is a nonzero ready count of 1 for the server on destination XCF 193.1.1.108 and the normalized server-specific weight for this target is 16. There is not a ready server on the destination XCF 193.1.1.181. As the tier 1 distributor receives the tier 2 weights, it adds the weight of the corresponding XCF address. The tier 1 normalized weight for the server on destination XCF 193.1.1.108 is twice that of the server weight on destination XCF 193.1.1.181 because the tier 2 distributor did not have a ready server on destination XCF 193.1.1.181.

Dynamic VIPA Destination Port Table for TCP/IP stacks:
Dest: 203.2.108.10..10002
DestXCF: 193.1.1.108
TotalConn: 0000000000 Rdy: 001 WLM: 16 TSR: 100
Flg: ServerWLM, Tier2, OptLocal
Dest: 203.2.108.10..10002
DestXCF: 193.1.1.181
TotalConn: 0000000000 Rdy: 000 WLM: 00 TSR: 100
Flg: ServerWLM, Tier2, OptLocal

Steps for diagnosing Tier 1 non-z/OS sysplex distribution problems

For an in depth explanation of distribution to non-z/OS targets, see the Sysplex distribution with DataPower section in e/OS Communications Server: IP Configuration Guide.

For Tier 1 sysplex distribution

On the tier 1 distributor, run the Netstat VIPADCFG/-F DETAIL report to see the tier 1 configuration statement information.

- In the VIPA DEFINE section of the report, verify the new flag field, 1, which indicates that this is a VIPA definition for a tier 1 DVIPA.
- In the VIPA DISTRIBUTE section, verify the tier 1 configuration parameters.
- The flag fields show that the targets are Tier1, and that the distribution method is target controlled (TargCtrl). Target controlled distribution means that the tier 1 targets will send weight recommendations to the distributor. The other routing types supported are ROUNDROBIN and WEIGHTEDACTIVE.
- The routing type used to send packets to the non-z/OS tier 1 targets is GRE. GRE is the only type that is supported.
- When the RtgType is GRE, a control connection is established from the distributor to each target. In this example, the destination port (CtrlPort) that will be used for the control connection is 1702. The control connection will be used to receive:
  - Weight recommendations from the target (since the distribution method is TargCtrl) - the distributor will load balance across the tier 1 targets using these weights similar to how the distributor uses WLM weight recommendations to load balance across z/OS targets
  - Connection awareness information (since the routing type is GRE) - this information will indicate if the target has a ready server listening on port 9000, and will also be used to indicate when connections are established and terminated.
- The group name is CICSGROUP. The group name is used when a tier 2 sysplex distributor is being used to correlate tier 1 targets with their tier 2 targets. This is explained further in the following tier 2 distribution section.

Dynamic VIPA Information:
VIPA Define:
  IpAddr/PrefixLen: 9.42.130.251/24
  Moveable: Immediate SrvMgr: No Flg: 1

VIPA Distribute:
Dest: 9.42.130.251..9000
DestXCF: 9.42.105.53
SysPt: No TimAff: No Flg: TargCtrl Tier1
OptLoc: No
GrpName: CICSGROUP RtgType: GRE CtrlPort: 01702
Dest: 9.42.130.251..9000
DestXCF: 9.42.105.73
SysPt: No TimAff: No Flg: TargCtrl Tier1
OptLoc: No
GrpName: CICSGROUP RtgType: GRE CtrlPort: 01702
Dest: 9.42.130.251..9000
DestXCF: 9.42.103.215
SysPt: No TimAff: No Flg: TargCtrl Tier1
OptLoc: No
GrpName: CICSGROUP RtgType: GRE CtrlPort: 01702
Dest: 9.42.130.251..9000
DestXCF: 9.42.103.216
SysPt: No TimAff: No Flg: TargCtrl Tier1
OptLoc: No
GrpName: CICSGROUP RtgType: GRE CtrlPort: 01702

Run the Netstat ALLCONN/-a report to verify that a control connection is active to each non-z/OS target. The foreign socket destination IP address and destination port correspond to the DestXCF IP address (of the non-z/OS target) and the control port of 1702 from the VIPA distribute section. In this example report, there are four TCP connections in established state to each of the configured Tier 1 targets.

```
USER1@VIC018:/i> netstat -a
MVS TCP/IP NETSTAT CS V1R11
TCPIP Name: TCPCS  17:23:36
User Id Conn State
------- ---- -----
Run the Netstat VDPT/-O report to see the current status of the Tier 1 targets.
Verify the weight recommendations are non-zero and that there is a ready server for each Tier 1 target. This information is updated each polling interval.

- The T1Wt field is the weight recommendation reported by each target. In the example, all are non-zero.
- The ready count is 1 for each Tier 1 target indicating that each has a server listening on port 9000.
- The CWt field is a CPC weight. This field should be non-zero if Tier 2 distributors are configured. If there are no corresponding Tier 2 distribution statements, then this field will be zero. This is explained further in the following Tier 2 distribution section.

Dynamic VIPA Destination Port Table for non-z/OS targets:

<table>
<thead>
<tr>
<th>Dest: 9.42.130.251..9000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Addr: 9.42.103.215</td>
</tr>
<tr>
<td>TotalConn: 0000000000 Rdy: 001 Wt: 03 CWt: 032</td>
</tr>
<tr>
<td>Flg: TargCtrl</td>
</tr>
<tr>
<td>T1Wt: 420</td>
</tr>
<tr>
<td>ActConn: 0000000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest: 9.42.130.251..9000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Addr: 9.42.103.216</td>
</tr>
<tr>
<td>TotalConn: 0000000000 Rdy: 001 Wt: 05 CWt: 032</td>
</tr>
<tr>
<td>Flg: TargCtrl</td>
</tr>
<tr>
<td>T1Wt: 672</td>
</tr>
<tr>
<td>ActConn: 0000000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest: 9.42.130.251..9000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Addr: 9.42.105.53</td>
</tr>
<tr>
<td>TotalConn: 0000000000 Rdy: 001 Wt: 03 CWt: 064</td>
</tr>
<tr>
<td>Flg: TargCtrl</td>
</tr>
<tr>
<td>T1Wt: 420</td>
</tr>
<tr>
<td>ActConn: 0000000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest: 9.42.130.251..9000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Addr: 9.42.105.73</td>
</tr>
<tr>
<td>TotalConn: 0000000000 Rdy: 001 Wt: 03 CWt: 064</td>
</tr>
<tr>
<td>Flg: TargCtrl</td>
</tr>
<tr>
<td>T1Wt: 415</td>
</tr>
<tr>
<td>ActConn: 0000000000</td>
</tr>
</tbody>
</table>

**Tier 2 sysplex distribution**

- A Tier 1 target may distribute connections to Tier 2 targets on the same CPC using a Tier 2 sysplex distributor configured with CPCSCOPE. The Tier 2 sysplex distributor will only load balance to z/OS targets that are on the same CPC.
- In the VIPA DEFINE section of the report, the new flag field, 2, indicates that this is a VIPA definition for a Tier 2 DVIPA. The flag, "C", indicates that the Tier 2 DVIPA is configured with CPCSCOPE.
Each Tier 2 distributor will send a combined CPC weight of all ready servers to the Tier 1 distributor; the group name is used to correlate a group of Tier 1 targets with their Tier 2 targets.

On each Tier 2 distributor run the Netstat VIPADCFG/-F DETAIL report to verify that the same group name is used on the Tier 1 and corresponding Tier 2 VIPADISTRIBUTE statements. In the example report from the VIPA distribute section, the group name "CICSGROUP" matches the Tier 1 group name.

A combined weight will be used to update a Tier 1 target with a matching group name if this Tier 2 distributor (and therefore its Tier 2 targets) are on the CPC used by that Tier 1 target. A VIPADEFINE CPCSCOPE definition is required to determine the CPC used by a Tier 1 target. If the IP address of the target is in the same subnet of this DVIPA, then its targets are on that CPC.

In the example report from the VIPA define section, DVIPA 9.42.103.17 is a CPCSCOPE DVIPA because the flag field is "C".

The subnet for this DVIPA is 9.42.103.xx/24. So all Tier 1 targets (from the Tier 1 Netstat VDPT/-O report above) in this subnet are using the same CPC as that stack.

The Tier 1 targets in this subnet are 9.42.103.215 and 9.42.103.216. So as combined weights are received from a Tier 2 distributor, with the same CPC ID as the stack in which the CPCSCOPE definition was defined, the Tier 1 combined weight fields (CWt) for those targets (in the CPCSCOPE subnet) are updated with those weights.

In this example, the CPCSCOPE DVIPA is configured on the same stack as the Tier 2 VIPADISTRIBUTE statement. So this Tier 2 distributor will be providing the combined weights for the non-z/OS targets in the CPCSCOPE DVIPA subnet, 9.42.103.215 and 9.42.103.216. Ideally the CPCSCOPE DVIPA would be defined on the same stack as the Tier 2 Distributor, and VIPABACKUP statements for these DVIPAs would be on the same stacks with the same ranks. However the CPCSCOPE DVIPA could be defined on any stack as long as it is in the same CPC as the Tier 2 distributor.

VIPA Define:
IpAddr/PrefixLen: 9.42.103.17/24
  Moveable: Immediate  SrvMgr: No  Flg: C
IpAddr/PrefixLen: 203.2.108.3/24
  Moveable: Immediate  SrvMgr: No  Flg: 2C

VIPA Distribute:
Dest: 203.2.108.3..11000
  DestXCF: ALL
  SysPt: No  TimAff: No  Flg: ServerWLM Tier2
  OptLoc: No
  GrpName: CICSGROUP

On each Tier 2 distributor, run the Netstat VDPT/-O DETAIL report to see the status of each Tier 2 target. In the example, there is a non-zero ready count of 1 for each server, and there is a non-zero normalized server-specific weight for each target.

The combined weight in the Tier 1 VDPT (CWt) displays for the Tier 1 targets on that CPC, 9.42.103.214 and 9.42.103.216 is the total weight of the Tier 2 ready servers, 32.

Dynamic VIPA Destination Port Table for TCP/IP stacks:
Dest: 203.2.108.3..11000
  DestXCF: 193.1.1.108
  TotalConn: 0000000000  Rdy: 001  WLM: 16  TSR: 100
  Flg: ServerWLM, Tier2
  TCSR: 100  CER: 100  SEF: 100
  Weight: 64
    Raw CP: 64 zAAP: 00 zIIP: 00
    Proportional CP: 63 zAAP: 00 zIIP: 00
Abnorm: 0000  Health: 100
ActConn:  0000000000
QosPclAct:  *DEFAULT*
W/Q: 16
Dest:  203.2.108.3..11000
DestXCF:  193.1.1.181
TotalConn:  0000000000  Rdy:  001  WLM:  16  TSR:  100
Flg:  ServerWLM, Tier2
TCSR:  100  CER:  100  SEF:  100
Weight:  64
  Raw CP:  64  zAAP:  00  zIIP:  00
  Proportional CP:  63  zAAP:  00  zIIP:  00
Abnorm:  0000  Health:  100
ActConn:  0000000000
QosPclAct:  *DEFAULT*
W/Q: 16
Chapter 12. Diagnosing access control problems

This topic describes selected procedures for TCP/IP Services component trace, packet trace, and Socket API trace.

This topic contains the following sections:
- "Overview of access control support"
- "Diagnosing multilevel security consistency check messages (EZD1215-EZD1234)" on page 409

Overview of access control support

Communications Server is a resource manager that provides access control support over many of its services.

This can be a powerful tool to prevent unwanted usage of communications services. At times, it might also prevent intended usage. TCP/IP uses SAF (Security Access Facility) interfaces to ask your installed security server access control questions.

Note: The examples and terminology in this topic assume you are using RACF. However, you can use any SAF-conforming security server.

Tip: The SAF interface allows security servers to return the following responses to access control questions:

- **Allow** User is permitted to resource with requested level of access.
- **Deny** User is not permitted to resource with requested level of access.
- **No decision** Class is not active or covering profile is not defined.

For many resources, TCP/IP allows access when a No decision is returned. RACF supports the No decision response. Some security server products do not support the No decision response. They always return Deny when a resource has no profile. If you are using one of these other security servers, you must define profiles for these resources to allow any user to use them.

TCP/IP creates resource names in the SERVAUTH class to represent the services it protects.

These resource names are comprised of the following tokens:
- The first token is always EZA or EZB.
- The second token represents the type of services.
- The third token is the eight-character MVS system image name.
- The fourth token is often the TCP/IP job name.

Additional tokens can be defined for more granularity on certain types of services. For more information about services that TCP/IP protects and the resource names used, refer to the security topic in "z/OS Communications Server: IP Configuration Guide".
You define RACF profiles in the SERVAUTH class to control access permissions to these resource names. A discrete profile has the same name as a resource and covers only that resource. A generic profile uses wildcard symbols to cover many resource names. The SERVAUTH class is a general resource class, so you use the RACF RDEFINE, RLIST, RALTER, RDELETE and PERMIT commands to manage these profiles. For more information, refer to z/OS Security Server RACF Security Administrator’s Guide and z/OS Security Server RACF Command Language Reference.

Except for a few documented cases, TCP/IP checks for READ access to resources. Users might be given access to a resource in several ways. A RACF profile defines universal access (UACC) that provides the default level of access for all users not explicitly named. Individual users and user groups might be given a different level of access, higher or lower, with the PERMIT command. Use the WHEN clause to define conditions that must be met before the specified access is granted.

Tip: The RACF WHEN(PROGRAM(....)) clause has restrictions on profiles in the SERVAUTH class. It can be ignored on some resource checks and should only be used for resource names that explicitly document support.

RACF can be configured to write audit messages to the console. The default for profiles in the SERVAUTH class is to write a message when access is denied. These messages indicate the user, resource name, profile name and access level requested. When you first put an access control policy in place, you might want to configure the profile to produce audit messages on successes as well. You might also want to configure the profile with the WARNING parameter. This causes RACF to write the audit failure messages and then return allow to the resource manager. This allows you to test the effectiveness of a proposed policy without impacting usage.

Tips:

- Some policy changes do not take effect until the next time a user logs on or starts a job. After changing the policy, the user might need to log off or a job might need to be canceled and restarted.
- TCP/IP caches results when it checks access to NETACCESS resources. This cache is purged when a NETACCESS statement is found in a file used with the VARY TCPIP, OBEYFILE command. It is also purged when an ENF signal is received from RACF indicating that the SERVAUTH class or SECLABEL class has been refreshed. If your security server does not produce this ENF signal then, after making policy changes, you must issue the VARY TCPIP, OBEYFILE command with a file containing the NETACCESS statement to cause TCP/IP to purge cached responses.

Several of the TCP/IP services that provide access control check socket calls made through several different interfaces. When access to a resource is denied, the errno returned is EACCES. The errno2 field provides additional information about the failure. Programs that provide diagnostic logs should include the errno2 field. For information on the contents of the value returned, refer to z/OS UNIX System Services Programming: Assembler Callable Services Reference.

Tip: Many C programs use the perror() or strerror() library service to display errors encountered. There is an environment variable _EDC_ADD_ERRNO2, which when set to 1, appends the current errno2 value to the end of the perror() string as shown below:

EDC5121I Invalid argument. (errno2=0x0C0F8402)
TCP/IP access control failures are recorded in the event trace (SYSTCPIP) for TCP/IP stacks with the ACCESS option.

**Diagnosing multilevel security consistency check messages (EZD1215-EZD1234)**

Secure communication in a multilevel secure environment requires configuration of several statements in the TCPIP.PROFILE and security server resource profiles in the SERVAUTH, SECLABEL and STARTED classes. Inconsistencies in this configuration can allow unintended communication or prevent intended communication. When the RACF MLACTIVE option is set, TCP/IP checks the TCPIP.PROFILE and security server resource profiles for consistency. Consistency checking occurs at TCP/IP initialization, when a VARY TCPIP,OBEYFILE command is processed and when RACF sends an ENF signal specifying that a RACLST REFRESH was done on the SERVAUTH or SECLABEL class.

TCP/IP writes an informational message to the job log for each inconsistency detected. If inconsistencies are found, a final message, EZD1217I, summarizing the number of problems found is written to the system console. You should check the job log for messages in the range EZD1219I-EZD1234I whenever message EZD1217I appears on the system console. You should correct your configuration as indicated by the job log messages until TCP/IP no longer detects any errors.

TCP/IP’s default behavior is to continue running when inconsistent security configurations are detected. If you plan to run in a multilevel-secure environment, it is recommended that you specify GLOBALCONFIG MLSCHKTERMINATE in your TCPIP.PROFILE when running production workloads and GLOBALCONFIG NOMLSCHKTERMINATE while you are making planned changes to your security environment.

**Steps for verifying the configuration**

**Before you begin:** Refer to [z/OS Communications Server: IP Configuration Guide](#) for information about networking in a multilevel-secure environment.

Perform the following steps to verify the configuration:

1. TCP/IP stack is running under the intended user ID. If the stack is a submitted job, check the USER= parameter on the job card. If the stack is a started procedure, check the $DATA segment of the profile in the STARTED class.

2. TCP/IP stack is running with the intended security label. If the stack is a submitted job, check the SECLABEL= parameter on the job card. If the stack is a started procedure or SECLABEL= was not specified on the job card, check the default security label in the USER profile. Verify that the user ID is permitted to the SECLABEL profile. If running with the RACF SECLBYSYSTEM option, verify that the security label is active on this system image.

3. TCP/IP stack recognizes the multilevel-secure environment. The TCPIP.PROFILE must contain a valid NETACCESS statement with the following:
   - INBound

Chapter 12. Diagnosing access control problems 409
• OUTBound
• At least one valid security zone definition

4. TCP/IP stack has the intended IP addresses defined. Verify the IP addresses on DEVICE and INTERFACE statements in the TCPIP.PROFILE. Verify the IP addresses on VIPADEFINE, VIPABACKUP, VIPARANGE and VIPADISTRIBUTE statements in the TCPIP.PROFILE. Verify that IP addresses are manually configured for IPv6 interfaces. Verify that the INTFID keyword is specified on all IPv6 interfaces. Verify that the IPADDR keyword is specified on all IPv6 interfaces that support autoconfiguration.

5. TCP/IP stack has IP addresses mapped into the intended network security zones. Verify that the base IP address, mask and zone name are correct on each line in NETACCESS statement in the TCPIP.PROFILE. Verify that these addresses are in security zones:
   • INADDR_ANY (IPv4 0.0.0.0/32, IPv6 ::/128)
   • LOOPBACK (IPv4 127.0.0.1/8, IPv6 ::1/128)
   • Any required Multicast (IPv4 224.0.0.0/4, IPv6 FF00::/8)

Tips:
• The console command D TCPIP,,N,ACC,NETW displays the current NETACCESS statement configuration. The SERVAUTH profile name covering the security zone resource name and the security label defined on that profile are also shown.
• The security zone that a given IP address is currently configured into is displayed by the console command D TCPIP,,N,ACC,NETW,ipaddress.

6. SERVAUTH resources are covered by the intended profile. The RACF command RLIST SERVAUTH resource_name AUTHUSER displays the discrete or generic profile that most closely matches the specified resource name. It also displays the universal access, the security label, the access list and the conditional access list for that profile.
Chapter 13. Diagnosing line print requester and daemon (LPR and LPD) problems

Line print requester (LPR) and line printer daemon (LPD) compose a functional unit in which the LPR client sends data files to a printer controlled by an LPD server. These files can be in ASCII form or extended binary-coded decimal interchange code (EBCDIC) form.

In most environments, customers have different types of LPR clients and LPD servers, running on platforms, such as MVS, OS/2®, AIX®, and UNIX. However, all print client and servers must follow the standards contained in RFC1179. Some clients and servers provide more than what is required by the RFC, while some clients and servers are restricted or limited, which can cause errors or require more configuration to work.

On platforms, such as MVS, UNIX, and AIX, you can start the LPR client program with command prompts, through batch (in MVS), or through shell scripts (in UNIX/AIX®). The MVS LPD server allocates temporary data sets to process incoming print requests from various clients. These data sets use the TCP/IP high level qualifiers (HLQs) or the prefix defined in the LPD server cataloged procedure.

The MVS LPD server can also act as a client when a remote print server is defined in the LPD configuration file as a service. In this case, when the LPD server receives an incoming print job, it opens a new connection through a client port, and sends the data to the remote print server. When a remote print server is used, LPD specifications, such as line size and page size, do not apply. Instead, the specifications of the remote server apply.

For information on configuring your LPD server, refer to the z/OS Communications Server: IP Configuration Reference. For information on using the client-related LPR, LPQ, and LPRM commands, refer to the z/OS Communications Server: IP User’s Guide and Commands.

Problems with the print function are usually easy to diagnosis if the problem is within the LPR client or the LPD server. More difficult problems can be encountered in the TCP/IP layer or in sockets. In addition, incorrectly built or defined translation tables can produce unpredictable results, such as abends, misprinted characters, and hang conditions (usually caused by delayed acknowledgments).

Diagnosing LPR client and LPD server problems

Problems with LPR and LPD generally fall into one of the following categories:

• Abends
• Timeouts, hangs, and waits
• Incorrect output

These categories are described in the following sections.
Abends

When an abend occurs during LPD processing, messages and other error-related information are sent to the MVS system console. If this information is insufficient to solve the problem, use the information provided in a dump. To produce a dump, code a SYSMDUMP DD or SYSABEND DD statement in the LPD cataloged procedure. If you do not do the coding before the abend occurs, code the statement after the abend, re-create the abend or wait for it to occur again. For information about analyzing dumps produced during LPD processing, refer to z/OS Problem Management.

It can also be helpful to obtain and analyze information from the following sources:
- LPD trace in the SYSPRINT data set
- Output of LPD started task
- System log (syslog)

Steps for diagnosing timeouts, hangs, and waits

Timeouts, hangs, and waits occur when the LPD server does not respond to client requests for a data packet, an acknowledgment that a data packet was received, or a reply to a command. Similarly, the LPD server can time out a connection if the LPR client does not respond.

Before you begin: Determine if one or more of the following problems caused a timeout, hang, or wait:
- Incorrect host name or IP address specified on the LPR command
- Malfunctioning remote server or remote host
- Problems with the network (for example, network congestion), bridge, gateway, or router in the routing path
- Problems with the device or channel attached to the host
- Corrupted TCP/IP address space
- Incorrectly built or defined translation tables
- Malfunctioning LPR client

Perform one or more of the following steps to diagnose timeouts, hangs, and waits.

1. Check to see if the target LPD print server is running, has enough paper, and is not jammed.

2. Check the LPR and LPD traces for possible error messages, or for the last activity performed by LPR or LPD (for example, waiting for a connection, port availability, or an acknowledgment). Be aware that when sending a print request to a remote printer through the LPD server, the LPR client can show a successful data transfer even though there might be a problem connecting to the remote printer.

3. Check the IP address or host name used with the LPR command.

4. Check the LPR, LPD, and packet traces. If the packet trace shows a problem during binding or connecting, then check the socket trace.

5. Verify that the translation tables are built correctly. Test them using the hlq.STANDARD.TCPXLBIN table supplied with TCP/IP.
Be aware that waits can occur because some LPD servers do not send acknowledgments until data is actually printed. In this situation, the LPR client does not show successful data transfer until it actually receives the acknowledgment.

**Incorrect output**
LPR problems with incorrect output usually fall into one of the following categories:
- Garbled data sent from the LPR client or received by the LPD server
- Truncated or missing print data
- LPR works with some options, but not others

These categories are described in the following sections.

**Steps for diagnosing garbled data**
Perform the following steps to diagnose garbled data problems.

1. Determine whether the binary option or the default EBCDIC was used when the data file was printed. If the binary option was used, the LPR client did not translate the data. If EBCDIC was used, check for erroneous control characters or conflicting combinations of options.

2. Check to see if other files print correctly from the same client and to the same server. Check to see if the problem file prints correctly to other servers.

3. Verify that the translate tables for the sender and receiver are reciprocals of each other. Determine which characters are consistently garbled and examine those entries in the tables. To determine the name of the translation table used by the LPR client, check the LPR messages issued at startup.

4. Check the IP packet trace to determine exactly what data was sent from the client and acknowledged by the LPD server.

5. If data shown in the IP packets from the LPR client to the server is correct, there might be an error on the server or printer. Check the server traces and setup on the printer or LPD server. Some servers require certain printer names or options to be specified on the LPR (lp from omvs) commands.

**Steps for diagnosing truncated or missing print data**
Perform one or more of the following steps to diagnose truncated or missing print data.

1. Check to see if the value for the record length is valid. The value is specified using the WIDTH option and variable on the LPR command.

2. If MVS displays truncated records, check the value of the LINESIZE option on the SERVICE statement in the LPD configuration file.
3. If you use the FILTER L or FILTER R options on the LPR command, check to see if the control characters on the first column of the source file are valid. LPR issues a message indicating whether a record of data has been ignored.

4. Using a packet trace and the file size listed in the LPR trace control record, verify that the correct number of bytes were sent by the LPR client and received by the LPD server.

5. Check the LPD trace for error messages. Verify that the Job xxx Received and Job xxx Finish Printing messages were received.

6. If sending a print request to a remote printer through the LPD server, check the LPD trace to determine if all data were sent successfully to the remote printer. If not or if data are incorrect, check the printer for errors or restrictions on the type of data it supports (for example, postscript only, text only, and so on).

7. Check for partial temporary data sets and either rename them or delete them. The LPD server creates temporary data sets when connections are broken, and the server does not completely process a print job. (Depending on the LPR client, the server can requeue the job for printing at a later time.) When the connection is restored, the daemon checks for temporary data sets and processes them. After processing, they are erased.

The temporary data sets are stored on a volume with a data set prefix you define in the LPD cataloged procedure. Following are samples of these data sets:

- TCPUSR4.PRT1.QUEUE WRKLB2
- TCPUSR4.RALVM12.CFnnn BROWSED WRKLB2
- TCPUSR4.RALVM12.DFAnnnLU BROWSED TCPWRK
- TCPUSR4.RALVM12.JFnnn WRKLB2

The QUEUE... represents, in this sample PRT1's print queue file. It will contain the name of the JOB files that have not been completely processed yet.

The CF... represents the CONTROL FILE. Contains the control data/commands sent to LPD.

The DF... represents the DATA FILE. The actual data sent to be printed.

The JF... represents the JOB FILE. Contains names of the above files that have not been processed yet.

where nnn is the three digit job number.

Occasionally, depending on the precipitating incident and the time the connection was broken, the LPD server creates only a portion of one or more data sets. When partial temporary data sets are created, the server issues allocation or failure-to-erase messages. If you receive any of these messages, search for the partial data sets and either rename or delete them. After doing this, you might need to reissue the print request or requests.

The LPD trace and the system log at the time a connection is broken show the status of all print jobs (and the status of some data sets) and identify the owners of the print requests.
Steps for diagnosing LPR working with some options only
If the LPR command works with some options, but not with others, perform one or more of the following:

1. If some print requests do not work with certain LPR options, check the LPR trace for error messages.

2. If the LPR command from batch fails, but works under TSO, check for possible errors in the batch-job output and for error messages in the LPR trace.

For information about the LPR command, refer to the z/OS Communications Server: IP User’s Guide and Commands.

LPR client traces

This section provides information about activating LPR client traces. It also provides samples of trace output with explanations of selected messages.

Step for activating LPR client traces
You can activate LPR client traces by specifying the TRACE option in addition to the usual processing parameters on the LPR command.

For example, enter the following command to start the LPR client with trace on:

```bash
LPR filename (Printer p1 Host h1 TRACE
```

Step for creating client trace output
LPR trace output is sent to SYSOUT and can be displayed on the LPR client console. Figure 51 on page 416 is a sample of an LPR trace created by way of TSO with the following command:

- LPR soto.files(lpconfig) (p prt1 h 9.67.113.60 TRACE

________________________________________
EZB0915I Begin "LPR" to printer "prt1" at host "9.67.113.60"

EZB0920I Requesting TCP/IP service at 96155 18:52:53
EZB0921I Granted TCP/IP service at 96155 18:52:53
EZB0922I Resolving 9.67.113.60 at 96155 18:52:53
EZB0924I Host 9.67.113.60 name resolved to 9.67.113.60 at 96155 18:52:53
EZB0925I TCP/IP turned on.
EZB0926I Host "MVSA" Domain "TCP.RALEIGH.IBM.COM" TCPIP Service Machine TCP31S
EZB0927I Trying to open with local port 721 to foreign host address 9.67.113.60
EZB0928I Connection open from local port 721 to foreign host address 9.67.113.60
EZB0961I Control file name is cfA827MVSA
EZB0962I Data file name is dfA827MVSA Port Number=721. Remote IP Addr=9.7.113.60
EZB0916I Sending command 2 argument: prt1 Port Number=721. Remote IP Addr=9.67.113.60
EZB0917I Command successfully sent Port Number=721. Remote IP Addr=9.67.113.60
EZB1012I Receiving ACK Port Number=721. Remote IP Addr=9.6 7.113.60
EZB1013I ReceiveACK: TRUE for byte value 00 Port Number=721. Remote IP Addr=9.67.113.60
EZB0997I Byte size check starts at 96155 18:52:54
EZB0998I Byte size check ends at 96155 18:52:54
EZB0999I Send command starts at 96155 18:52:54 Port Number=721. Remote IP Addr=9.67.113.60
EZB0916I Sending command 3 argument:7434 dfA827MVSA Port Number=721. Remote IP Addr=9.67.113.60
EZB0917I Command successfully sent Port Number=721. Remote IP Addr=9.67.113.60
EZB1012I Receiving ACK Port Number=721. Remote IP Addr=9.67.113.60
EZB1013I ReceiveACK: TRUE for byte value 00 Port Number=721. Remote IP Addr=9.67.113.60
EZB1000I Send command ends at 96155 18:52:55 Port Number=721. Remote IP Addr=9.67.113.60
EZB1001I Send data starts at 96155 18:52:55 Port Number=721. Remote IP Addr=9.67.113.60
EZB1002I Send data ends at 96155 18:52:56 Port Number=721. Remote IP Addr=9.67.113.60
EZB1003I Send ACK starts at 96155 18:52:56 Port Number=721. Remote IP Addr=9.67.113.60
EZB1014I Sending ACK Port Number=721. Remote IP Addr=9.67.113.60
EZB1015I ACK successfully sent Port Number=721. Remote IP Addr=9.67.113.60
EZB1004I Send ACK ends at 96155 18:52:56 Port Number=721. Remote IP Addr=9.67.113.60
EZB1012I Receiving ACK Port Number=721. Remote IP Addr=9.6 7.113.60
EZB1013I ReceiveACK: TRUE for byte value 00 Port Number=721. Remote IP Addr=9.67.113.60
EZB1009I Data file sent. Port Number=721. Remote IP Addr=9.67.113.60
EZB1011I Queuing control line "HMVSA.TCP.RALEIGH.IBM.COM"
EZB1011I Queuing control line "PTCPUSR4"
EZB1011I Queuing control line "JTCPUSR4.SOTO.FILES(LPCONFIG)"
EZB1011I Queuing control line "CMVSA.TCP.RALEIGH.IBM.COM"
EZB1011I Queuing control line "LTCPUSR4"

Figure 51. Example of LPR trace output (Part 1 of 2)
Following are short descriptions of the numbered items in the trace:

1. Indicates the translation table used by the LPR client. In this print request, no translation tables were defined by the person submitting the request.

2. Indicates LPR port used to connect to the LPD server with the IP address 9.67.113.60. The LPR port range is from 721 through 731.

3. Indicates the LPR command sent to the LPD server identifying the name of the print queue where the output was sent. Refer to RFC1179 for details on commands and subcommands issued between LPR and LPD.

4. Indicates the command that provided the LPD print server with the byte size (7434) and name of the data file (dfA827MVSA) that was sent.
   - The character string dfA indicates that this was a data file.
   - The number 827 was the three-digit job number that was randomly generated by the LPR client or specified in the LPR command using the JNUM option.
   - MVSA was the name of the host from which the print request came.

5. Indicates the client is waiting for the LPD server to acknowledge the sending command in item 4. The message on the following line (TRUE (00)) indicates that the client received an acknowledgment. A FALSE message or any value other than zero terminates the LPR print request.

6. Indicates that the LPR client started and then stopped sending the data file.

7. Indicates that the LPR client notified the LPD server, by way of an acknowledgment, that the complete file was sent. The LPR client waits for the server to acknowledge receipt of the entire data file.

8. Indicates that the client received an acknowledgment from the server that the entire data field was received.

9. Confirms that the data file was sent to the LPD server.

10. Specifies one of the several control records sent by the LPR client. (The records are described in detail in RFC1179.) This control record is mandatory and represents the name of the data file created by the LPD server. The name is preceded by the filter specified on the LPR command. The letter f denotes the default filter.
Specifies the byte size (153) and the name of the control file (cfA827MVSA) that was sent.

Indicates that the LPD server received the command and expected the control file to be sent.

Indicates that the LPR client sent the control file and an acknowledgment that it finished sending the entire file. The last line in the block indicates that the client was waiting for an acknowledgment from the server.

TRUE (00) indicates that the client received an acknowledgment from the LPD server that the control file was received.

Confirms that the control file was sent to the LPD server. The job was then terminated.

Figure 52 is a sample LPR trace showing a print request in which the FILTER X option was specified on the LPR command. Since the LPD server does not support this type of filter, it rejects the print request. (For an example of an LPD trace that shows that this job was rejected, see Figure 57 on page 428.) The LPR trace does not show an error because it can send a print request to non-IBM LPDs that support other filters (for example, FILTER X). For detailed information about filters, refer to RFC1179 and to the z/OS Communications Server: IP Configuration Reference.

The trace was produced using the following command issued through TSO by user ID TCPUSR4:

LPR test (p TIANNA h 9.67.113.60 filter x TRACE

Following are short descriptions of the numbered items in the trace:

1. EZB0915I Begin "LPR" to printer "TIANNA" at host "9.67.113.60"
2. EZB1009I Data file sent. Port Number = 721. Remote IP Addr = 9.67.113.60
3. EZB1011I Queuing control line "HMVSA.TCP.RALEIGH.IBM.COM"
4. EZB1011I Queuing control line "UdfA947MVSA"
5. EZB09016I Sending command 2 argument: 122 cfA947MVSA Port Number = 721. Remote IP Addr = 9.67.113.60
6. EZB0917I Command successfully sent Port Number = 721. Remote IP Addr = 9.67.113.60
7. EZB1018I Control file sent Port Number = 721. Remote IP Addr = 9.67.113.60
Indicates that the print request was issued to a printer named TIANNA at IP address 9.67.113.60.

Indicates that the data file was sent. The error was not recognized until the LPD server tried to process the print job. (See Figure 57 on page 428.)

Indicates control commands sent to the LPD server. For details about these commands, refer to RFC1179.

Represents the name of the data file. The character string xdf indicates that the x filter was used.

Indicates that the control file was sent to the LPD server. The job was then terminated.

Figure 53 is a sample showing a print request using the following command lpr test (p njeSOTO host MVSA without the TRACE option. The output shows an error because the printer name was not entered entirely in capital letters.

1
EZB1006E Host MVSA did not accept printer name njeSOTO.
   Port Number = 721 Remote IP Addr = 9.67.113.60

2
EZB1049E Send printer command did not receive ACK. ACK message = .
   Port = 721. Remote IP Addr = 9.67.113.60

Figure 53. Example of LPR output with unknown printer

Following are short descriptions of the numbered items in the trace.

1 Indicates that a SERVICE statement for a printer named njeSOTO did not exist in the LPD server configuration file.

2 Indicates that the LPD server did not send a positive response to the LPR client. The job was then terminated.

Figure 54 on page 420 is a sample LPR trace output produced with the following command the JNUM option and variable, along with the LANDSCAPE and TRACE options:

lpr test (p TIANNA host 9.67.113.60 JNUM 111 LANDSCAPE TRACE

The trace output shows the scanning that occurred to identify the first available port.
Following are short descriptions of the numbered items in the trace:

1. Indicates that the LPR client inserted a landscape header, written in postscript, at the beginning of the data file.
2. Indicates that the LPR client was attempting to use the first available client port. The port range for the LPR client is 721 through 731. If no ports are available, an error message is displayed.
3. Indicates that a connection was opened using port 724.
4. Indicates that the value specified for JNUM (111) was used to build the control and data file names.
5. Indicates the name of the file containing the three-digit job number that was used with the file name sent to the print server.

Following is a clipping of the header that was inserted into the data file. For more information about header files, refer to z/OS Communications Server: SNA Customization.

Figure 54. Example of LPR trace with JNUM, LANDSCAPE, and TRACE options

Following is a clipping of the header that was inserted into the data file. For more information about header files, refer to z/OS Communications Server: SNA Customization.
In this sample, the server was not running, so the connection was not established. For detailed information about using and creating your own translate tables, refer to "z/OS Communications Server: SNA Customization">

EZB0915I Begin "LPR" to printer "TIANNA" at host "MVSA"
1 EZB0957I Loaded translation table from "TCPUSR4.GXS.TCPXLBIN".
2 EZB0920I Requesting TCP/IP service at 96155 20:04:14
3 EZB0921I Granted TCP/IP service at 96155 20:04:15
4 EZB0922I Resolving MVSA at 96155 20:04:15
5 EZB0924I Host MVSA name resolved to 9.67.113.60 at 96155 20:04:17
6 EZB0925I TCP/IP turned on.
7 EZB0926I Host "MVSA" Domain "TCP.RALEIGH.IBM.COM" TCPIP Service Machine TCP31S
8 EZB0927I Trying to open with local port 721 to foreign host address 9.67.113.60
9 EZB1051E Failed to Open connection to Port Number = 515. Return Code = -1. Error Number = 61. Port Number = 721.
   Remote IP Addr = 9.67.113.60

Figure 55. Example of LPR trace with XLATE option

Following are short descriptions of the numbered items in the trace:

1. Indicates the name of the translation table used by the LPR client. To avoid problems such as errors and data corruption, be sure that the LPD server is using the equivalent code pages.

2. Indicates the time the LPR client started trying to resolve the specified host name. The LPR client checks the name server table, the site, and address information files to resolve the host name.

3. Indicates the amount of time the LPR client took to resolve the specified host name. To reduce the amount of time, use the host IP address instead of the host name.

4. Indicates that the connection was not established. (In this sample, the LPD server was not running.) For a list of error numbers and their definitions, refer to "z/OS Communications Server: IP and SNA Codes".

LPD server traces

This section includes information on activating LPD server traces. It also provides samples of LPD trace output with explanations of selected messages.

Step for activating server traces

You can activate the tracing facilities within the LPD server in any of the following ways:

- Include the TRACE parameter in the LPSERVE PROC statement in the LPD server cataloged procedure. Be sure that a slash (/) precedes the first parameter and that each parameter is separated by a blank. For example:
  //LPSERVE PROC MODULE='LPD',PARMS='/TRACE'

- Enter the command SMSG procname, where procname is the name of the procedure used to start the LPD server.

- Specify the DEBUG statement in the LPD configuration file, LPDDATA.
Step for creating server trace output

LPD server traces go to the SYSPRINT data set. You can also define a DD card in the LPD cataloged procedure to write output to another data set. This section contains some samples of LPD server trace output.

Figure 56 is a sample of an LPD trace invoked by specifying the DEBUG option in the LPD configuration file, LPDDATA.

```
EZB0832I
EZB0621I LPD starting with port 515
EZB0679I Allocated ObeyBlock at 00005870
EZB0679I Allocated ObeyBlock at 00005860
EZB0679I Allocated ObeyBlock at 00005850
EZB0628I Allocated PrinterBlock at 000058C0
EZB0629I  prtl added.
EZB0641I Service prtl defined with address
EZB0628I Allocated PrinterBlock at 00005630
EZB0629I  PRT1 added.
EZB0641I Service PRT1 defined with address
EZB0628I Allocated PrinterBlock at 000053A0
EZB0629I  TIANNA added.
EZB0641I Service TIANNA defined with address
EZB0628I Allocated PrinterBlock at 00005110
EZB0629I  PRT2 added.
EZB0641I Service PRT2 defined with address
EZB0628I Allocated PrinterBlock at 00005040
EZB0629I  njesoto added.
EZB0641I Service njesoto defined with address
EZB0628I Allocated PrinterBlock at 000041A0
EZB0629I  rda added.
EZB0686I Host "9.37.33.159" resolved to 9.37.33.159. Printer name is "lpt1".
EZB0641I Service rda defined with address
EZB0628I Allocated PrinterBlock at 00001820
EZB0629I  POST added.
```

Figure 56. Example of LPD trace specified with the DEBUG option (Part 1 of 5)
Figure 56. Example of LPD trace specified with the DEBUG option (Part 2 of 5)
Figure 56. Example of LPD trace specified with the DEBUG option (Part 3 of 5)
Chapter 13. Diagnosing line print requester and daemon (LPR and LPD) problems

Figure 56. Example of LPD trace specified with the DEBUG option (Part 4 of 5)
Following are short descriptions of the numbered items in the trace:

1. Indicates that a control block was allocated for each service defined in the LPD configuration file. TIANNA is the name of one of the local printers.

2. Indicates that the remote printer, LPT2, was defined in a SERVICE statement with the name POST. LPT2 has the IP address 9.67.105.55.

3. Indicates that the LPD server listened on port 515 and that port 515 was opened.

4. Indicates that the LPD server waited for work.

Figure 56. Example of LPD trace specified with the DEBUG option (Part 5 of 5)
Indicates that a connection was opened for an incoming LPR client and that the LPD server was receiving a command from that client.

Indicates that a subcommand was received from an LPR client. The subcommand indicates LPD was receiving a data file named dfA827MVSA, containing 7434 bytes of data. For details on commands and subcommands, refer to RFC117.

Indicates that the LPD server had a passive open connection on the restricted LPD port, 515.

Indicates that the LPD server was receiving a control file named cfA827MVSA, containing 153 bytes of data.

Note: Data files use the naming convention of dfx. Control files use the naming convention cfx.

Indicates the control blocks that were allocated and released as files were received and processed. Control blocks are used primarily by IBM support for debugging purposes, in coordination with dumps.

Indicates that all data files for a particular job were received.

Note: Job number 827 is a three-digit job number generated by the LPR client.

Indicates that job 827 was added to this print queue. The LPD server maintains a work queue of jobs.

Indicates that job 827 was scheduled to be spooled to the output queue.

Indicates that the LPD server was processing print jobs from the work queue, and started sending print data to the JES output queue. The message JOBstartPRINTING does not mean that the file is physically printing.

Indicates that data was being sent for output. Depending on the size of the file, you might see this status many times for a single job.

Indicates checking for the end of the file as it is being processed. The number of IsAtEof entries depends on the data and size of the file.

Indicates that all data was processed and placed in the output queue.

Indicates that job 827 was completely processed by the LPD server and removed from the print queue, prt1, on host MVSA. Temporary data sets and control blocks for this job were also erased or released.

Indicates that the LPD server completed the jobs in that queue and scans the work queue again.

Indicates that the LPD server was waiting for more work to do.

Indicates that the LPR-to-LPD connection was closed normally.

Indicates that someone stopped the LPD server normally.

Figure 57 on page 428 is a sample of LPD trace output showing that job 947 failed to print because the client passed a filter that was not supported by the LPD server. In cases such as these, you can lose printouts. In this case, the LPD trace showed why, but the LPR trace did not show an error. (See Figure 52 on page 418 for the corresponding LPR trace output.)
Figure 57. Example of an LPD server trace of a failing job (Part 1 of 2)
Figure 57. Example of an LPD server trace of a failing job (Part 2 of 2)

Following are short descriptions of the numbered items in the trace:

1. Indicates that the LPD server received a command indicating the byte size and name of a data file sent by an LPR client.
2. Indicates that the LPD server received a command indicating the byte size and name of a control file sent by an LPR client.
3. Indicates that print job 947 was received, placed in the print queue named TIANNA on host MVSA, and was scheduled to be processed.
4. Indicates that the LPD server did not support filter x and discarded the print job.
5. Indicates that the job was finished. The flag JOBfinishPRINTING indicates the job is to be removed from the work queue and purged.
6. Indicates that the job was removed from the work queue and that the control blocks were released.
7. Indicates that the job was purged.

Figure 58 on page 431 is a sample of an LPD trace output generated by specifying the DEBUG statement in the LPD configuration file (LPDDATA). This sample shows that an LPR client issued a request, through an LPD server, to a printer defined as a remote server. (The LPD server acted as an LPR client by sending the request to a remote server.) Since the remote server was not running, the print job was purged.

Initially, the LPR client was unaware that the server was not running because the LPD server correctly acknowledged receipt of the data files and control files. Furthermore, the LPR trace did not indicate any problems. However, if you specify the option FAILEDJOB MAIL on the SERVICE statement for the remote printer,
notification is sent to the user ID of the LPR client. For notification to be sent, Simple Mail Transfer Protocol (SMTP) must be running.

**Note:** The FAILEDJOB DISCARD option is the default.

The command `LPR lpd.config (p SOTO h MVS7` was used to generate the trace output. SOTO is the name of the printer specified on the SERVICE statement, and MVS7 is the host on which the LPD server is running.
Figure 58. Example of an LPD server trace for a remote print request (Part 1 of 3)
Figure 58. Example of an LPD server trace for a remote print request (Part 2 of 3)
Following are short descriptions of the numbered items in the trace:

1. Indicates the date and time the LPD server was activated. This information can be compared to the date and time on an LPR trace to assure that both traces were generated for the same incident.
Indicates the IP address of the host. If the name of the host was specified instead of the IP address, this message would indicate if the IP address of the host was resolved.

Indicates that the name SOTO was defined on the SERVICE statement for the remote printer, lpt1, which had the address 9.37.34.39.

Indicates the byte size and the name of the data file sent from the LPR client on host MVS7.

Indicates the byte size and name of control file sent from the LPR client on host MVS7.

Indicates that the connection between the LPR client and the LPD server was closing, after the server received the data and control files.

Indicates that the print job was received, placed in the LPD print queue, represented by SOTO, and scheduled to be sent to its destination.

Indicates that the LPD server started to send print job 502 to the remote server.

Tip: If the printer was local, rather than remote, the message would have read 502 JOBstartPRINTING.

Indicates that the LPD server, acting as a client, was opening a connection to the remote printer using local port 721.

Indicates that the LPD server removed the job from its work queue.

Indicates that the connection to the remote server timed out.

Indicates that the remote server did not respond to the request to open.

Indicates that the FAILEDJOB MAIL option was defined under the SERVICE statement and that SMTP was running. The text in these messages was sent to the user ID of the LPR client.

Indicates that the print job was completely purged.

Describes additional activity between the LPD server and other clients.
Chapter 14. Diagnosing File Transfer Protocol (FTP) problems

This topic describes how to diagnose problems with the z/OS Communications Server FTP server and FTP client. If, after reading this topic, you are unable to solve your problem and you need to call the IBM Software Support Center, see one or both of the following sections for the documentation you need to provide: "Documenting server problems" on page 464 and "Documenting FTP client problems" on page 486.

This topic assumes your security product is RACF. However, you can use any SAF-compliant security product.

FTP server

This section contains the following topics:
- "Structural overview"
- "Definitions and setup" on page 436
- "Error exit codes" on page 436
- "Name considerations for z/OS UNIX FTP" on page 436
- "Translation and data conversion support" on page 437
- "DB2 query support" on page 439
- "JES support" on page 441
- "Common z/OS UNIX FTP problems" on page 443
- "Diagnosing FTP server problems with traces" on page 456
- "Documenting server problems" on page 464

Structural overview

The z/OS model for the FTP server includes a daemon process and a server process. The daemon process starts when you start your cataloged procedure (for example, START FTPD) and it listens for connection requests on a specific port. The port is the well-known port 21 unless otherwise specified. For methods of choosing a different port number, see the information about configuring ETC.SERVICES and configuring the FTPD cataloged procedure in the z/OS Communications Server: IP Configuration Guide. When the daemon accepts an incoming connection, it creates a new process (server’s address space) for the FTP server, which handles the connection for the rest of the FTP login session. Each login session has its own server process.

The server process inherits the accepted connection from the daemon process. This connection is called the control connection. The server receives commands from the client and sends replies to the client using the control connection. The control connection port is the same as the daemon’s listening port.

The client and server use a different connection for transferring data; this connection is called the data connection. By default, the data port is one less than the control connection port. For example, if the control connection port is 21, the data port is 20. An FTP client can override the default data port by directing the server to run in passive mode. In passive mode, the server uses an ephemeral port for the data port. Passive mode is requested by firewall-friendly clients and by clients initiating three-way data transfers.
Definitions and setup

This section describes the definitions and setup for the FTP server.

Start procedure

The sample start procedure for the FTP server is EZAFTPAP (alias FTPD) in the SEZAINST data set. Changes might be necessary to customize the start procedure for your MVS host system.

Keep the following in mind for the FTP server start procedure:

- The library containing FTPD and FTPDNS must be APF authorized and must be either in the MVS link list or included on the STEPLIB DD statement.
- The C run-time libraries are needed for FTPD and FTPDNS. They must be APF authorized. If the C run-time library is not in the MVS link list, it must be included on the STEPLIB DD statement.
- If the FTP server is used for SQL queries, the DB2® DSNLOAD library must be APF authorized and must be either in the MVS link list or included on the STEPLIB DD statement.
- Several start options are available for the FTP server. If specified in the start procedure, these values override the default values for the FTP server and any values specified in the FTP.DATA data set.

For more information about the FTP server start procedure, refer to the z/OS Communications Server: IP Configuration Reference.

FTP.DATA data set

The FTP.DATA data set is an optional data set that allows the FTP server configuration parameters to be customized. Refer to the z/OS Communications Server: IP Configuration Reference for more information about the FTP.DATA data set.

TCPIP.DATA data set

The TCPIP.DATA data set provides the following information to the FTP server:

- High-level qualifier to be used for configuration data sets
- Whether messages are to be written in uppercase or mixed-case
- Which DBCS translation tables are to be used

For more information about the TCPIP.DATA data set, refer to the z/OS Communications Server: IP Configuration Reference.

Error exit codes

z/OS UNIX FTP uses the following error exit codes:

12 Daemon initialization failed; unable to accept an incoming connection. An EZY message identifying the specific problem is sent to syslogd.

24 The client session’s initialization terminated because the FTP server load module cannot be loaded or executed. Message EZYFT53E is sent to syslogd.

28 Daemon initialization was terminated because the IBM TCP/IP is not enabled in the IFAPRDxx parmlib member. Message EZYFT54E is sent to syslogd and the operator console.

Name considerations for z/OS UNIX FTP

This section explains the MVS and z/OS UNIX file system naming conventions.
MVS naming conventions

Restrictions: MVS data set names used with all FTP commands sent to the z/OS UNIX FTP server must meet MVS data set naming conventions as follows:

- Data set names cannot be longer than 44 characters.

  If the path name parameter sent with an FTP command is not enclosed in single quotation marks, the path name is appended to the current working directory to create the data set name. The combination of the current working directory and the path name cannot be longer than 44 characters. Issue the PWD command to display the current working directory.

- Each qualifier in a data set name, or each member name for a partitioned data set, must conform to the following:
  - No longer than 8 characters.
  - Begin with a letter or the special characters $, @, or #.
  - Contain only numbers, letters, or the special characters $, @, #, -, or {.

- Generation data group data set names must be in the format gdg_name(generation_level). The generation_level is either 0, +nn, or −nn, where nn is the generation number. For example, the GDG data set MYGDG could be specified as MYGDG(0) for the current generation level, MYGDG(-1) for the next to the latest generation level, or MYGDG(+1) for the new generation level.

z/OS UNIX file system naming conventions

Guidelines: The following list describes some naming conventions you should know about when using z/OS UNIX file system files with the z/OS UNIX FTP server:

- The z/OS UNIX file system name is case-sensitive.
- If a name begins with a single quotation mark, specify QUOTESOVERRIDE FALSE in FTP.DATA, or use the SITE NOQUOTESOVERRIDE command.
- Names can contain imbedded blanks for special characters.
  Tip: Some FTP clients might truncate trailing blanks.
- The LIST and NLST subcommands, including all client subcommands that invoke the NLST subcommand, such as MGET or MDELETE, require special handling for certain special characters. For more information, refer to z/OS Communications Server: IP User’s Guide and Commands.
- The START and SITE parameters have additional restrictions on the path name used with SBDATACONN. Refer to z/OS Communications Server: IP Configuration Reference and z/OS Communications Server: IP User’s Guide and Commands.
- When specifying a z/OS UNIX FTP subcommand with a file name containing special characters, some FTP clients might:
  - Truncate trailing blanks
  - Compress multiple internal blanks
  - Interpret special characters to have special meanings

Unique specification of the file name such as enclosing in double or single quotation mark, or escaping special characters, might be necessary to make the client send the file name to the server correctly. Refer to your client documentation to see if this is necessary.

Translation and data conversion support

This section describes translation and data conversion support for the FTP server.
Double-byte character set (DBCS) support
If you enter quote type b <n> at the client and if the DBCS translate table has not been loaded, the following reply is displayed:
504-Type not Supported. Translation table not loaded.

Do one or both of the following:
• Check the LOADDBCSTABLES statement in the TCPIP.DATA configuration file.
  If the statement wraps to the next line, parameters on the continued line are ignored. If all the parameters for the LOADDBCSTABLES statement do not fit on one line, use multiple LOADDBCSTABLES statements.
• Check the precedence order for TCPIP.DATA to ensure that the file being used contains the LOADDBCSTABLES statement or statements. Be aware that the location of TCPIP.DATA statements can be influenced in multiple ways, for example, by a GlobalTCPIPData specification or the RESOLVER_CONFIG environment variable. Refer to the z/OS Communications Server: IP Configuration Reference for the TCPIP.DATA search order.

Single-byte character (SBCS) support
Data conversion occurs for single-byte data on the data connection when ENCODING=SBCS is in effect and the data type is ASCII. For more information, refer to the FTP.DATA statement ENCODING in the z/OS Communications Server: IP Configuration Reference and the SITE ENCODING command in the z/OS Communications Server: IP User's Guide and Commands.

If you choose SBDATACONN as a statement in the FTP.DATA file or with the SITE SBDATACONN command, the FTP server builds a translation table using the code pages specified by SBDATACONN. If you receive the following reply to the SITE command, ask for a trace of the server with the UTL option to determine which characters cannot be translated.
200 Some characters cannot be translated between codepage_1 and codepage_2.

If none of the untranslatable characters appear in the data, the data transfers are not affected. If, however, one of the untranslatable characters does appear, the data transfer fails and the client receives the following reply:
557 Data contains codepoints that cannot be translated.

You can avoid the failure if you specify a substitution character to replace non-translatable characters. For details on how to ask for character substitution, refer to SBSUB and SBSUBCHAR as FTP.DATA statements in the z/OS Communications Server: IP Configuration Reference and as parameters on the SITE command in z/OS Communications Server: IP User's Guide and Commands. If substitution occurs during the transfer, the client receives the following reply:
250 One or more characters were substituted during the transfer.

When substitution occurs at the destination of a data transfer, a subsequent transfer of the resulting data does not produce an exact copy of the original. For example, if you put a file to the server and one or more characters are substituted, the untranslatable characters are overlaid in the server copy with the substitution character. You cannot restore the original file by getting it from the server.

Multibyte character set (MBCS) support
Data conversion occurs for multibyte data on the data connection when ENCODING=MBCS is in effect and the data type is ASCII. For more information, refer to the FTP.DATA statement ENCODING in the z/OS Communications Server: IP Configuration Reference and the SITE ENCODING command in the z/OS Communications Server: IP User's Guide and Commands.
If you choose ENCODING=MBCS, you must specify MBDATACONN with a statement in the FTP.DATA file or with the SITE MBDATACONN command to name the code pages for the multibyte data transfer. If you attempt an ASCII data transfer with ENCODING=MBCS and no MBDATACONN specified, the client receives the following reply:

504  Multibyte encoding set but code pages are not defined.

If the multibyte data that you transfer has codepoints that cannot be translated, the transfer fails and the client receives the following reply:

557  Data contains codepoints that cannot be translated.

You can determine which bytes of the data cannot be translated by repeating the transfer with the DUMP 42 extended trace option active at the server.

**DB2 query support**

This section describes how to use FTP server DB2 query support and how to diagnose SQL problems.

**Steps for using FTP server SQL support**

**Before you begin:** Before you can use the FTP server to submit queries to the DB2 subsystem, complete the following steps:

1. Start the DB2 subsystem.

2. BIND the DBRM called EZAFTPMQ. This must be done whenever the part EZAFTPMQ.CSQLMVS has been recompiled.
   The DBRM must be bound into the plan named EZAFTPMQ, unless the keyword DB2PLAN was used in your FTP.DATA file to specify a different plan name.
   If you are running multiple instances of the z/OS UNIX FTP server at different maintenance levels, you must use DB2PLAN in FTP.DATA for each server and specify unique plan names.

3. Grant execute privilege to the public for the plan created in the previous step.

To submit a query to DB2 through the FTP server, issue the following commands as necessary:

- **SITE FILETYPE=SQL**
- **SITE DB2=db2name** where db2name is the name of a DB2 subsystem at the host
- **RETR fname1 fname2** where fname1 is a file at the host that contains a SQL SELECT statement

**Symptoms of SQL problems**

Table 26 on page 440 and Table 27 on page 441 show some symptoms and possible causes of SQL problems. Table 26 on page 440 shows problems that generate a reply beginning with 55x.
Table 26. SQL problems generating 55x replies (FTP Server)

<table>
<thead>
<tr>
<th>Reply</th>
<th>Output file</th>
<th>Possible causes</th>
</tr>
</thead>
</table>
| Reply 551: Transfer aborted: SQL PREPARE/DESCRIBE failure | The output file contains the SQL code and error message returned by the DB2 subsystem. | • A syntax error in the SQL statement in the host file.  
• The time stamp in the load module is different from the BIND time stamp built from the DBRM (SQL code = -818). This occurs if a BIND was not done for the EZAFTPMQ DBRM that corresponds to the current load module, or if the server is not configured to use the correct DB2 plan name. If this is the problem, every SQL query submitted through the FTP server fails. |
| Reply 551: Transfer aborted: unsupported SQL statement | No output is sent from the host. | The file type is SQL, but the host file being retrieved does not contain an SQL SELECT statement. |
| Reply 551: Transfer aborted: attempt to connect to db2name failed (code) | No output is sent from the host. | • The site db2name specifies a nonexistent DB2 subsystem.  
• The DB2 subsystem has not been started. |
| Reply 551: Transfer aborted: SQL not available. Attempt to open plan <plannam> failed (DB2_reason_code). | No output is sent from the host. | • BIND was not done for the specified plan.  
• BIND was done for plan name other than EZAFTPMQ, but FTP.DATA does not contain a DB2PLAN statement to specify this plannam.  
• User does not have execute privilege for the DB2 plan being used by the FTP server. |
| Reply 550: SQL query not available. Cannot load CAF routines. | No output is sent from the host. | The DSNLOAD library is not in the link list or the FTP server STEPLIB. |

Note: For more information about the messages, refer to z/OS Communications Server: IP and SNA Codes.
Table 27 shows other SQL problems.

Table 27. Other SQL problems (FTP Server)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Possible causes</th>
</tr>
</thead>
</table>
| Output file contains only the SQL SELECT statement. | - The file type is SEQ, rather than SQL. If the file type is SEQ, a retrieve is done, but the host file is just sent back to the client. The query is not submitted to the DB2 subsystem.  
- The SELECT is for a VIEW for which the user ID does not have DB2 select privilege. The DB2 subsystem returns an empty table. |
| Client closes the connection because server is not responding. | The processing time needed by DB2 and FTP or both for the SQL query has exceeded the client's time limit for send or receive.  
An FTP server trace with the options FSC and SQL indicates the amount of SQL activity through FTP and the approximate time when each query was processed. |

JES support

This section describes the procedures to follow when JES output is not found and when remote job submission functions fail.

JES output not found (zero spool files)

In some cases, the server is in JESINTERFACELEVEL=1 and FILETYPE=JES, and a job has been submitted, but the output of the job cannot be found. You get zero spool files from a DIR command.

Use the following checklist to investigate:

1. Is the job name correct? The job name must consist of the user ID followed by a single character.
2. Was the job output spooled to the hold queue? The server is only be able to retrieve job output that is in the hold queue. For JES/3, output must be assigned to an output queue held for external writer.
3. Did you set SBSENDEOL to a value other than CRLF for your original outbound file transfer? If so, it is not be possible to restart the file transfer. You should send the entire file to the server again.

Example

If JESINTERFACELEVEL=2, ensure the JESJOBNAME, JESSTATUS and JESOWNER filters are set correctly with the STAT command.

If the server is in JESINTERFACELEVEL=2 and FILETYPE=JES, and a job has been submitted, but the output of the job cannot be found (that is, you get zero spool files from a DIR command), check the 125 reply message to verify that the JESOWNER, JESJOBNAME, and JESSTATUS filters are set to values that apply to your job.

Example
If the JESJOBNAME=USER1* and the job submitted was USER2A, use the SITE command to set the JES filter to the appropriate value to find the job requested. If the SITE command does not allow the end user to change the values of the three JES filters, refer to the z/OS Communications Server: IP User's Guide and Commands to determine if the proper Security Access Facility resources allow changing of the JES filters for the user.

Remote job submission functions fail
For problems with remote job submission, run the FTP JES trace to check for the following:
- Cannot allocate internal storage
- JES is not communicating
- JES unable to find output for the specified job ID
- Unable to acquire JES access
- Unknown return code from GET JES spool request
- JES unable to provide spool data set name now
- JES unable to get a job ID for a PUT or GET request
- JES PUT or GET aborted, job not found
- JES PUT or GET aborted, internal error
- JES PUT or GET aborted, timeout exceeded
- JES internal reader allocation failed
- JES user exit error

To trace the FTP JES activity, use the DEBUG=(JES) or DUMP=(JES) options of FTP syslog tracing. See “Diagnosing FTP server problems with traces” on page 456 for information about activating FTP syslog tracing.

FTP connection stops during FILETYPE=JES processing
This problem occurs due to extended periods of inactivity on the control or data connection while the FTP client is waiting for the FTP server to complete the job. Significant delays are possible during FILETYPE=JES processing due to conditions such as heavy system utilization and dispatching priorities.

You can avoid timeouts by using keepalive packets on the control and data connection. You can activate keepalive packets by:
- Coding the INTERVAL parameter on the TCPCONFIG statement in the TCP/IP profile (PROFILE.TCPIP). See z/OS Communications Server: IP Configuration Reference for more information on the TCPCONFIG statement.
- Coding the FTPKEEPALIVE statement in FTP.DATA. This activates keep alive packets for the control connection only, and overrides the TCPCONFIG INTERVAL statement. See z/OS Communications Server: IP Configuration Reference for more information on the FTPKEEPALIVE statement.
- Configuring the DATAKEEPALIVE option to activate keepalive packets for the data connection only. This overrides the TCPCONFIG INTERVAL statement in the stack. You can configure DATAKEEPALIVE by coding a statement in FTP.DATA, by using the site subcommand from the z/OS FTP client, or by sending a SITE command with the DATAKEEPALIVE parameter to the FTP server. See z/OS Communications Server: IP Configuration Reference for more information on the DATAKEEPALIVE statement.

Logging FTP server activity
The z/OS FTP server provides a way to log standardized information for the following types of activity:
- Connections from the client end user to the server
• Authentication of the client/server session (for example, through the use of Transport Layer Security)
• Access to the FTP server through User ID/password verification
• Allocation of MVS data sets and z/OS UNIX file system files
• Deallocation of MVS data sets and z/OS UNIX file system files
• Data transfers
• JES job submissions
• SQL queries
• Abnormal end (ABEND) conditions
• Confidence levels assigned to file transfers when CHKCONFIDENCE TRUE has been coded in FTP.DATA

Set the following server’s FTP.DATA statements to enable logging:

```
FTPLOGGING
ANONYMOUSFTPLOGGING
```

For more information about these statements, refer to z/OS Communications Server: IP Configuration Reference.

Until the client sends the USER command to the server, the server cannot know whether this is an anonymous login. Therefore, up to the point the server processes the USER command, the FTPLOGGING statement and ANONYMOUSFTPLOGGING statement produce identical results.

This information is recorded in the SYSLOGD file. The data has an identification field that allows correlation of all entries for a specific login session.

For more information about configuring the SYSLOGD file, refer to z/OS Communications Server: IP Configuration Guide.

Refer to the z/OS Communications Server: IP Configuration Reference for the server’s FTP.DATA configuration.

**Common z/OS UNIX FTP problems**

This section describes some common z/OS UNIX FTP problems.

**FTP daemon initialization problems**

You might encounter the following problems when the FTP daemon is initialized.

No "Initialization Complete" message: If the EZY2702I Server-FTP Initialization completed at ... message does not appear on the system console within a few minutes after starting the FTP daemon, verify that the daemon background job is still running. For example, if you started FTP with a procedure called FTPD, you can use the D A,L command to see if the job FTPD1 is active.

If the background daemon job is running (for example, FTPD1), verify that TCP/IP is running. If it is not, start TCP/IP. The FTP initialization completes when TCP/IP starts.

If the background daemon job is not running, check the system console for nonzero exit codes from the background job. Look for messages in message or trace output from syslogd for an EZY error message from FTP. The following are possible exit codes and the appropriate responses:
FTP is unable to use the port specified for the control connection. Look in the syslogd messages for the specific reason. Possible errors include the following:

- **EZYFT13E bind error...Operation not permitted**
  
  Ensure that FTP has BPX.DAEMON authority.

- **EZYFT13E bind error...Address already in use**
  
  Ensure that FTP is trying to use the correct port. The FTP server trace with the INT option indicates the port the daemon expects to use. If this is the correct port, you can use the TSO NETSTAT CONN command to determine the job that is currently using that port.

- **EZYFT13E bind error...Permission denied**
  
  Ensure that the port you want FTP to use has been reserved for the FTP background job name. For example, if your start procedure is called FTPD and you want FTP to use port 21, the PORT statement in your `hlq.PROFILE.TCPIP` data set must specify `21 TCP FTPD1`.

This FTP daemon is not available because the IBM TCP/IP is not enabled.

**Incorrect configuration values:** If you experience incorrect configuration values, check the following:

- Look in the syslogd output for message EZY2640I to verify that configuration values are coming from your intended FTP.DATA file. Verify that no errors were encountered reading this file.

- Determine whether your FTP.DATA file has sequence numbers. If it does, any statement with an optional parameter omitted picks up the sequence number as the parameter value.

  For example, the BLKSIZE statement has an optional parameter size. If you specify the size, the sequence number is ignored. If you do not specify the size, the system assumes the sequence number is the size, causing an error.

**FTP daemon not listening on expected port:** If the daemon is not listening on the expected port, verify that the correct port number is specified. Following is the preference order for a port number:

1. PORT start parameter
2. `/etc/services`
3. `hlq.ETC SERVICES`
4. A default port number of 21

**AUTOLOG does not start the FTP daemon:** If your start procedure name contains fewer than eight characters, ensure that the AUTOLOG and PORT statements in the `hlq.PROFILE.TCPIP` data set specify the FTP background job name. For example, if your start procedure is called FTPD, your `hlq.PROFILE.TCPIP` data set should specify FTPD1, as shown in the following examples:

```
AUTOLOG
  FTPD JOBNAME FTPD1
ENDAUTOLOG
```

```
PORT
  20 TCP OMVS NOAUTOLOG  ;FTP data port
  21 TCP FTPD1     ;FTP control port
```
User exit routine is not invoked
If the user exit routine is not invoked, check the FTP trace in syslogd to see if the exit routine was loaded. FTCHKIP is loaded once by the FTP daemon during initialization. The remaining user exits (FTCHKPWD, FTCHKCMD, FTCHKJES, FTPOSTPR, and FTPSMFEX) are loaded in the FTP server address space for each client session.

For example, check for one of the following:
main: ret code from fndmembr() for FTCHKIP is: 4
main: user exit FTCHKIP not found. Bypassing fetch().

or
main: ret code from fndmembr() for FTCHKCMD is: 0
main: chkcmdexit successfully loaded

If you have user-written exit routines and the FTP server is not able to find them, ensure that the user-written exit routines exist in an APF-authorized partitioned data set which is in the search order.

FTP Messages and FTP trace entries
If messages and trace entries do not appear in the syslog output file, do one or more of the following:
• Ensure that syslogd is configured for daemon entries. The file /etc/syslog.conf must have an entry for daemon.info to get FTP messages or an entry for daemon.debug to get FTP messages and trace entries.
• Ensure that the files specified for daemon entries exist at the time that syslogd started. If not, you need to create the files and recycle syslogd.
• Ensure that the files specified for daemon entries have appropriate permission bits (for example, 666).
• Ensure that syslogd is active.

If messages and trace entries display on the system console, it means that syslogd cannot write to the files specified for daemon entries and that /dev/console is defined. Check that syslogd is configured correctly and that the files specified for daemon entries have appropriate permission bits (for example, 666).

If you consider the volume of EZYFT47I messages logged by the server during initialization to be excessive, you can suppress these messages by adding a SUPPRESSIGNOREWARNINGS statement to the server’s FTP.DATA. However, if you use this statement, the FTP server does not warn you when it ignores statements coded in FTP.DATA.

Guideline: Add SUPPRESSIGNOREWARNINGS to FTP.DATA only after you have verified all statements in FTP.DATA are correct.

FTP server abends
If the FTP server abends, check the following:
• S683 or U4088 abend validating user ID or password.
  – Ensure that the sticky bit has been turned on for the files /usr/sbin/ftpd and /usr/sbin/ftpdns.
  – Ensure that the FTPD and FTPDNS modules reside in an APF authorized partitioned data set, which is specified in the MVS linklist.
  – Ensure that all programs loaded into the FTP address space are APF authorized and are marked as controlled. This means that any FTP user exits,
the SQL load library, and the loaded run-time library need to be marked as controlled, using the RACF RDEFINE command. For more information, refer to

**z/OS UNIX System Services Planning** or refer to the RACF publications.

### FTP session problems

The following sections describe some common FTP session problems.

**Connection terminated by the server after user enters user ID:** The system console might display one of the following nonzero exit codes from the FTP server address space:

- **0012** This exit code indicates a socket error. See the syslogd messages for the specific error.
- **0024** This exit code indicates that the system was unable to load the server load module /usr/sbin/ftpdns. Ensure that the symbolic link or links for ftpdns are correct, that ftpdns exists in the z/OS UNIX file system and that the sticky bit is on, and that FTPDNS exists in the search order.

If your system is not configured to display exit codes, check the syslogd output for an FTP error message.

**Connection terminated by the server after user enters password:** If the server terminates a connection after the user enters a password, ensure that the FTP load modules (FTPD and FTPDNS) reside in the APF authorized data set. Also, check that all programs accessed by the FTP address space are APF authorized and marked as “controlled.” Additional symptoms include the following:

- The FTP daemon is running, but the FTP server address space abends.
- The FTP server trace is active with the ACC option, and the last FTP trace entry reads:
  
  RA0\nnn pass: termid is ...

**Connection terminated by the server after user enters any subcommand:** If the server terminates a connection after the user enters a subcommand, either one or both of the following events might occur:

- FTP server address space shows an exit code of 0000.
- Last FTP server trace entry for the client session is RX\nnnn Server thread terminates rc = -2. The preceding entries indicate a “select” error due to a bad file descriptor.

These events indicate that the server inactive time limit has probably expired with no activity from the client. If this happens frequently, check the inactive time set for the server. If necessary, increase it, and recycle the FTP daemon.

**Password validation fails; session continues:** If password validation fails and the session continues, you receive the following reply:

530 PASS command failed

Additional replies might be generated if ACCESSERRORMSGS TRUE is coded in FTP.DATA.

If you receive this reply, do one or more of the following:

- Ensure all libraries, possibly indicated by ICH420I message, used by FTP are controlled and APF authorized.
- Ensure FTP is authorized if you are using BPX.DAEMON.
Ensure that the FTP daemon has been started from a user ID running with superuser authority if the daemon has been started from the z/OS UNIX shell.

Ensure that the login user ID has an OMVS segment defined, or that a default OMVS segment is established.

Obtain additional information about the error by enabling tracing with the ACC option.

**Anonymous login fails:** If an anonymous login fails, use the following checklist to investigate:

__1. Ensure that you have specified ANONYMOUS as a start parameter or in FTP.DATA.__

__2. Check the setting of the ANONYMOUSLEVEL variable in FTP.DATA. If ANONYMOUSLEVEL is not explicitly set in FTP.DATA, its value is equal to one.__

__3. If you have activated mixed-case passwords in RACF or in another SAF compliant security product, verify the following:

   • The anonymous password in FTP.DATA is coded in the correct case
   • The anonymous password passed to the FTP daemon by the FTPD start procedure is coded in the correct case
   • The anonymous password specified by the MVS operator to override the parameters specified in the FTPD start procedure was coded in the correct case.

**Rule:** Enclose the FTP parameters in single quotes when overriding the parameters specified in the FTPD start procedure while mixed-case passwords are enabled.

If ANONYMOUS is set in FTP.DATA, and the STARTDIRECTORY is in the z/OS UNIX file system, and ANONYMOUSLEVEL is two or three, verify that the required executable files are installed in the anonymous user’s root directory. If the required executable files are not installed in the anonymous user’s home directory, SYSLOGD contains error messages. For information about setting up the anonymous user’s root directory, refer to the z/OS Communications Server: IP Configuration Guide.

If you did not specify a user ID on the ANONYMOUS start parameter or FTP.DATA statement, ensure that the user ID ANONYMO is defined to TSO and RACF and that it has a defined OMVS segment or that a default OMVS segment exists for your system. For information about the z/OS UNIX environment and its security considerations, refer to z/OS UNIX System Services Planning.

If you did specify a user ID on the ANONYMOUS start parameter or FTP.DATA statement, ensure that the specified user ID is defined to TSO and RACF and that the specified user ID has a defined OMVS segment or that a default OMVS segment exists for your system.

If ANONYMOUSLEVEL is two or three, verify that the STARTDIRECTORY value is compatible with the ANONYMOUSFILEACCESS value and that the FILETYPE value is compatible with the ANONYMOUSFILETYPESEQ, ANONYMOUSFILETYPEJES, and ANONYMOUSFILETYPESQL values.

If ANONYMOUSLEVEL=3 and if ANONYMOUS or ANONYMOUS/USERID/PASSWORD is coded, the user is prompted to enter an e-mail address as a password. Verify that the e-mail address entered by the user is consistent with the requirements of the EMAILADDRCHECK statement in FTP.DATA. If
ANONYMOUS/USERID is coded, the user must provide the password for USERID. Refer to the z/OS Communications Server: IP Configuration Reference for more information about these FTP.DATA statements.

Wrong initial working directory: If the initial working directory is userid instead of a z/OS UNIX file system directory, ensure that the STARTDIRECTORY statement is specified in the FTP.DATA data set and that the $HOME directory (defined or defaulted) exists for the login user ID.

Unable to open data connection message from server: If, after issuing a command such as RETR, STOR, or LIST, the client receives the message 425 unable to open data connection from the server, check the FTP server trace for an error.

Tip: The trace option SOC should be active when you diagnose data connection errors.

See “Diagnosing FTP server problems with traces” on page 456 for information about starting the FTP server trace. One possible trace entry is data_connect: bind() error...permission denied. If you see this trace entry, ensure that the FTP data connection port is reserved to OMVS in the PROFILE.TCPIP data set.

Example

```
PORT
  20 TCP OMVS NOAUTOLOG ;FTP data port
  21 TCP FTPD1 ;FTP control port
```

Another possible trace entry is data_connect: seteuid(0) error...Permission denied. If you see this trace entry when the trace option ACC is active, ensure that FTP has BPX.DAEMON authority.

AT-TLS problems: The FTP server and client provide a level of security using the Application Transport Transparent Layer Security (AT-TLS) protocol. The FTP server and client use the services of System SSL as described in z/OS Cryptographic Services System SSL Programming, SC24-5901. This document describes how system SSL works and also contains a topic about obtaining diagnostic information.

If you are experiencing problems with the AT-TLS support, gather AT-TLS trace information from FTP by activating security processing trace. You activate the trace before the FTP server starts by adding the DEBUG SEC statement to the server’s FTPDATA file or after the server starts (and before client connection) by using the MODIFY operator command MODIFY jobname,DEBUG=(SEC).

One of the common problems with the AT-TLS handshake is a mismatch in the ciphersuites supported by client and server. For a list of ciphersuites supported by z/OS FTP, refer to z/OS Communications Server: IP Configuration Reference.

Tip: Each ciphersuite has an associated number that is known to AT-TLS.

The following is a portion of the FTP server trace for a successful AT-TLS negotiation. In this example, the server of the FTPDATA file was coded to accept only ciphersuites (cipherspecs) 01 and 02:

```
auth: entered with mechname TLS
ftpAuth: keyring = /u/user33/keyring/key.kdb
ftpAuth: stash = /u/user33/keyring/key.sth
ftpAuth: environment_open()
ftpAuth: connect as a server
ftpAuth: environment_init()
```
ftpAuth: environment initialization complete
authClient: secure_socket_open()
authClient: cipherspecs = 0102
authClient: secure_socket_init()
tlsLevel: using TLSV1 with SSL_NULL_MD5 (01)

If the client were coded to not accept ciphersuites 01 and 02, the trace would look like this:
auth: entered with mechname TLS
ftpAuth: keyring = /u/user33/keyring/key.kdb
ftpAuth: stash = /u/user33/keyring/key.sth
ftpAuth: environment_open()
ftpAuth: connect as a server
ftpAuth: environment_init()
authClient: secure_socket_open()
authClient: cipherspecs = 0102
authClient: secure_socket_init()
authClient: init failed with rc = 402 (GSK_ERR_NO_CIPHERS)
ndSecureConn: entered
EYFT96I TLS handshake failed

Data transfer problems
This section describes various problems involving data transfer.

PASV and EPSV commands fail because no PASSIVEDATAPORTS are available: If you code the PASSIVEDATAPORTS statement in the server's FTP.DATA, you must code enough ports to accommodate the server workload. Otherwise, EPSV and PASV commands to the server fail. Syslog tracing or CTRACE indicates bind() failed with errno 1116 - address not available, and errno2 of JRBINDNoPort.

To transfer data in passive mode, the FTP server must obtain a port from the PASSIVEDATAPORTS range. Therefore, allow at least one port per simultaneous data transfer. For example, if you expect one hundred users to log in to FTP at once to transfer data, code at least one hundred ports on the PASSIVEDATAPORTS statement.

The PASSIVEDATAPORTS statement does not preclude other applications from obtaining ports in the coded range. To prevent other applications from consuming ports in the PASSIVEDATAPORT range to the exclusion of FTP, code a PORTRANGE statement in PROFILE.TCPIP with the AUTHPORT parameter, specifying some or all ports in the PASSIVEDATAPORTS range. Refer to z/OS Communications Server: IP Configuration Reference for more information about the PORTRANGE statement and PROFILE.TCPIP.

TCP/IP does not release ports that the FTP server has released until the connection associated with the port has exited the TIMEWAIT state. If all the PASSIVEDATAPORTS connections are in TIMEWAIT state, the server is not able to obtain a port to process a PASV or EPSV command. You can verify the connections are in TIMEWAIT state by issuing the netstat -a command from the USS shell. To correct this problem, increase the number of ports coded on the PASSIVEDATAPORTS statement.

Load module transfer failures: This section describes failures when transferring MVS load modules.

If the MVS load module transfers, but is not executable on the target system:
- Ensure that all hosts involved in the load module transfer are at the Communications Server for OS/390® V2R10 level or higher.
  - For proxy transfers, both servers and the client must be Communications Server for OS/390 V2R10 or higher.
- Ensure that the user did not attempt an operation that is not supported by load module transfer:
  - Ensure that the user did not attempt to rename the load module on transfer.
  - Ensure that the working directory on both the current and target systems is a load library of the correct type. An MVS load library for purposes of this support is a PDS with RECFM=U or a PDSE. Files can only be transferred between the same types of load libraries. This means that a PDS load library member must be transferred to another PDS, and a PDSE load library member must be transferred into another PDSE. The FTP client displays a terminal message EZA2841I Local directory might be a load library when a user changes local directory into a PDS or PDSE eligible for load module transfer support. The FTP server sends a 250-The working directory might be a load library reply to the client when a CWD command is processed that causes the server working directory to become a PDS or PDSE eligible for load module transfer support. If both the message and the reply are not seen when changing directories before a transfer, load module transfer processing is not be used to transfer any files between the two directories.
  - Ensure that the load modules are transferred by member names only. The current working directory on both the target and destination systems must be the load library. Fully qualifying the member names is not permitted.
- Ensure that there are no problems with the IEBCOPY invocation. If an error is detected with an IEBCOPY invocation, the FTP server or client furnishes the IEBCOPY SYSPRINT output as messages to either the console (in the server’s case) or the terminal session (in the client’s case). Specify the FSC(2) debug option for the general trace for the FTP client and for the FTP server to display the IEBCOPY SYSPRINT output for both successful and unsuccessful transfers. At the client, enter debug fsc(2) before the transfer. See “Start tracing” on page 457 for information about how to set the trace for the server.

If the MVS load module fails to transfer, check the following:

1. If Reload of the load library failed or Unload of the load library failed messages or replies are seen, then these messages indicate a problem with a call to the IEBCOPY system utility. Ensure that the IEBCOPY system utility is installed on the system and available to be called from application programs. If so, examine the FTP debug trace to determine if IEBCOPY was successfully invoked (see the “Diagnosing FTP server problems with traces” on page 456 for information about activating FTP syslog tracing.) (Some client environments, particularly REXX scripts running under the UNIX system services shell, are not fully authorized to call IEBCOPY). If IEBCOPY was successfully invoked, examine the IEBCOPY SYSPRINT output (described above) to see if IEBCOPY reported any errors.

2. If allocation failure messages or replies are seen, then:
   - If the data set whose allocation failed is either the source or destination load library, ensure that no other process has allocated the load library for exclusive use.
   - If no data set name appears, or if the data set name ends in the characters XLMT, ensure that sufficient temporary DASD is available on the system. Load module transfer requires the use of sufficient temporary DASD to hold all data that could be transferred in one transfer command. Consider
breaking up large mget or mput transfers into smaller groups to reduce the amount of required temporary DASD. If sufficient temporary DASD is not immediately available, then the setting of the AUTOMOUNT/NOAUTOMOUNT site option regulates whether or not FTP attempts to mount additional temporary storage to complete a load module temporary file allocation request.

If the MVS load module transfer hangs, the system is probably waiting for temporary DASD to be mounted. If your system does not respond promptly to mount requests for temporary DASD, consider setting the NOAUTOMOUNT (LOC)SITE option about the hanging system, and breaking up large load module transfer mgets and mputs into smaller requests to reduce the requirement for temporary DASD.

**Data set allocation fails:** If data set allocation is failing (MKD, STOR/STOU, or APPE), check for the following:

- Issue the STAT command and check for problems with the variables that define data set characteristics (LRECL, RECFM, BLKSIZE, PRIMARY, SECONDARY, or DIRECTORY).
  - Do they all have a valid value defined?
  - If the variable is not listed in the STAT command output, no value is assigned to this variable. If no value is assigned to the variable, the value must be picked up from another source — either a model DCB or SMS. Does either the DCBDSN or DATACLASS (SMS) parameter have a valid value to provide a source for the missing variables?
  - If an SMS data class is specified, is SMS active at the server system? (current SMS status is displayed as part of the output for the STAT command).
  - If an SMS data class is specified, do the data class definitions contain values for the missing variables?
  - Are both PRIMARY and SECONDARY either specified or not specified? If either PRIMARY or SECONDARY are specified, neither of the values are picked up from an SMS data class. Both must be unspecified to pick up the value from SMS or both must be specified to override the SMS values.
  - If a model DCB is specified, are the characteristics of this data set valid for the data set being allocated?

- Issue the STAT command and check the PRIMARY, SECONDARY, and SPACETYPE values to determine how large the new data set is. The VOLUME and UNIT value of the STAT command indicate where the data sets are allocated. (If neither volume or unit is shown by the STAT command, data sets are allocated on the system default SYSDA DASD.) Does the server system have sufficient space where the data sets are allocated to allocate the data set? The SITE QDISK command provides information about the space available at the server system.

- Ensure that the destination at the server site is writable. Check with the operator at the server system to verify that the destination of the new data set is not write protected.

**Data set allocation not picking up correct characteristics:** If the data set is being allocated successfully, but the resulting data set does not have the expected data set characteristics, check for the following:

1. All values obtained from SITE variables
   - Issue the STAT command to verify that the settings of all the SITE variables are correct. If any variables are missing from the STAT output, check for
values specified for the DCBDSN or DATACLASS parameters. If a value is specified for the DCBDSN data set, go to Step 3. If a value is specified for the DATACLASS parameter, go to Step 2.

- Check for variables overridden by a client. The VM and MVS FTP clients automatically issue SITE commands when doing a STOR, STOU, or APPE command. The values sent automatically by the client could be overriding values set by specific SITE commands issued by the user. To prevent the VM or MVS client from automatically sending new SITE settings, issue the SENDSITE command at the client.

2. Values from SMS

If the DATACLASS parameter has been specified, but the actual data set characteristics do not match the values in the specified SMS data class, issue the STAT command and check the information shown in the output from the STAT command for the following:

- Is SMS active at the server system? If SMS is not active, the SMS data class cannot be used to define the data set.

- Are values specified for any of the data set characteristic variables (LRECL, RECFM, BLKSIZE, PRIMARY, SECONDARY, RETPD, or DIRECTORY)? If these keywords are missing from the STAT output, no value is assigned to them and the data set characteristics should be picked up from the SMS data class. If, however, a value is present for any of these variables, the setting shown by the STAT command overrides any information in the SMS data class. To pick up the value from the data class, issue the SITE command with the keyword with no value (for example, SITE RECFM) to turn off the parameter setting.

- Is a value specified for the DCBDSN parameter? If a DCBDSN data set is specified, the values for LRECL, RECFM, BLKSIZE, and RETPD are obtained from the model DCB data set and overrides any values in the SMS data class. Issue the SITE DCBDSN command to turn off the DCBDSN parameter setting.

- Check for variables overridden by a client. The VM and MVS FTP clients automatically issue SITE commands when doing a STOR, STOU, or APPE command. The values sent automatically by the client could be overriding values set by specific SITE commands issued by the user. To prevent the MVS or VM client from automatically sending new SITE settings, issue the SENDSITE command at the client.

3. Values from DCBDSN

If the DCBDSN parameter has been specified, but the actual data set characteristics do not match the characteristics of the specified data set, issue the STAT command and check the information shown in the output from the STAT command the following:

- Are values specified for any of the data set characteristic variables (LRECL, RECFM, BLKSIZE, or RETPD)? If these keywords are missing from the STAT output, no value is assigned to them and the data set characteristics are picked up from the DCBDSN data set. If, however, a value is present for any of these variables, the setting shown by the STAT command overrides the values of the DCBDSN data set. To pick up the value from the DCBDSN data set, issue the SITE command with the keyword with no value (for example, SITE RECFM) to turn off the parameter setting.

- Are variables being overridden by a client? The VM and MVS FTP clients automatically issue SITE commands when doing a STOR, STOU, or APPE command. The values sent automatically by the client could be overriding
values set by specific SITE commands issued by the user. To prevent the VM
or MVS client from automatically sending new SITE settings, issue the
SENDSITE command at the client.

**MVS data set not found:** If the server is not able to find the MVS data set, check
for the following problems:

- Can the server find the data set to list it? Issue the DIR command to display the
data set.
- Is the MVS data set at the server in the catalog? The server can only locate
cataloged MVS data sets. Check the user level of access to the catalog. FTP
servers at the z/OS V1R2 level and later display only the data sets to which the
user has access.
- Was the pathname on the FTP command entered in single quotation marks? If
not, the path name specified is appended to the end of the current working
directory. Issue the PWD command to display the current working directory. If
current_working_directory.pathname is not the correct name of the file, either
change the current working directory with the CWD command or issue the
correct data set name in single quotation marks as the *pathname*.

**RETR, STOR, RNFR, RNTO, APPE, or DELE of data set fails:** If RETR, STOR,
RNFR, RNTO, APPE, or DELE for the data set fails, check for the following
problems:

1. Is the data set protected by a security system, such as RACF or permission
   bits or a retention period?
2. Is the data set being used at the server site by another program or user?
3. Was the data set available to the system, or was it migrated or on an
   unmounted volume?
4. Did the data set or member exist?
5. For RETR or STOR commands, did a REST command immediately precede
   the RETR or STOR?
   If so, the client is attempting to restart a file transfer. The server cannot
detect certain REST argument errors until the RETR or STOR command is
processed. If the trace options CMD and FSC are active, the server reply and
server trace output provide insight into whether the REST command is
implicated. Verify that the client and server have reestablished the original
file transfer environment before attempting the restart.

The following problems apply to MVS data sets only:

1. Did the specified path name follow MVS data set naming conventions?
2. Was the requested data set a type of supported data set organization (PS,
PDS, or PDS member) on a supported device type (DASD or tape)?
3. Were the path name specifications consistent with the type of data set? For
   example, if a member was requested, was the data set a PDS?

**REST fails:** Use the STAT command to determine the current mode.

If mode is Block, report the problem to IBM.

If the mode is Stream, check the following:

- Verify that the server is configured for stream mode restarts. The server FEAT
  reply includes REST STREAM if the server is configured correctly.
Inspect the REST reply for more insight into the reason the server rejected the REST command. Refer to z/OS Communications Server: IP and SNA Codes for more information about FTP server replies.

Data transfer terminated: If data transfer terminated, check for the following problems:
1. Is the data set at the server large enough to receive the data being sent? If not, use the SITE command to change the space allocation for new data sets.
2. If storing a member of a PDS, is there room in the PDS for an additional member? Is there room in the PDS directory for another directory entry?
3. Did the client send an ABOR command?
4. Is the file type correct? For example, if filetype=SQL when it should be set to SEQ or JES, the host file being retrieved is assumed to be a SQL statement and FTP attempts to connect to DB2 and submit the statement to DB2 for processing.

Client abends during RETR command data transfer: If the client abends while processing a RETR command, issue the STAT command, and check the value of the checkpoint interval. If this value is greater than zero and data is being transferred in EBCDIC, either block mode or compressed mode, the server is sending checkpoint markers with the data being transferred. If the client being used does not support checkpoint/restart, this checkpoint information can cause unpredictable results, such as abends or data errors at the client. Change the setting of the checkpoint interval by issuing SITE CHKPTINT=0.

Data set disposition incorrect when transfer fails: If the data set disposition is incorrect when transfer fails, check for the following problems:

- Data sets cataloged instead of deleted
  - Issue the STAT command and check the setting of the conditional disposition. If the STAT command output indicates New data sets will be catalogued if a store operation ends abnormally, the server catalogs new data sets, even if the data transfer fails. To change this setting, issue the SITE CONDDISP=DELETE command.
  - Did the transfer fail because the FTP server was either abending or being terminated by a STOP or CANCEL command? If this is the case, the data set is kept.
  - Is the client sending checkpoint information? If the data is being transferred in EBCDIC, either in block mode or compressed mode and the client has sent at least one checkpoint marker, the FTP server keeps the data set even if the conditional disposition is set to delete.

- Data sets deleted instead of cataloged
  - Issue the STAT command and check the setting of the conditional disposition. If the STAT command output indicates New data sets will be deleted if a store operation ends abnormally, the server deletes new data sets if the data transfer fails. To change this setting, issue the SITE CONDDISP=CATALOG command.

Checkpoint markers do not appear to be sent: Issue the STAT command and check the settings for data transfer. Checkpoint information is only transferred in EBCDIC, with either block or compressed mode. The checkpoint interval must be greater than zero.

The sender of the data initiates the checkpoint information. Therefore, checkpointing must be set on at the client for a STOR, STOU, or APPE.
MVS FTP client, this is done by issuing the LOCSITE CHKPTINT=nn command with a value larger than zero) and set on at the server (by issuing the SITE CHKPTINT=nn command with a value larger than zero) for a RETR.

LOADLIB directory information is not sent with module transfer: Issue the STAT command and check the settings for data transfer. Load module directory information is only sent for EBCDIC with a mode of either block or compressed.

Restriction: The client you are using must support the SDIR command.

Server PDS member statistics not created or updated: ISPFStats must be set to TRUE in order to create or update the statistics for the PDS Member when using PUT, MPUT, GET, MGET, or APPEND subcommands. For PUT, MPUT, or APPEND, make sure the server’s ISPFStats is set to TRUE. Issue the STAT command to determine this. If it is not set to TRUE, you can set it by using the SITE subcommand. For example, SITE ISPFStats sets ISPFStats to TRUE, and SITE NOISPFStats sets ISPFStats to FALSE.

Result: If the PDS directory block is full, PDS member statistics are not updated.

File transfers to the BatchPipe subsystem fail
If a file transfer to a batch pipe fails, the problem could be that the batch pipe reader has not been started. Verify that the batch pipe reader is active.

Guideline: In a JESMAS environment, if you use FTP to submit a job starting the batch pipe reader, the job can run on any system in the JESMAS environment unless you include this JCL statement in your job:

/*JOBPARM SASAFF==

If your file transfer runs on a different system in the JESMAS environment, the JWT time limit will expire, and the FTP server JOB will appear to hang. You can avoid this by adding the JCL statement:

/*JOBPARM SASAFF==

to your batch pipe job.

When a file transfer to a batch pipe fails with an allocation error, the problem could be that the BatchPipe subsystem has not been started. Verify that the BatchPipe subsystem is started. For example:

put 'user3.source.data' 'user3.subsys.output1'
>>> SITE FIXrecfm 80 LRECL=80 RECFM=FB BLKSIZE=12960
200 SITE command was accepted
>>> PORT 9,42,104,22,4,15
200 Port request OK.
>>> STOR 'user3.subsys.output1'
550 Allocation of USER3.SUBSYS.OUTPUT1 failed while executing STOR command.
Command:

You can activate the FTP server trace to obtain more information about the failure. See “Diagnosing FTP server problems with traces” on page 456 for information on activating the FTP server trace. Inspect the SYSLOG output for messages related to the file transfer failure.

A sample syslog output is:

Mar 2 16:29:47 MVS117 ftps[21]: GU1850 logALLOC: ALLOC error in routine alloc_dasd
Mar 2 16:29:47 MVS117 ftps[21]: GU1852 logALLOC: SVC99 ALLOC failed with rc 4
Mar 2 16:29:47 MVS117 ftps[21]: GU1852 logALLOC: dsname = USER3.SUBSYS.OUTPUT1
Error code 04AC means that the subsystem is not operational.

When FTPLOGGING is set to TRUE in the servers FTP.DATA file, the error messages will be logged in SYSLOGD. You can look up the messages and error codes.

### Diagnosing FTP server problems with traces

Syslog tracing is available to aid in debugging z/OS UNIX FTP server problems. The following methods are available to start, stop, or modify syslog daemon and server tracing:

- TRACE start option
- FTP.DATA DEBUG statement
- FTP.DATA DUMP statement
- MODIFY jobname,DUMP operator command
- MODIFY jobname,DEBUG operator command
- server SITE DEBUG command
- server SITE DUMP command

Refer to the following for more information:

- See “Start tracing during FTP initialization” on page 457 and the z/OS Communications Server: IP Configuration Reference for details about the TRACE start option and FTP.DATA statements.
- See “Controlling the FTP server traces with MODIFY operator command” on page 458 and the z/OS Communications Server: IP System Administrator’s Commands for details about the MODIFY operator command.

After a client has logged in to FTP, the client can issue SITE DEBUG or SITE DUMP commands to change tracing for that session only.

### Where to find traces

The z/OS UNIX FTP server sends its trace entries to syslogd. As shown in the following example, the daemon.debug statement in /etc/syslog.conf specifies where syslogd writes FTP trace records:

```
# All ftp, rexec, rsh
# debug messages (and above
# priority messages) go
# to server.debug.a
# daemon.debug /tmp/syslogd/server.debug.a
```

All z/OS UNIX FTP trace entries are written to the same z/OS UNIX file system file.
Note: The TRACE parameter and MODIFY operator command options are issued to the FTP daemon and affect all client sessions that connect to the z/OS UNIX FTP server while tracing is active.

Refer to the z/OS Communications Server: IP Configuration Guide for more information about syslogd.

**Start tracing**

This section discusses the following methods of starting the FTP server traces:

- During FTP initialization
- After FTP initialization

**Start tracing during FTP initialization:** You can use the TRACE start parameter, the TRACE statement, or the DEBUG and DUMP statements in FTP.DATA to begin tracing during FTP daemon initialization. This continues tracing for all FTP events for all FTP sessions. The trace data is routed to a file in your z/OS UNIX file system through a definition in your syslogd configuration file (/etc/syslog.conf).

Tracing remains active until you issue a MODIFY operator command to end it. See “Controlling the FTP server traces with MODIFY operator command” on page 458

**Tip:** When you issue a MODIFY operator command to end tracing, tracing does not occur for any subsequent client sessions; however, tracing continues for any sessions that were already connected.

**Start tracing after FTP initialization:** After initialization, you can enable tracing using an MVS MODIFY operator command to the FTP server listener process. See “Controlling the FTP server traces with MODIFY operator command” on page 458. Previously established FTP connections are not affected by a MODIFY operator command. Only FTP connections that are established after the MODIFY operator command was issued are subject to tracing.

If you have coded DEBUGONSITE TRUE and DUMPONSITE TRUE in the server's FTP.DATA file, you can use the SITE DEBUG command and the SITE DUMP command, respectively, to change tracing after you log in to FTP. For example, if you want to add JES general tracing and JES extended tracing, enter the following: SITE DEBUG=(JES) DUMP=(JES)

If you want to restrict the use of the SITE command to change the tracing and your installation has a security product that supports the SERVAUTH class, you can provide additional levels of access control. If the installation has activated the SERVAUTH class and provided a profile for the SITE DEBUG command, only users who have read access to the profile are allowed to use the SITE DEBUG command. The profile name is:

EVB.FTP.systemname.ftpdaemonname.SITE.DEBUG

For example, if the procedure FTPD is used to start the server on system MVS164, the profile name is:

EVB.FTP.MVS164.FTPD1.SITE.DEBUG

The user's SITE DEBUG command is rejected if the security product determines that the user does not have read access to the profile.
If the installation has activated the SERVAUTH class and provided a profile for the SITE DUMP command, only users who have read access to the profile are allowed to use the SITE DUMP command. The profile name is:

```
EZB.FTP.<systemname>.<ftpdaemonname>.SITE.DUMP
```

For example, if the procedure FTPD is used to start the server on system MVS164, the profile name is:

```
EZB.FTP.MVS164.FTPD1.SITE.DUMP
```

The user’s SITE DUMP command is rejected if the security product determines that the user does not have read access to the profile.

**Stop tracing**

Use the MODIFY operator command to stop global tracing. For example, your FTP jobname is FTPD1. You can issue `F FTPD1,DEBUG=(NONE)` to stop global tracing. Previously established FTP connections that were started with tracing enabled continue to produce trace output until the connections are terminated, but new connections start without tracing enabled.

If you have coded DEBUGONSITE TRUE in the server’s FTPDATA, the FTP client can use a SITE DEBUG=NONE command to stop tracing. The SITE command affects only tracing for the current FTP session.

**Tracing activity for one user**

A filter can be specified so that the traces are active only for certain clients that log in. Trace data can include both general and JES-related activity and includes data such as parameter lists and storage areas. The filtering can be done by either IP address of the client or by user ID for the session. Use the IPADDR(filter) and USERID(filter) operands on the FTP SITE command, or on MODIFY operator command, to enable trace filtering.

A client could use the SITE DEBUG and SITE DUMP subcommands to write excessive debugging information to the syslog and effectively disable the syslog function. To prevent this, a RACF profile controls whether a client is allowed to use these parameters on the SITE subcommand. FTP uses the SERVAUTH resource class. The resource name is

```
EZB.FTP.<systemname>.<ftpdaemonname>.SITE.<tracename>
```

The lowest level is tracename, which is either DEBUG or DUMP.

**Controlling the FTP server traces with MODIFY operator command**

To start the general trace for the FTP server for all user IDs during initialization, specify the TRACE parameter either as a start option in the FTP server start procedure, or code a DEBUG BAS statement in FTPDATA.

After initialization, use the MODIFY operator command to control the general and extended tracing for the FTP server. The command supports the following parameters:

- DEBUG for general tracing
- DUMP for extended tracing

Each allows a filter to be specified so that the traces are active for certain clients that log in. The filtering can be done by either IP address of the client or by user ID for the session.
**Guideline:** The *jobname* is the name associated with the FTP daemon background job. It is documented in message EZYFT41I in SYSLOGD. If you started the z/OS UNIX server using a procedure named FTPD, the job name to use for the MODIFY operator command is probably FTPD1. As client sessions connect to the FTP server, the session process adopts the trace options currently active. These options remain in effect for the life of the client session process, regardless of subsequent MODIFY operator commands issued to the FTP daemon.

**Controlling general tracing:** To control the general trace, enter one of the following:

- `MODIFY jobname,DEBUG=(option_1,option_2,...,option_n,USERID(filter_name))`
- `MODIFY jobname,DEBUG=(option_1,option_2,...,option_n,IPADDR(filter))`

Where options are one of the following:

- `?` Displays the status of the general traces.
  - The status of the trace is displayed as a response to all uses of the operator MODIFY DEBUG command. The `?` allows you to get the status without making a change.

- `ACC` Shows the details of the login process.

- `ALL` Sets all of the trace points.
  - When the ALL parameter is processed, both the FSC and the SOC trace are set to level 1.

- `BAS` Sets a select group of traces that offer the best overall details without the copious output generated by certain trace options. Specifying this value is the same as the following:
  - `MODIFY jobname,DEBUG=(CMD,INT,FSC,SOC)`

- `CMD` Shows each command and the parsing of the parameters for the command.

- `FLO` Shows the flow of control within FTP. It is useful to show which services of FTP are used for an FTP request.

- `FSC(n)` Shows details of the processing the following file services commands
  - APPE
  - STOR
  - STOU
  - RETR
  - DELE
  - RNFR
  - RNTO

  This trace can be very intense; therefore, it allows you to specify levels of granularity for the trace points. The level 1 tracing that is specified by entering FSC or FSC(1) is the level normally used unless more data is requested by TCP/IP service group. The variable *n* can be a number in the range 1–8.
Level 1
Covers the major steps of the file services processing, which includes the following:
- Entry to a command processor
- Determination of the type of file being processed
- Choice of allocation method
- Choice of open method
- Choice of transfer routine
- Recognition of end of file or data
- Close and deallocation
- Call for SMF processing

Level 2
Provides more details for the major steps that are executed. These should be one-time events that enhance the information for the steps of level 1 tracing. An example would be some additional information about the allocation process.

Level 3
Provides trace information of repetitive events that occur during the processing. For example, a trace for each full buffer (180K) of data that is received. Another example is a trace for each restart marker that is sent. The rate of repetition should be low enough that this level does not flood the trace.

Level 4
Provides trace information of repetitive events that occur at a higher rate than those of level 3. For example, a trace for each time data must be moved to the top of a buffer before the next receive_data.

Level 5
Provides trace information of repetitive events that occur at a higher rate than those of level 4. This is the most intense and covers events such as the processing of each block of data.

Tip: This level of tracing produces an extremely large amount of data and should not be used for large file transfers.

INT
The INT trace shows the details of the initialization and termination of the FTP session.

JES
The JES trace shows details of the processing for JES requests, such as when SITE FILETYPE=JES is in effect.

NONE
This value is used to turn off all of the traces.

PAR
The PAR trace shows details of the FTP command parser. It is useful for debugging problems in the handling of the command parameters.

SEC
The SEC trace shows the processing of security functions such as AT-TLS and GSSAPI negotiations.

SOC(n)
The SOC trace shows details of the processing during the setup of the interface
between the FTP application and the network as well as details of the actual amounts of data that is processed. This trace can be very intense; therefore, it allows you to specify levels of granularity for the trace points. The level 1 tracing that is specified by entering SOC or SOC(1) is the level normally used unless more data is requested by the TCP/IP service group. The variable n can be a number from 1 to 8.

**Level 1**
Covers the major steps of the socket services processing. Connection initiation and closing steps are included.

**Level 2**
Adds more detail for level 1 events. For example, it traces the three steps that occur when a data connection is closed.

**Level 3**
The events for this trace are the send() and recv() calls for the data connection.

**SQL**
Shows details of the processing for SQL requests, such as when SITE FILETYPE=SQL is in effect.

**UTL**
Shows the processing of utility functions such as CD and SITE.

**USERID(filter_name)**
Filters the trace for user IDs matching the filter_name pattern.

If the user ID matches the filter at the time the clients log in, their tracing options are set to the current value of the options. Otherwise, tracing options are not set. Clients can use the SITE command to set their options after login if the initial ones are not appropriate. An example for the USERID filter is MODIFY jobname,DEBUG=(CMD,USERID(USER3*)), which activates the CMD trace for a user whose ID starts with USER3.

**IPADDR(filter)**
This optional parameter filters the trace for IP addresses matching the filter pattern.

If the IP address matches the filter at the time clients connect, its tracing options are set to the current value of the options. Otherwise, tracing options are not be set. Clients can use the SITE command to set their options after connect if the initial ones are not appropriate. An example of the IPADDR filter is MODIFY jobname,DEBUG=(JES,IPADDR(9.67.113.57)), which activates the JES trace for a client whose IP address is 9.67.113.57. Another example is MODIFY jobname,DEBUG=(JES,IPADDR(FEDC:BA98::0/32)), which activates the JES trace for a client whose IP address is FEDC:BA98::0/32.

If the filter is an IPv4 address, submasking can be indicated by using a slash followed by a dotted decimal submask. For example, 192.48.32/255.255.255.0 allows addresses from 192.48.32.0 to 192.48.32.255.

If the filter is an IPv6 address, network prefixing can be indicated by using a slash followed by a prefix length. For example, FEDC:BA98::0/32 allows all IP addresses from FEDC:BA98::0 to FEDC:BA98:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF.

The specification of the trace on the MODIFY operator command is not additive. That is, the trace setting is that of the last MODIFY operator command. For example:
MODIFY FTPDJG1,DEBUG=(NONE)
+EZYFT82I Active traces: NONE
MODIFY FTPDJG1,DEBUG=(CMD)
+EZYFT82I Active traces: CMD
MODIFY FTPDJG1,DEBUG=(FSC,USERID(USER33))
+EZYFT82I Active traces: FSC(1)
+EZYFT89I Userid filter: USER33
MODIFY FTPDJG1,DEBUG=(SOC)
+EZYFT82I Active traces: SOC(1)

Guidelines: The following are some guidelines to use for migrating from previous versions of the MODIFY operator command:

- **MODIFY jobname,TRACE**
  This is still accepted and is equivalent to MODIFY jobname,DEBUG=(BAS). The old response message EZY2704I is replaced by EZYFT82I.

- **MODIFY jobname,NOTRACE**
  This is still accepted and is equivalent to MODIFY jobname,DEBUG=(NONE). The old response message EZY2705I is replaced by EZYFT82I.

- **MODIFY jobname,JTRACE**
  This is still accepted and is equivalent to MODIFY jobname,DEBUG=(CMD,FSC,JES).
  The old response message EZY2710I is replaced by EZYFT82I.

- **MODIFY jobname,NOJTRACE**
  This is still accepted and is equivalent to MODIFY jobname,DEBUG=(NONE). The old response message EZY2711I is replaced by EZYFT82I.

- **MODIFY jobname,UTRACE=USER33**
  This is rejected as an obsolete command. Its function can be replaced with the following pair of commands:
  
  MODIFY jobname,DEBUG=(ALL,USERID(USER33))
  MODIFY jobname,DUMP=(ALL,USERID(USER33))

- The use of the ALL parameter can produce an extensive amount of trace data and should not be specified on a routine basis.

- **MODIFY jobname,NOUTRACE**
  This is rejected as an obsolete command. If complete tracing was activated as suggested in the previous step, then the tracing can be stopped as follows:
  
  MODIFY jobname,DEBUG=(NONE)
  MODIFY jobname,DUMP=(NONE)

Controlling extended tracing: To control the extended trace, enter one of the following:

- **MODIFY jobname,DUMP=(option_1,option_2,...,option_n,USERID(filter_name))**

- **MODIFY jobname,DUMP=(option_1,option_2,...,option_n,IPADDR(filter))**

Where options are one of the following:

- **id**  Specifies the ID number of a specific extended trace point that is to be activated in the FTP code. The ID number has a range of 1=99.
- **?**   Displays the status of the extended traces.
- **ALL** Activates all of the trace points.
- **NONE** Resets (turns off) all extended traces.
FSC
Activates all of the extended trace points in the file services code. The numbers activated are 20–49.

SOC
Activates all of the extended trace points in the network services code. The numbers activated are 50–59.

JES
Activates all of the extended trace points in the JES services code. The numbers activated are 60–69.

SQL
Activates all of the extended trace points in the SQL services code. The numbers activated are 70–79.

USERID(filter_name)
Filters the trace for user IDs matching the filter_name pattern.

If a client’s user ID matches the filter when the client logs into the server, its tracing options are set to the current value of the options. Otherwise, tracing options are not set. Clients can use the SITE command to set their options after login if the initial ones are not appropriate. An example for the USERID filter is MODIFY jobname,DEBUG=(21,USERID(USER33)), which activates the dumpID 21 trace for a user if his user ID is USER33.

IPADDR(filter)
Filters the extended trace for IP addresses matching the filter pattern.

If the client’s IP address matches the filter when the client connects to the FTP server, its extended tracing options are set to the current value of the options. Otherwise, tracing options are not set. Clients can use the SITE command to set their options after connect if the initial ones are not appropriate.

An example of the IPADDR filter is MODIFY jobname,DUMP=(JES,IPADDR(9.67.113.57)), which activates the JES extended trace for a client whose IP address is 9.67.113.57. Another example is MODIFY jobname,DUMP=(FSC,IPADDR(FEDC:BA98:7654:3210:0/32)), which activates all file services extended traces for a client whose IP address is FEDC:BA98:7654:3210:0/32.

If the filter is an IPv4 address, submasking can be indicated by using a slash followed by a dotted decimal submask. For example, 192.48.32/255.255.255.0 allows addresses from 192.48.32.0 to 192.48.32.255.

If the filter is an IPv6 address, network prefixing can be indicated by using a slash followed by a prefix length. For example, FEDC:BA98::/64 allows all IP addresses from FEDC:BA98:: to FEDC:BA98:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF.

The specification of the trace on the MODIFY operator command is not additive. That is, the trace setting is that of the last MODIFY operator command. For example:

MODIFY FTPDJG1,DUMP=(NONE) +EZYFT83I Active dumpIDs: NONE
MODIFY FTPDJG1,DUMP=(21) +EZYFT83I Active dumpIDs: 21
MODIFY FTPDJG1,DUMP=(22) +EZYFT83I Active dumpIDs: 22

Guidelines: The following are guidelines for migrating from the old parameters that were used with the MODIFY operator command:
• MODIFY jobname,DUMP
  This format is rejected. DUMP requires at least one parameter (see above).
• MODIFY jobname,NODUMP
  This is still accepted and is equivalent to MODIFY jobname,DUMP=(NONE). The old
  response message EZY2656I is replaced by EZYFT83I.
• MODIFY jobname,JDUMP
  This is rejected as an obsolete command with a suggestion to use the DUMP
  parameter. For example, use the command MODIFY jobname,DUMP=(JES).
• MODIFY jobname,NOJDUMP
  This is rejected as an obsolete command with a suggestion to use the DUMP
  parameter. For example, use the command MODIFY jobname,DUMP=(NONE).

**Documenting server problems**

If the problem is not caused by any of the common errors described in this section,
collect the following documentation before calling the IBM Support Center.

Documentation is divided into the following categories: essential and helpful (but
not essential).

• Essential
  - Precise description of problem, including expected results and actual results
  - z/OS UNIX FTP server dump (for abends)
  - z/OS UNIX FTP server traces (see "Diagnosing FTP server problems with
    traces" on page 456 for information about collecting FTP server traces)
    - Minimum for initial problem reporting: DEBUG BAS

• Helpful
  - FTP client output
    - If the FTP client is a z/OS client, include a trace in the output by one of
      these methods:
      • Coding DEBUG statements in the client's FTP.DATA file. See [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com) for information about the DEBUG statement.
      • Invoking the FTP client with the -d or TRACE invocation option. See [z/OS Communications Server: IP User's Guide and Commands](https://www.ibm.com) for more information.
      • Specifying a DEBUG subcommand in the client command input stream
        before the affected transfer. Use this option only if the problem does not
        involve the initial establishment of the FTP control session.
  - Server FTP.DATA data set
  - TCPIP.DATA data set
  - PROFILE.TCPIP data set
  - ETC.SERVICES data set
  - The reply from the STAT or XSTA command issued to the server.

**Guidelines:**

- Issue the STATus subcommand from the z/OS FTP client to retrieve STAT
  command output from the server. From non z/OS clients, you may have to
  issue QUOTE STAT to retrieve the output from the server.
- Issue the STATus subcommand with a parameter from the z/OS FTP client
to issue the XSTA command to the server. The XSTA command retrieves
FTP client

This section describes the following topics:
- “Execution environments”
- “Setup”
- “Naming considerations” on page 466
- “Directing the client to exit when an error occurs” on page 466
- “Translation and data conversion support” on page 466
- “File tagging support” on page 468
- “DB2 query support” on page 471
- “Restarting file transfers” on page 473
- “Diagnosing FTP connection and transfer failures with EZA2589E” on page 474
- “Problems starting the client” on page 479
- “Problems logging into the server” on page 480
- “Problems transferring data” on page 482
- “Other problems” on page 485
- “Diagnosing FTP client problems with tracing” on page 485
- “Documenting FTP client problems” on page 486

Execution environments

The FTP client can run in any of the following environments:
- Interactive (under the TSO or the z/OS UNIX shell)
- Batch (under TSO only)
- REXX exec (under TSO)

When run interactively, you can redirect terminal I/O. When run under TSO, server responses and debug messages can be redirected to a file. For example, you can use the ftp 9.68.100.23 > 'USER27.FTPOUT' command to redirect output from a TSO command line to a data set. When run under the z/OS UNIX shell, both input and output can be redirected. To redirect input from the file /user27/ftpin and output to the file /user27/ftpout, issue the following command: ftp 9.68.100.23 > /user27/ftpout < /user27/ftpin.

Tip: When redirecting output under z/OS UNIX, nothing is displayed on the system console, not even command prompts, and it is difficult to know when input is requested. Consequently, use output redirection only when also using input redirection.

Setup

Use an FTP.DATA data set to customize configuration parameters. You can use a SOCKSCONFIGFILE data set or file to instruct the client to connect to certain FTP servers through a SOCKS server. For information about the FTP.DATA data set and SOCKS configuration data set or file used by the FTP client, refer to z/OS.
Message EZY2640I displays the name of the FTP.DATA file. Use the FTP client locstat subcommand to display the name of the SOCKS configuration data set or file that is being used.

The TCPIP.DATA configuration file provides information for the FTP client, such as the high-level qualifier to be used for configuration data sets and which DBCS translation tables can be used. For more information about the TCPIP.DATA configuration file, refer to the z/OS Communications Server: IP Configuration Reference.

Tip: The z/OS UNIX search order for the file is used even if the FTP client is invoked under TSO.

Naming considerations
The FTP client can access both MVS data sets and z/OS UNIX file system files. For more information, see “Name considerations for z/OS UNIX FTP” on page 436.

Directing the client to exit when an error occurs
You can direct the FTP client to exit whenever an error occurs, rather than to continue processing. You also have some control over whether the client exits with a generic return code or with a return code that reflects the type of error that occurred. For a description of all the FTP client return code options, refer to the z/OS Communications Server: IP User’s Guide and Commands.

Translation and data conversion support
This section describes translation and data conversion support for the FTP client.

Double-byte character set (DBCS) support
If the DBCS translate tables are not available, the client issues the following message after a valid command to establish a double-byte transfer type (for example, SJISKANKI, BIG5, or ‘TYPE B n’) is entered:
"EZA1865I Command not Supported. Translation Table not Loaded.

If this message is displayed, check the LOADDBCSTABLES statement in the TCPIP.DATA file. If the statement wraps to the next line, parameters on the continued line are ignored, and no error message is issued. If all parameters for the LOADDBCSTABLES statement do not fit on one line, use multiple LOADDBCSTS statements.

Check the precedence order for the TCPIP.DATA file to ensure that the file being used contains the LOADDBCSTABLES statement or statements. Be aware that the location of TCPIP.DATA statements can be influenced in multiple ways. For example, by a GlobalTCPIPData specification or the RESOLVER_CONFIG environment variable. Refer to the z/OS Communications Server: IP Configuration Guide for the TCPIP.DATA search order.

Single-byte character (SBCS) support
Data conversion occurs for single-byte data on the data connection when ENCODING=SBCS is in effect and the data type is ASCII. For more information, refer to the FTP.DATA statement ENCODING and the LOCSITE ENCODING subcommand in the z/OS Communications Server: IP User’s Guide and Commands.
If you choose SBDATACONN as a statement in the FTP.DATA file or with the LOCSITE SBDATACONN subcommand, the FTP client builds a translation table using the code pages specified by SBDATACONN. If you receive the following message from the LOCSITE subcommand, start the trace with the DEBUG UTL option to determine which characters cannot be translated:

EZYFS08I
Some characters cannot be translated between codepage_1 and codepage_2.

If none of the untranslatable characters appear in your data, your data transfers are not affected. If an untranslatable character is present in the data you are trying to transfer, your data transfer fails and you receive the following message:

EZA2930I
Transfer failed because data cannot be translated.

To avoid the failure, specify a substitution character to replace non-translatable characters. For more information about how to specify character substitution, refer to SBSUB and SBSUBCHAR as FTP.DATA statements and as parameters on the LOCSITE subcommand in z/OS Communications Server: IP User’s Guide and Commands. If substitution occurs during the transfer, you receive the following message:

EZA2947I
One or more characters were substituted during the transfer.

When substitution occurs at the destination of a data transfer, a subsequent transfer of the resulting data does not produce an exact copy of the original. For example, if you get a file from the server and one or more characters are substituted, the untranslatable characters are overlaid with the substitution character. You cannot restore the original file by putting it to the server.

**Multibyte character set (MBCS) support**

Data conversion occurs for multibyte data on the data connection when ENCODING=MBCS is in effect and the data type is ASCII. For more information, refer to the FTP.DATA statement ENCODING and the LOCSITE ENCODING subcommand in the z/OS Communications Server: IP User’s Guide and Commands.

If you choose ENCODING=MBCS, you must specify MB DATACONN with a statement in the FTP.DATA file or with the LOCSITE MB DATACONN subcommand to name the code pages for the multibyte data transfer. If you attempt an ASCII data transfer with ENCODING=MBCS and no MB DATACONN specified, you receive the following message:

EZZ9793I
Multibyte encoding requested but code pages are not defined.

If the multibyte data that you transfer has codepoints that cannot be translated, the data transfer fails and you receive the following message:

EZA2930I
Transfer failed because data cannot be translated.

To determine which bytes of the data cannot be translated, repeat the transfer with the DUMP 42 extended trace option active at the client.
File tagging support

When the server writes a z/OS UNIX file system file, it might tag the file using the USS support for file tagging. In some cases you might experience conflicts when you try to read a file that has been tagged. A tagged file has a file tag, which is an attribute that identifies the coded character set ID (ccsid) of the text data within the file. When a tagged file is read from the file system, the data is translated using the ccsid if SBDATACONN has specified a network transfer code page to use with the file's code page. A file might also be untagged or tagged binary.

ASCII file transfers

If you put data into a z/OS UNIX file system file when the data type is ASCII, the file is tagged if you have used SBDATACONN to specify the code page for the file system and for the network transfer. That is, you have specified SBDATACONN=(file_system_cp,network_transfer_cp). If the data conversion table is the FTP_STANDARD_TABLES or is specified using XLATE, the file is not tagged. The following client session example shows the effects of combining data type ASCII and SBDATACONN defined tables using code pages:

1 (01) Command: ascii
2 (02) >>> TYPE A
3 200 Representation type is Ascii NonPrint
4 (03) Command: site sbd=(ISO8859-1,ISO8859-1)
5 (04) SITE sbd=(ISO8859-1,ISO8859-1)
6 200 Site command was accepted
7 (05) Command: put a afile
8 (06) PORT 9,67,113,57,4,121
9 (07) STOR afile
10 125 Storing data set /u/user33/tagging2/afile
11 250 Transfer completed successfully.
12 200 bytes transferred in 0.070 seconds. Transfer rate 2.86 Kbytes/sec.
13 (08) Command: site sbd=(IBM-1047,ISO8859-1)
14 (09) SITE sbd=(IBM-1047,ISO8859-1)
15 200 Site command was accepted
16 (10) Command: put a efile
17 (11) PORT 9,67,113,57,4,122
18 (12) STOR efile
19 125 Storing data set /u/user33/tagging2/efile
20 250 Transfer completed successfully.
21 200 bytes transferred in 0.005 seconds. Transfer rate 40.00 Kbytes/sec.
22 (13) Command: site sbd=FTP_STANDARD_TABLES
23 (14) SITE sbd=FTP_STANDARD_TABLES
24 200 Site command was accepted
25 (15) Command: put a ufile
26 (16) PORT 9,67,113,57,4,123
27 (17) STOR ufile
28 125 Storing data set /u/user33/tagging2/ufile
29 250 Transfer completed successfully.
30 200 bytes transferred in 0.005 seconds. Transfer rate 40.00 Kbytes/sec.
31 (18) Command: ls -T
32 (19) PORT 9,67,113,57,4,124
33 (20) NLST -T
34 (21) LIST started OK
35 (22) T=on afile
36 (23) T=on efile
37 (24) T=off ufile
38 250 List completed successfully.
39 (25) Command: get afile
40 (26) PORT 9,67,113,57,4,125
41 (27) 200 Port request OK.
Change the data type to ASCII.

Site command requests a file system code page ISO8859-1, an ASCII code page.

Put a file and name it afile.

Site command requests a file system code page IBM-1047, an EBCDIC code page.
Put a file and name it efile.

Site command requests standard FTP translation tables.

Put a file and name it ufile.

Use the ls subcommand to determine whether files in an hfs directory are tagged (that is, have a file tag). You use the -T option to request the file tagging information. When options are specified on the ls subcommand, name parameters cannot be specified.

afile is a tagged file. Its file system code page is ISO8859-1. It is a Text file.

efile is a tagged file. Its file system code page is IBM-1047. It is a Text file.

ufile is an untagged file. It is not a Text file.

Retrieve afile, which is a tagged file.

Client receives an indication that the tagged file is translated using the current tables because the current data connection tables were not specified with a network transfer code page (see 6).

Since this is an untagged file, no indication is needed about the tables used.

Specify translation tables with a file system code page and a network transfer code page.

The code page of the tagged ASCII file is used with the network transfer code page to translate the data in the file.

The code page of the tagged EBCDIC file is used with the network transfer code page to translate the data in the file.

Change the data type to EBCDIC.

The 557 reply informs the client that the data type must be ASCII when the file that is tagged as ASCII is retrieved.

The EBCDIC file is OK to send with data type EBCDIC since no translation occurs and the data is already EBCDIC.

**Binary file transfers**

If you put data into a z/OS UNIX file system file when the data type is binary, the file is tagged as a binary file. The following client session example shows the effects of the binary file tagging:

```
(01) Command: binary
(02) >>> TYPE I
(03) 200 Representation type is Image
(04) Command: put a file
(05) >>> PORT 9,67,113,57,4,44
(06) 200 Port request OK.
(07) >>> STOR file
(08) 125 Storing data set /u/user33/newtag/file
(09) 250 Transfer completed successfully.
(10) 190 bytes transferred in 0.050 seconds. Transfer rate 3.80 Kbytes/sec.
(11) Command: ascii
(12) >>> TYPE A
(13) 200 Representation type is Ascii NonPrint
(14) Command: ls -T
(15) >>> PORT 9,67,113,57,4,45
(16) 200 Port request OK.
(17) >>> NLST -T
(18) 125 List started OK
(19) b binary T=off file
```
20) 250 List completed successfully.
21) Command: get file
22) >>> PORT 9,67,113,57,4,46
23) 200 Port request OK.
24) >>> RETR file

4) (25) 557 File contains binary data - enter TYPE I command before entering RETR command
26) Command: binary
27) >>> TYPE I
28) 200 Representation type is Image
29) Command: get file
30) >>> PORT 9,67,113,57,4,47
31) 200 Port request OK.
32) >>> RETR file
33) 125 Sending data set /u/user33/newtag/file
34) 250 Transfer completed successfully.
35) 190 bytes transferred in 0.005 seconds. Transfer rate 38.00 Kbytes/sec.

Notes:

1. Request binary data type.
2. Use the Is subcommand to determine whether files in an hfs directory are tagged. You use the -T option to request the file tagging information. When options are specified on the Is subcommand, name parameters cannot be specified.
3. The tagging information shows that the file is a binary file.
4. The 557 reply informs the client that the data type must be binary when the file is retrieved.

DB2 query support

This section describes how to use the FTP client DB2 query support and how to diagnose SQL problems.

Steps for using FTP client SQL support

Before you begin: Before you can use the FTP client to submit queries to the DB2 subsystem, complete the following steps:

1. Start the DB2 subsystem.

2. BIND the DBRM called EZAFTPMQ. This must be done whenever the part EZAFTPMQ.CSQLMVS has been recompiled.

   The DBRM must be bound into the plan named EZAFTPMQ, unless the keyword DB2PLAN was used in your FTP.DATA file to specify a different plan name.

3. Grant execute privilege to the public for the plan created in the previous step.

To use the FTP client to submit a query to DB2 and send the output to the FTP server, issue the following commands as necessary:

- **LOCSITE FILETYPE=SQL**
- **LOCSITE DB2=db2name** where db2name is the name of a DB2 subsystem at the local host
- **PUT fname1 fname2** where fname1 is a local file that contains a SQL SELECT statement
### Symptoms of SQL problems

Table 28 and Table 29 on page 473 show some symptoms and possible causes of SQL problems.

**Table 28** shows problems that generate a reply beginning with 55x.

#### Table 28. SQL problems generating 55x replies (FTP Client)

<table>
<thead>
<tr>
<th>Reply</th>
<th>Output file</th>
<th>Possible causes</th>
</tr>
</thead>
</table>
| EZA2570E: Transfer aborted: SQL PREPARE/DESCRIBE failure | The output file contains the SQL code and error message returned by the DB2 subsystem. | • A syntax error in the SQL statement in the host file.  
• The time stamp in the load module is different from the BIND time stamp built from the DBRM (SQL code = -818). This occurs if a BIND was not done for the EZAFTPMQ DBRM that corresponds to the current load module, or if the server is not configured to use the correct DB2 plan name. If this is the problem, every SQL query submitted through the FTP server fails. |
| EZA2573E: Transfer aborted: unsupported SQL statement | No output is sent from the host. | The file type is SQL, but the host file being retrieved does not contain an SQL SELECT statement. |
| EZA2568E: Transfer aborted: attempt to connect to db2name failed (code) | No output is sent from the host. | • The locsite db2name specifies a nonexistent DB2 subsystem.  
• The DB2 subsystem has not been started. |
| EZA2569E: Transfer aborted: SQL not available. Attempt to open plan <planname> failed (DB2_reason_code). | No output is sent from the host | • BIND was not done for the specified plan.  
• BIND was done for plan name other than EZAFTPMQ, but FTPDATA does not contain a DB2PLAN statement to specify this plan name.  
• User does not have execute privilege for the DB2 plan being used by the FTP server. |
| EZA2740E: SQL query not available. Cannot load CAF routines. | No output is sent from the host. | The DSNLOAD library is not in the link list or the FTP server STEPLIB. |

**Note:** For more information about these messages, refer to [z/OS Communications Server: IP and SNA Codes](https://www.ibm.com/servers/resourcelink).
Table 29 shows other SQL problems.

**Table 29. Other SQL problems (FTP Client)**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Possible causes</th>
</tr>
</thead>
</table>
| Output file contains only the SQL SELECT statement. | • The file type is SEQ, rather than SQL. If the file type is SEQ, a retrieve is done, but the local file is just sent to the server. The query is not submitted to the DB2 subsystem.  
• The SELECT is for a VIEW for which the user ID does not have DB2 select privilege. The DB2 subsystem returns an empty table. |
| Connection terminated. | The processing time needed by DB2 or FTP or both for the SQL query has exceeded the server time limit for send or receive.  
If you are using the MVS FTP server and the server trace shows a select error due to a bad file descriptor, check the inactive time set for the server and, if necessary, increase the time.  
An FTP client trace indicates the amount of SQL activity through FTP and the approximate time when each query is processed. |

**Restarting file transfers**

A valid restart of an interrupted file transfer depends on reestablishing the environment that existed at the time the file transfer failed. Environment includes:

- The current FTP.DATA statements
- The current SITE and LOCSITE settings
- The sequence of commands (such as Type, Mode, and Structure) that affect the way FTP transfers files
- The current translation tables in use on the data connection

**Restriction:** All environment settings must be re-created before attempting to restart a file transfer.

The following sections describe some possible problems that you might encounter.

**Client rejects the RESTART subcommand**

Use the following checklist if the client rejects the RESTART subcommand:

- • Verify that you have re-created the original file transfer environment.
- • Verify that your environment met all the restrictions for the RESTART subcommand.
- • Verify that checkpointing was active during the failed file transfer.
- • Refer to [z/OS Communications Server: IP User’s Guide and Commands](#) for information about RESTART subcommand restrictions and checkpointing a file transfer.
Client rejects SRESTART subcommand

Use the following checklist if the client rejects the SRESTART subcommand:

- Verify that you have re-created the original file transfer environment.
- Verify that the environment met the SRESTART subcommand restrictions.
- Refer to the z/OS Communications Server: IP User’s Guide and Commands for information about SRESTART subcommand restrictions. Unlike the restart subcommand, you do not need to activate checkpointing, but you do need to enter the SRESTART parameters correctly.

Client accepts SRESTART subcommand, but server rejects RESTART

Use the following checklist if the client accepts SRESTART subcommand, but server rejects RESTART:

- Verify that the server supports stream mode restarts by issuing a FEAT command to the server. The FEAT reply I includes the keyword REST_STREAM if the server supports stream mode restarts.
- Some FTP servers other than z/OS FTP servers reply to the FEAT command with REST_STREAM when they support stream mode restarts in one direction only, such as server to client file transfers. Contact the provider of the FTP server software to verify the server support stream restarts for the direction of the transfer you are attempting.
- Refer to z/OS Communications Server: IP User’s Guide and Commands for information about the feature subcommand.
- Verify that the server has re-created the environment extant during the failed file transfer.
- Did you restart a retrieve and SBSENDEOL is configured to a value other than CRLF at the server, and the server is a z/OS FTP server? If so, you cannot restart the file transfer. Retrieve the file from the server again.

Diagnosing FTP connection and transfer failures with EZA2589E

EZA2589E is issued to describe a timeout or interruption while the FTP client was processing. The following example shows the message format:

EZA2589E Connection to server interrupted or timed out. operation

The message indicates the operation that was in progress when the FTP interruption occurred. Each operation is listed below along with the timer being used and the suggested response. Timers can be set individually in FTP.DATA, or all the timers can be set to one value using the (TI xx or -t xx option when starting the FTP client. Refer to the z/OS Communications Server: IP User’s Guide and Commands for more information regarding the timers.

If the message was generated due to a user interruption, such as using Ctrl-C, ensure the FTP client had enough time to complete before being interrupted. In some cases, a packet trace or a CTRACE might be required to determine why the message was issued. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47 or INFOAPAR III2014 for instructions for taking packet traces and CTRACES.

For more information about EZA2589E, refer to z/OS Communications Server: IP Messages Volume 1 (EZA).
Values and explanations for operation in EZA2589E

This section lists and describes the values for operation in EZA2589E.

**Initial Connection**

**Timer**  MYOPENTIME

**Explanation**
The FTP client is trying to establish a connection with the FTP server. Either the TCP connection has not completed yet or the initial reply from the server has not been received.

**User Response**
Ensure the remote server responds to a ping request. The value of MYOPENTIME can be increased to allow more time for the server to send the initial reply. If the problem recurs, contact the system programmer.

**System Programmer Response**
If there are firewalls between the FTP client and FTP server, ensure the firewalls are allowing FTP traffic from the client IP address to the FTP server for the port being used. A packet trace of the failing transfer shows whether the TCP connection has been completed, the IP addresses being used, and any replies sent by the server.

**Initial IPv6 connection**

**Timer**  MYOPENTIME

**Explanation**
The FTP client is trying to establish a connection with an FTP server using an IPv6 address. Either the TCP connection has not completed yet, or the initial reply from the server has not been received.

**User Response**
Ensure the remote server responds to a ping request. The value of MYOPENTIME can be increased to allow more time for the server to send the initial reply. If the problem recurs, contact the system programmer.

**System Programmer Response**
If there are firewalls between the FTP client and FTP server, ensure the firewalls are allowing IPv6 FTP traffic from the client to the FTP server for the port being used. A packet trace of the failing transfer shows if the TCP connection has been completed, the IP addresses being used, and any replies sent by the server.

**Waiting for data connection**

**Timer**  INACTTIME

**Explanation**
The FTP client is waiting for the FTP server to establish a data connection. A PORT or EPRT command, shown in a previous EZA1701I message, has been sent to the FTP server indicating the IP address and port on which the client is listening. The server should initiate a TCP connection to the FTP client. This connection has not completed yet.

**User Response**
Increase the value of INACTTIME and retry. Contact the system programmer if the failure recurs.

**System Programmer Response**
Ensure that active data connections or PORT or EPRT commands are
allowed by any firewalls between the client and server. Take a packet trace of the failure to determine if the remote FTP server has attempted the connection to the FTP client. If the packet trace does not show an SYN packet arriving from the server to the specified IP address and port, investigate the FTP server and the path to the FTP client to determine if the connection is being blocked. If the FTP client is not responding to the SYN packet, take a CTRACE (with options TCP and INTERNET) and a packet trace. Send these to IBM customer service. The FTP client could also be configured to use firewall friendly data connections by issuing the 
locsite fwfriendly subcommand before the get or put subcommand or by coding FWFRIENDLY TRUE in FTP.DATA. This might allow the data connection to complete because it causes the client to send a PASV or EPSV command instead of a PORT or EPRT command.

**Guideline:** The PORT or EPRT command sent to the server determines the port and IP address the FTP server connects to. For EPRT, the format is EPRT |X|Y|Z|, where X is the address family, Y is the IPv4 or IPv6 address and Z is the port. For the PORT command, the port being used must be calculated. For PORT, the format is a,b,c,d,x,y, where a.b.c.d is the IPv4 address, and (x * 256) + y is the port number.

**Sending a command**

**Timer** INACTTIME

**Explanation**

The FTP client has timed out sending a command to the FTP server. This indicates that the TCP layer is unable to transmit data to the remote server.

**User Response**

Contact the system programmer.

**System Programmer Response**

Take a packet trace to investigate the TCP traffic between the two hosts.

**Sending ABORT command**

**Timer** INACTTIME

**Explanation**

The FTP client has timed out sending a ABORT command to the FTP server. This indicates that the TCP layer is unable to transmit data to the remote server.

**User Response**

Contact the system programmer.

**System Programmer Response**

Take a packet trace to investigate the TCP traffic between the two hosts.

**Receiving data**

**Timer** DATACTTIME

**Explanation**

The FTP client is waiting for data from the FTP server on the data connection. A full buffer of data has not arrived within the DATACTTIME seconds, or the FTP client was interrupted by the user before a full buffer of data arrived. The FTP client issues a recv() call, which returns only when its buffer is full or when the connection has ended. The FTP client uses a default buffer size of 180K. The FTP client is dependent on the data connection.
connection closing cleanly. This informs the FTP client that all the data has arrived from the server. If the connection does not close cleanly, this message is issued.

**User Response**
Increase the DATACTTIME to allow more time for data to arrive. If the failure recurs, contact the system programmer.

**System Programmer Response**
Take a packet trace to investigate the data transfer. The packet trace should be analyzed for conditions which would slow down the transfer, such as retransmitted packets or decreasing window sizes. Increasing the DATACTTIME can allow the FTP client more time to recover from these types of network issues. DATACTTIME should also be increased for transfers over low bandwidth connections, such as dialup. If the packet trace shows that the connection does not close cleanly (for example, the FIN packet is not properly acknowledged), the remote server might need to be investigated as well.

**Tip:** For best results, specify the Session option when formatting the packet trace.

### Sending data

**Timer** DATACTTIME

**Explanation**
The FTP client has timed out sending data to the FTP server over the data connection. The FTP client sets a timer to the value of DATACTTIME seconds before issuing a send call. If the send does not complete in that time period or the FTP client is interrupted by the user, the FTP transfer fails. This timeout can be caused by a slowdown in the transfer, such as network congestion or the remote machine not accepting data.

**User Response**
Increase the value of DATACTTIME to allow more time for the data transmission to occur. If the failure recurs, contact the system programmer.

**System Programmer Response**
Take a packet trace to investigate the data transfer. Analyze the trace for causes of a slowdown. For slow networks, such as dialup, increase the DATACTTIME. If the packet trace shows many retransmitted packets, investigate the network to determine why packets are being dropped.

The window size advertised by the FTP server can also slow down the connection. If the FTP server is advertising a small window size, investigate the server to determine whether the window size can be increased. If the FTP server is very busy, causing the window size to decrease or even go to 0, increase the DATACTTIME to allow more time for the server to handle the data.

**Tip:** Specify the Session option when formatting the packet trace for best results.

### Waiting for reply

**Timer** INACTTIME

**Explanation**
The FTP client is waiting for an expected reply from the FTP server on the control connection. The timer has expired, or the user has interrupted the

---

Chapter 14. Diagnosing File Transfer Protocol (FTP) problems
FTP client before a reply was received. The reply from the FTP server tells the FTP client whether the previous command was successful or not. When a reply is not received, the FTP client must assume that the command was not successful.

**User Response**
INACTTIME could be increased to allow the FTP server more time to reply. If the failure recurs, contact the system programmer.

**System Programmer Response**
For long running jobs, firewalls might time out the control connection due to inactivity. FTPKEEPALIVE can be coded in FTP.DATA to cause the TCP layer to send KeepAlive packets on the control connection. The firewalls can also be configured with longer inactive times. Use a packet trace to determine if the replies arrive at the FTP client. If the packet trace does not show the FTP reply, determine where the reply is being rejected. Otherwise, contact the IBM Support Center to investigate the packet trace.

---

**Sending command to SOCKS server**

**Timer** INACTTIME  

**Explanation**
The FTP client has timed out sending a command to the SOCKS server. This indicates that the TCP layer is unable to transmit data to the SOCKS server.

**User Response**
Contact the system programmer.

**System Programmer Response**
Take a packet trace to investigate the TCP traffic between the two hosts. Use the **locstat** subcommand to determine the IP address of the SOCKS server.

---

**Waiting for reply from SOCKS server**

**Timer** INACTTIME  

**Explanation**
The client is trying to establish a control or data connection to the FTP server through the SOCKS server. The client has sent a connection establishment SOCKS command to the SOCKS server and is waiting for a reply. The FTP client has timed out or been interrupted while waiting for the reply. The SOCKS server might not have replied because it was not processing SOCKS commands in a timely fashion; it was waiting for the remote FTP server to respond, or the SOCKS server did not process the FTP server response in a timely fashion.

**User Response**
INACTTIME can be increased to allow the SOCKS server more time to process commands. If the message occurred while trying to build a data connection through the SOCKS server, issuing the **locsite fwfriendly** subcommand prior to the **put** or **get** subcommand might allow the data connection to be built. If the failure recurs, contact the system programmer.

**System Programmer Response**
Verify with the administrator of the SOCKS server that the server is receiving the commands and processing them in a timely fashion. The IP address of the SOCKS server can be determined with a **locstat** command. Ensure that the SOCKS server can communicate with the FTP server.
Firewalls between the SOCKS server and the FTP server must allow FTP connections and FTP data connections. Take a packet trace to trace the network traffic between the FTP client and SOCKS server. An FTP client trace, enabled by coding DEBUG SOC(2) and DUMP 85 in FTP.DATA, shows the SOCKS commands sent to the server. If a firewall is blocking the data connection, issuing the `locsite fwnfriendly` subcommand prior to the `put` or `get` subcommand or specifying FWFRIENDLY TRUE in FTP.DATA might allow the data connection to complete.

**Establishing data connection through SOCKS server**

**Timer**  MYOPENTIME

**Explanation**
The FTP client is trying to establish a TCP connection to the SOCKS server so that a data connection can be established to the FTP server. The client has already successfully logged into the FTP server using the SOCKS server. The TCP connection has not completed. The SOCKS server might be too busy to accept new connections in a timely fashion.

**User Response**
The value of MYOPENTIME can be increased to allow more time for the SOCKS server to accept the connection. If the failure recurs, contact the system programmer.

**System Programmer Response**
Contact the administrator of the SOCKS server to determine if the SOCKS server is accepting new connections. Take a packet trace to verify that the SOCKS server is not responding to the connection attempt.

**Initial connection to SOCKS server**

**Timer**  MYOPENTIME

**Explanation**
The FTP client is trying to establish a TCP connection to the SOCKS server so that a control connection can be established with the FTP server. The TCP connection has not completed. The SOCKS server might be too busy to accept new connections in a timely fashion.

**User Response**
Use the `loconfig` subcommand to determine the IP address of the SOCKS server. Verify that the SOCKS server is reachable by pinging the server. Increasing the value of MYOPENTIME allows the SOCKS server more time to accept the connection. If the problem recurs, contact the system programmer.

**System Programmer Response**
Verify that the SOCKS server is reachable. Contact the administrator of the SOCKS server to determine if the SOCKS server is accepting new connections. Take a packet trace to determine if the TCP connection to the SOCKS server completes. Use the `loconfig` subcommand to determine the IP address of the SOCKS server; the port number of the SOCKS server is always 1080.

**Problems starting the client**
This section lists and describes possible problems starting the FTP client.
Enabling or suppressing message EZYFT47I during startup

When the FTP client reads a statement in FTP.DATA that is supported by the z/OS FTP server but not by the FTP client, it issues the message EZYFT47I as a warning. For example, if the client finds an ANONYMOUS statement in FTP.DATA, it issues EZYFT47I for that statement because the ANONYMOUS statement has meaning only for the z/OS FTP server.

If you use the same FTP.DATA configuration file for both client and server, you might want to suppress the EZYFT47I messages. You can prevent the client from issuing this warning by coding a SUPPRESSIGNOREWARNINGS statement in FTP.DATA. Code SUPPRESSIGNOREWARNINGS in FTP.DATA only after you verify all statements in FTP.DATA are correct.

If you require message EZYFT47I for diagnostic purposes, verify no SUPPRESSIGNOREWARNINGS statements are coded in FTP.DATA, or else code SUPPRESSIGNOREWARNINGS FALSE in FTP.DATA ahead of those statements you want to debug.

Abends

If the client abends immediately after entering the FTP command and the following message is displayed, ensure that the local TSO user ID has an OMVS segment defined or that a default OMVS segment is established:

```
ftp
CEES101C During initialization, the OpenEdition callable service
BPXIMS failed. The system return code was 0000000156
, the reason code was 0B0C00FB . The application will be
terminated
IKJ566411 FTP ENDED DUE TO ERROR+
READY
```

Incorrect configuration values

Issue the LOCSTAT subcommand to determine the name of the file being used for your local site configuration parameters. If the file you want is not being used, start the FTP client with the `-d` or `TRACE` options to trace the client as it follows the search order for the FTP.DATA file. For more information about the search order used by the client, refer to `z/OS Communications Server: IP User's Guide and Commands`.

Determine whether your FTP.DATA file has sequence numbers. If it does, any statement with an optional parameter omitted picks up the sequence number as the parameter value. For example, the BLKSIZE statement has an optional parameter `size`. If you specify the size, the sequence number is ignored. If you do not specify the size, the system assumes the sequence number is the size, causing an error.

Problems logging into the server

This section lists and describes possible problems logging into the server.

Client ignores SOCKS configuration file

If you suspect that the client consistently ignores the SOCKS configuration file, use the `locstat` subcommand to display the name of the SOCKS configuration file.

- If no SOCKS configuration file name appears in the LOCSTAT output, the client is not configured correctly. Verify that a SOCKSCONFIGFILE statement is in FTP.DATA.
Inspect the client syslog output for error messages relating to SOCKSCONFIGFILE in FTP.DATA. Use the client DEBUG INT statement to trace client initialization, and look for messages relating to the SOCKS configuration.

The FTP client references the SOCKSCONFIGFILE only when it is connecting to servers with IPv4 IP addresses; it is supposed to ignore the SOCKSCONFIGFILE when logging in to an FTP server with an IPv6 IP address. If you specify the FTP server by DNS name, that name might resolve to an IPv6 address rather than to an IPv4 address. Use the LOCSTAT subcommand to display the IP address used to log in to the server; the port number of the SOCKS server is always 1080.

**Client connects to wrong SOCKS server**

If the client connects to a wrong SOCKS server, to a SOCKS server when it should not, or ignores SOCKS configuration file some of the time, use the locstat subcommand to display the name of the SOCKS configuration file.

- If the name displayed is not correct, correct the SOCKSCONFIGFILE statement in FTP.DATA.
- If the SOCKS configuration file name displayed by LOCSTAT is correct, inspect the contents of the SOCKS configuration file.

The client processes the statements in the order they are coded and applies the first statement that specifies the target FTP server. Check and arrange the statements as appropriate, or add a new statement specific to the FTP server at the beginning of the file.

**Connection through SOCKS server to FTP server fails**

A SOCKS connection involves a connection between the client and SOCKS server, and the SOCKS server and the target server.

When a connection fails, try to isolate the point of failure by checking the following:

- Can client connect to the SOCKS server host?
  Use the client SOC(2) trace and the DUMP 85 trace during connection establishment, and inspect any messages to gain insight into whether the client was able to connect to the SOCKS server.

- Is the link between the client and the SOCKS server good?
  Use ping to test the link.

- Is the SOCKS server active?

- Is the SOCKS server configured to reject the connection?
  Contact the administrator of the SOCKS server for assistance.

- Is the link between the SOCKS server and the FTP server good?
  Ask the administrators of the SOCKS server and the FTP server to verify the link.

- Is the FTP server active and accepting connections?
  Contact the administrator of the FTP server. For the z/OS FTP server, activate the trace and check the syslog to determine whether the FTP server received a connection from the SOCKS server on behalf of the client.

**Message EZA2589E appears while trying to log in**

See “Diagnosing FTP connection and transfer failures with EZA2589E” on page 474.
Server rejects password
The z/OS FTP server supports case-sensitive passwords when your RACF administrator has enabled mixed-case passwords. Verify that you have entered the password correctly, and in the correct case.

If you are using a NETRC data set to provide the FTP login password, verify that the password is coded correctly and in the correct case.

If the z/OS FTP server rejects a mixed-case or lower-case password that it formerly accepted, it is possible your RACF administrator has disabled RACF mixed-case password support. In that case, it is not possible to login with any ID whose password has been set to mixed or lower case. Ask your RACF administrator to reset the password.

Unknown host error message
The FTP client displays EZA1551I Unknown Host: <hostname> if it receives a negative response from the resolver. This occurs when the hostname specified on the FTP command cannot be resolved either by the name server or the local resolution file.

Rule: The FTP client always uses the z/OS UNIX search order for TCPIP.DATA, even when FTP is invoked from TSO.

Use the host IP address instead of the hostname on the FTP command, or see Chapter 39, “Diagnosing resolver problems,” on page 853 for information about diagnosing name server problems.

Problems transferring data
This section lists and describes possible problems transferring data.

Many data transfer problems that apply to a server apply also to a client. See “Cannot establish conversion between <codeset> and UCS-2” for more information.

Cannot establish conversion between <codeset> and UCS-2
If you invoke the FTP client under TSO, and issue a TYPE U2 or UCS2 subcommand, the following message might be issued:

EZA2749E Cannot establish conversion between <codeset>
and UCS-2.

To transfer data encoded in UCS-2 during an FTP session, invoke the FTP command with the _ICONV_UCS2_PREFIX environment variable, specifying the prefix used for your runtime library. Following is an example:

FTP ENVAR("_ICONV_UCS2_PREFIX=CEE.OSVIR4") / <host_ip_addr> <port>

Secure IPv4 FTP session cannot transfer data through an NAT firewall
If you are using an encrypted FTP control connection, as is the case when using AT-TLS security, and the client sends PASV or PORT to establish a data connection for file transfer, and a NAT (network address translation) firewall exists between the client and server, you might find that while you could sign into the server, you cannot establish the data connection for the transfer. This is because a NAT firewall monitors the FTP control connection as well as the IP headers, changing IP
addresses as needed. If the control connection is encrypted, the NAT cannot monitor and change the IP addresses exchanged between the FTP client and server by PASV and PORT.

Use the `locsite` subcommand with the EPSV4 parameter, or code EPSV4 TRUE in FTP.DATA, to direct the client to use EPSV instead of PORT or PASV on IPv4 sessions to establish the data connection. The EPSV command exchanges only port numbers between FTP client and server, so the NAT firewall does not need to translate IP addresses. The server must support EPSV on IPv4 sessions for this solution to be effective.

If the server does not support the EPSV command, you can use the PASSIVEIGNOREADDR configuration option to ignore the IP address that is returned on a PASV command reply and use only the port. For more information about the EPSV command, see RFC 2428. For more information about the `locsite` subcommand, see [z/OS Communications Server: IP User's Guide and Commands](https://www.ibm.com/support/docview.ws?rs=1806&uid=swg27017811). For more information about the EPSV4 or PASSEIVEIGNOREADDR statement in FTP.DATA, refer to [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com/support/docview.ws?rs=1806&uid=swg27017811).

### Firewall does not permit FTP client to establish a data connection

You might be able to log in to an FTP server through a firewall, but find you cannot transfer files using a passive data connection. The reason is that the ephemeral ports chosen for the data connection are outside the range of ports permitted by the firewall.

If the client sends EPSV or PASV to the server to start the data connection, FTP is said to be establishing a passive data connection, or is said to be operating in passive mode. In passive mode, the server chooses the ephemeral port for the data connection. Ephemeral port numbers are part of EPSV and PASV replies the server sends to the client. You can configure the z/OS FTP server to use only a specific range of ephemeral ports for the data connection compatible with what you have configured for your firewall by coding the PASSIVEDATAPORTS statement in FTP.DATA. Refer to [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com/support/docview.ws?rs=1806&uid=swg27017811) for information about the PASSIVEDATAPORTS statement.

If the client sends PORT or EPRT to the server to start the data connection, the client is said to be establishing an active data connection, or operating in active mode. Active mode FTP is not recommended for sessions through firewalls. Use the `locsite` subcommand with the FWFRIENDLY parameter, or code FWFRIENDLY TRUE in FTP.DATA, to direct the client to operate in passive mode.

### Server rejects PORT or EPRT command with 504 replies

Data transfer command sequences that use the PORT or EPRT command fails when the server that receives the PORT or EPRT command is configured to reject all or certain PORT and EPRT commands. The reply code 504 indicates a problem of this nature.

For an ordinary transfer of data between client and server, the z/OS FTP client sends the PORT command to server when:

- The server does not support the EPSV command or the FTP session protocol is IPv4, and
- The client is not configured to be firewall-friendly

You can correct this problem in one of these ways.
• Make the client firewall-friendly. Do this for the z/OS FTP client by coding FWFRIENDLY TRUE in the client’s FTP.DATA or by using a LOCSITE FWFRIENDLY subcommand before attempting the data transfer. The client sends EPSV or PASV to the server instead of PORT and the problem is avoided.
• Log in to the server using the server IPv6 address. The client uses EPSV instead of PORT and the problem is avoided.

  Restriction: The server must have an IPv6 address.

• Change the server configuration so that it does not reject PORT or EPRT commands.
• Change the server so that it supports the EPSV command. The z/OS FTP server supports the EPSV command.

To change the client, see the z/OS Communications Server: IP User’s Guide and Commands for information about the FWFRIENDLY statement and the LOCSITE subcommand.

If you used the proxy subcommand to start the transfer, you are transferring data between two servers instead of between client and server. For a transfer of data between two servers, the client must send PORT or EPRT to one of the servers, and PASV or EPSV to the other server. If the server receiving the PORT or EPRT command is configured to reject the PORT or EPRT command, the proxy transfer fails with a 504 reply.

You can fix this problem in one of the following ways.
• Reverse the order in which you open the server connections. That is, if you opened a connection to ServerA and proxy opened a connection to ServerB, open the connection to ServerB and proxy open the connection to ServerA. The client then sends PORT or EPRT to the other server during the proxy transfer. Provided the other server does not also reject PORT or EPRT, this avoids the problem.

  Restriction: If the file you are transferring is a load module, changing the order in which you open server connections does not always cause the client to send PORT or EPRT to the other server.
• Transfer the file to a client, and then to the other server.
• Change the server so that it does not reject PORT and EPRT commands.

The following are z/OS server FTP.DATA statements that can be coded to reject PORT and EPRT commands:

PORTCOMMAND
  Reject all PORT and EPRT commands.

PORTCOMMANDPORT
  Reject PORT and EPRT commands whose port number argument is a well-known port number.

PORTCOMMANDIPADDR
  Reject PORT and EPRT commands whose argument is an IP address that is different from the client’s IP address.

Refer to z/OS Communications Server: IP Configuration Guide for more detail.

Message EZA2589E appears when trying to transfer data

See “Diagnosing FTP connection and transfer failures with EZA2589E” on page 474.
Other problems

This section lists and describes other problems diagnosing FTP connection and transfer failures.

Client PDS member statistics not created or updated

ISPFStats must be set to TRUE in order to create or update the statistics for the PDS Member when using GET and MGET subcommands. When the PDS directory block is full, PDS member statistics are not updated. Use the locstat subcommand to verify that the client’s ISPFStats setting is TRUE. Use the LOCSITE ISPFStats subcommand change the ISPFStats value. Refer to [z/OS Communications Server: IP User’s Guide and Commands](#) for information about using the LOCSITE subcommand.

**Diagnosing FTP client problems with tracing**

You can activate tracing on startup by doing the following:

- Coding DEBUG statements in FTP.DATA. Refer to the DEBUG statement in [z/OS Communications Server: IP Configuration Reference](#) for more information.
- Starting the FTP client with the -d command-line option. Refer to [z/OS Communications Server: IP User’s Guide and Commands](#) for more information about the FTP environment.

Alternatively, you can activate tracing by toggling tracing on or off during an FTP session with the DEBUG command.

The DEBUG and DUMP subcommands activate the general and the extended levels of tracing. The general tracing shows key events in the processing of a subcommand (for example, the opening of a file) and the extended trace shows data areas that are used during processing. The extended trace produces large amounts of output and should be used at the direction of IBM service team. The format of DEBUG allows multiple parameters to be specified on one subcommand. Refer to [z/OS Communications Server: IP User’s Guide and Commands](#) for the syntax and parameters for the DEBUG and DUMP subcommands.

For example, the following sequence of subcommands would set traces:

```
DEBUG ACC SQL  *Activates the ACC and SQL traces
DEBUG BAS      *Activates the default traces
               *CMD, INT, FSC, and SOC in addition
               *to the two already set
DEBUG          *Resets all tracing
```

When running FTP interactively or from a REXX exec, all tracing goes to the terminal unless output is redirected. When running FTP from a TSO batch job, all tracing goes to SYSOUT.

Use the following checklist to diagnosis FTP client problems with tracing:

- Ensure that the user has properly allocated the DDNAME being referred to. The TSO command LISTALC STAT HIST can be helpful in debugging allocations. Also, ensure that the allocations are correct. For example, if a file already exists, the disposition should not be new.
- Ensure that DDNAMEs are only used to refer to local files. For example, get //DD:FTP01 FILEONE is not valid because it attempts to use a DDNAME to refer to a host file. If you try to use a DDNAME for a remote file name, the name is sent to the remote host for processing as it is. If the remote host actually has a file named //DD:FTP01, then that file would be referred to, but most likely the remote host would reject it as a file name that is not valid.
To find attempts to access files by DDNAME, look for DD: in FTP trace output as shown below:

MF0573 seq_open_file: OSTN -> w, recfm=*, NOSEEK for dd: FTP02
MF0663 seq_open_file: ddname FTP02 has filename USER1.CCPYXLMT
MF0669 seq_open_file: set DDNAME characteristics- recfm=90, lrecl=128, blksize=6144

Tip: By using DDNAME support, the user is assuming responsibility for correctly allocating and deallocating the DDNAMEs being used.

Where to find the FTP client trace
The destination of the z/OS FTP client trace depends on the environment in which the client executes as described as follows:

• When the FTP client is invoked interactively from TSO or a REXX exec with an allocated OUTPUT DD, the trace is written to the destination associated with the OUTPUT DD.
• When the FTP client is invoked interactively from a TSO session with no allocated OUTPUT DD, the trace is written to the user’s console.
• When the FTP client is invoked interactively from OMVS, the trace is written to the user’s console, or it can be written to a file by using the OMVS redirect operand (>).
• When the FTP client is invoked interactively from a REXX exec with no allocated OUTPUT DD, the trace is written to the destination for STDOUT (which might be the user’s console).
• When the FTP client is invoked from any application using the FTP Callable Application Programming Interface (API), the trace output is stored in the interface buffer until the application issues a request to retrieve the output. Refer to z/OS Communications Server: IP Programmer’s Guide and Reference for a complete description of the FTP Callable API.

Rules: When the FTP client is invoked from a batch job, the following rules apply:
– If the client is invoked directly (EXEC PGM=FTP), the trace is written to the destination associated with the OUTPUT DD.
– If the client is invoked from TSO in batch (EXEC PGM=IKJEFT01), the trace is written to the destination associated with the OUTPUT DD if one exists. Otherwise, the trace is written to the destination associated with the SYSTSPRT DD.
– If the client is invoked from a REXX exec in batch, whether or not under batch TSO, the trace is written to the destination for the OUTPUT DD (if one exists). Otherwise, the trace is written to the destination for STDOUT (under batch TSO, this might be the SYSTSPRT DD).

Documenting FTP client problems
If the problem is not caused by any of the common errors described in this section, collect the following documentation before calling the IBM Software Support Center. Documentation is divided into the following categories: essential and helpful (but not essential).

• Essential
  – Precise problem description, including client console, expected results, and actual results
  – Include trace in the output by one of these methods:
- Coding DEBUG statements in the client’s FTP.DATA. See z/OS Communications Server: IP Configuration Reference for information about the DEBUG statement.

- Invoking the FTP client with the -d or TRACE invocation option. See z/OS Communications Server: IP User’s Guide and Commands for information about entering the FTP environment.

- Specifying a DEBUG subcommand in the client command input stream before the affected transfer. Use this option only if the problem does not involve the initial establishment of the FTP control session.

- FTP.DATA file used by the client.

- You can use DEBUG ALL to capture all details possible.

- When activating the trace, use the DEBUG option TIMESTAMPS to time stamp the client trace output. See z/OS Communications Server: IP User’s Guide and Commands for information on the DEBUG subcommand and z/OS Communications Server: IP Configuration Reference for information on the DEBUG statement.

- If executing the client in batch, collect all the JES output.

• Helpful:
  - Output from the client locstat subcommand
  - Output from the client stat subcommand
  - FTP.DATA data set
  - TCPIP.DATA data set
  - If appropriate, sample data to re-create the problem
  - If the FTP.DATA parameter LOGCLIENTERR is TRUE, report the contents of message EZZ9830I. The message is written to the system log and the job log when the client is running in batch and to the user’s terminal during an interactive client session.
Chapter 15. Diagnosing z/OS UNIX Telnet daemon (otelnetd) problems

This topic provides diagnostic information for z/OS UNIX Telnet daemon (otelnetd) and contains the following sections:

- "Common problems"
- "Debug traces" on page 490

Common problems

The following list describes common problems that you might encounter during execution of the Telnet daemon (otelnetd).

- Diagnostic messages are not being printed to the appropriate file.
  - The diagnostic messages are printed out with the use of syslogd. Ensure that the syslogd is currently active by checking for /etc/syslog.pid.
  - If syslogd is active, ensure that the file where the output is sent is currently allocated. Syslogd creates the file if it is started with the -c runtime option. z/OS UNIX Telnet uses local1.debug for logging messages. Ensure that the syslog.conf file contains an entry for local1.debug or the *.* default file. Refer to the z/OS Communications Server: IP Configuration Guide for more detailed information about syslogd.
  - Ensure also that the specified file exists. Ensure that the permissions on the file are at a minimum 666.
  - Make sure you specify -t or -D all, or -t and -D all, as the z/OS UNIX Telnet options in /etc/inetd.conf.

- Use of the arrow keys.
  The arrow keys are not functional in raw mode. This is AIX-like behavior, except that, in AIX, the arrow key produces peculiar characters such as ~B on the screen to let the user know not to use arrows. Under rlogin, the cursor moves to where you would want it and correction is allowed, but the shell also treats these characters as part of the original command.

- The keyboard appears to be locked and the user cannot issue commands.
  When executing UNIX-type clients (for example, AIX), if the -k option is specified for Telnet in inetd.conf, Telnet does not allow kludge linemode (see “Setting up the inetd configuration file” on page 595). UNIX clients require character-at-a-time mode to process correctly. If you remove the -k option from the parameters, then the software processes correctly.
  If this does not work, run tracing -t d all. Look for Ept to determine what the exception conditions are for the pty. The number of bytes should equal four.
  Verify that the exception conditions identified are processed by the TN3270E Telnet server. (Check EZYTE67I messages for more information; see Figure 60 on page 491)

- EDC5157I An internal error has occurred, rsn=0b8802AF.
  The 2AF of the reason code signifies that the user did not have the proper authority to execute the command. This might result in either the user system having BPX.DAEMON authority set up in its environment, and the proper authorities have not been issued to the user. Another result might be that the user does not have super user authority, which might be required to issue some of these commands.
Debug traces

Table 30 describes options that relate to user-controlled trace information.

<table>
<thead>
<tr>
<th>Option</th>
<th>Sub-Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-t</td>
<td>Internal tracing, intended to replace the DIAGNOSTICS compile option currently in place within the BSD code.</td>
<td></td>
</tr>
<tr>
<td>-D</td>
<td>authentication</td>
<td>Turns on authentication debugging code.</td>
</tr>
<tr>
<td>-D</td>
<td>encryption</td>
<td>Turns on encryption debugging code.</td>
</tr>
<tr>
<td>-D</td>
<td>options</td>
<td>Prints information about the negotiation of TELNET options.</td>
</tr>
<tr>
<td>-D</td>
<td>report</td>
<td>Prints the options information, plus some additional information about what processing is going on.</td>
</tr>
<tr>
<td>-D</td>
<td>netdata</td>
<td>Displays the data stream received by telnetd.</td>
</tr>
<tr>
<td>-D</td>
<td>ptydata</td>
<td>Displays the data stream written to the pty.</td>
</tr>
<tr>
<td>-D</td>
<td>all</td>
<td>Supports all options/report/ptydata/netdata/authentication/encryption options.</td>
</tr>
</tbody>
</table>

Debug trace flows (netdata and ptydata)

When issuing any of the following three trace commands within /etc/inetd.conf (-D ptydata, -D netdata, or -D all), you have the contents in both hexadecimal and ASCII, and the data being sent over the sockets or between the ttys in your syslogd file. If the user is having problems between the parent and the client, try the -D netdata option. If it is between the parent and the child, try the -D ptydata option. If both or either might apply, try the -D all option.

Each set of hexadecimal data is preceded by a three-letter tag. This tag represents the direction the data is flowing from. Figure 59 is a pictorial representation of this flow:

- Int—client to parent
- Ont—parent to client
- Ipt—child to parent
- Opt—parent to child

![Figure 59. Trace between the Telnet client, parent, and child](image)

The user types a command on the command line. It flows Int -> Opt. The child responds and the flow is Ipt -> Ont.

Debug trace examples (-t -D all)

Figure 60 on page 491 gives an example of the trace generated from -t -D all, generated from an AIX Telnet client. A trace explanation follows the figure.
EZYTE29I Starting new telnet session. catfd = 168443936
EZYTE05I Initial EBCDIC codepage = IBM-1047, ascii codepage = ISO8859-1
EZYTE05I Trace 1 Debug 3 keepalive 1 kludgelinemode 0
telnetd: doit(Second_pass=0)
EZYTE11I doit: host_name laph.raleigh.ibm.com
EZYTE11I doit: IP address 9.37.83.93
EZYTE11I doit: PORT 2504
EZYTE11I doit: host MVSJ
>>>TELNETD: I support auth type 2 6
>>>TELNETD: I support auth type 2 2
>>>TELNETD: I support auth type 2 0
>>>TELNETD: I will support DES_CFB64
>>>TELNETD: I will support DES_OFB64
telnetd: getterminaltype() auth_level=0
state: send_do(option=37, init=1)
EZYST04I STATE:send_do: send DO AUTHENTICATION
EZYTU14I UTILITY: netwrite 3 chars.
EZYTU21I Ont: fffd25 ...
EZYTU03I UTILITY: ttloop read 33 chars.
EZYTU20I Int: fffb25fffd26fffb26fffd03fffb18fffb1ffffb ................
EZYTU20I Int: 20fffb21fffb27fffd05 ............
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYST05I STATE:willoption: receive WILL AUTHENTICATION
>>>TELNETD: Sending type 2 6
>>>TELNETD: Sending type 2 2
>>>TELNETD: Sending type 2 0
utility: printsbuf(length=10)
EZYTU17I UTILITY: send suboption AUTHENTICATION
SEND
KERBEROS_V5
CLIENT|MUTUAL|ENCRYPT
KERBEROS_V5
CLIENT|MUTUAL
KERBEROS_V5
CLIENT|ONE-WAY
EZYST01I STATE:dooption: receive DO ENCRYPT
EZYST09I STATE:send will: send WILL ENCRYPT
EZYST05I STATE:willoption: receive WILL ENCRYPT
state: send_do(option=38, init=0)
EZYST04I STATE:send_do: send DO ENCRYPT
EZYTU17I UTILITY: send suboption ENCRYPT
SEND
ENCRIPT
SUPORT
DES_CFB64
DES_OFB64
EZYST01I STATE:dooption: receive DO SUPPRESS GO AHEAD
EZYST09I STATE:send will: send WILL SUPPRESS GO AHEAD
EZYST05I STATE:willoption: receive WILL TERMINAL TYPE
state: send_do(option=24, init=0)
EZYST04I STATE:send_do: send DO TERMINAL TYPE
EZYST05I STATE:willoption: receive WILL NAWS
state: send_do(option=31, init=0)
EZYST04I STATE:send_do: send DO NAWS
EZYST05I STATE:willoption: receive WILL TSPEED
state: send_do(option=32, init=0)
EZYST04I STATE:send_do: send DO TSPEED
EZYST05I STATE:willoption: receive WILL LFLOW
state: send_do(option=33, init=0)

Figure 60. z/OS UNIX Telnet trace using -t -D all (Part 1 of 11)
Figure 60. z/OS UNIX Telnet trace using -t -D all (Part 2 of 11)
EZYTU17I UTILITY: receive suboption
AUTHENTICATION
IS
KERBEROS_V5
CLIENT|MUTUAL|ENCRYPT
AUTH 110 130 1 198 48 130 1 194 160 3 2 1 5 161 3 2 1 14 162 7 3 5 0 32 0 0 0
163 130 1 18 97 130 1 14 48 130 1 10 160 3 2 1 5 161 16 27 14 75 82 66 51 57 48
46 73 66 77 46 67 79 77 162 43 48 41 160 3 2 1 3 161 34 48 32 27 4 104 111 115
109 46 99 111 110 109 163 129 195 48 129 192 160 3 2 1 1 161 3 2 1 1 162 129 179 4
129 176 28 188 181 169 95 210 174 19 209 43 213 123 8 209 49 51
164 133 200 164 71 60 88 87 51 222
>>>REPLY:2: [3] (91)
6f 59 30 57 a0 03 02 01 05 a1 03 02 01 0f a2 4b
>>>REPLY:2: [2] (21)
75 73 65 72 35 32 40 4b 52 42 33 39 30 2e 49 42
EZYTU14I UTILITY: netwrite 130 chars.
EZYTU21I Ont: fffa25020206036f593057a003020105a103020105a1030201 raaaron
EZYTU21I Ont: 0fa24b03049a003020101a24044117ac36284cd s........C.d.
EZYTU21I Ont: 65384025a9ee70511777f91aa4367ed20162f .z....j...;
EZYTU21I Ont: 73e4d6e61f0e60d0c74c25aa610ddc010526eea .w..<..K<../...
EZYTU21I Ont: b096de7e7e98a067a595cdeb92369be741fff0 .PHX?..k...X..0
EZYTU21I Ont: fffa25020260275736572324524233930 ............. 
EZYTU21I Ont: 2e4942d2e434f4dff0 ...(...)0
utility: printsub(length=4)
EZYTU17I UTILITY: receive suboption
ENCRIPT
REQUEST-START
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=5)
EZYTU17I UTILITY: receive suboption
ENCRIPT
SUPPORT
DES_OFB64
>>>TELNETD: He is supporting DES_OFB64 (2)
Creating new feed
utility: printsub(length=14)
EZYTU17I UTILITY: send suboption
ENCRIPT
IS
DES_OFB64
OFB64_IV 123 117 204 223 5 21 98 2
>>>TELNETD: (*ep->start)() returned 6
EZYTU17I Defer suboption negotiation
EZYTU14I UTILITY: netwrite 16 chars.
EZYTU21I Ont: fffa260002017b75ccdf0515620f0 .........0
utility: printsub(length=7)
EZYTU17I UTILITY: receive suboption
NAWS
0 120 (120)
0 50 (50)
auth_wait: auth_context a080a30, validuser 3
auth_wait: auth_level 0
telnetd: authteln client name: user52 auth name: user52
state: send do(option=38, init=0)
EZYTUS04I STATE:send do: send DO ENCRYPT
state: send do(option=24, init=1)
state: send do(option=32, init=1)
state: send do(option=35, init=1)
EZYTUS04I STATE:send do: send DO XDISPLOC
state: send do(option=39, init=1)
state: send do(option=36, init=1)
Figure 60. z/OS UNIX Telnet trace using -t -D all (Part 3 of 11)
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTS17I Defer suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=52)
EZYTU17I UTILITY: receive suboption
LINEMODE
SLC
SYNCH
DEFAULT
0;
IP
VARIABLE
|FLUSHIN|FLUSHOUT
3;
AO
VARIABLE
15;
AYT
DEFAULT
0;
ABORT
VARIABLE
|FLUSHIN|FLUSHOUT
28;
EOF
VARIABLE
4;
SUSP
VARIABLE
|FLUSHIN
26;
EC
VARIABLE
8;
EL
VARIABLE
21;
EW
VARIABLE
23;
RP
VARIABLE
18;
LNEXT
VARIABLE
22;
XON

Figure 60. z/OS UNIX Telnet trace using -t -D all (Part 4 of 11)
Figure 60. z/OS UNIX Telnet trace using -t -D all (Part 5 of 11)
Figure 60. z/OS UNIX Telnet trace using -t -D all (Part 6 of 11)
EZYTU17I UTILITY: receive suboption
ENCRIPT
START
>>>TELNETD: Start to decrypt input with type DES_OFB64
EZYS17I Defer suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=13)
EZYTU17I UTILITY: receive suboption
TERMINAL-SPEED
IS 9600,9600
EZYS17I Defer suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=16)
EZYTU17I UTILITY: receive suboption
NEW-ENVIRON
IS
VAR
U
S
E
R
VALUE
u
s
er
5
2
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=9)
EZYTU17I UTILITY: receive suboption
TERMINAL-TYPE
IS XTERM
EZYTE10I terminaltypeok: call tgetent (buf, XTERM)
EZYTE51W terminaltypeok: Tgetent failure EDC5129I No such file or directory.
rsn = 0594003D
EZYTE10I terminaltypeok: call tgetent (buf, xterm)
telnetd: getterminaltype() return 3
EZYTE22I herald() entered for /etc/otelnetd.banner
EZYTE22I herald() entered for /etc/otelnetd.banner
EZYTE88E herald: stat error on /etc/otelnetd.banner
EDC5129I No such file or directory. rsn = 053B006C
EZYTE01I read_pw: Character ignored 0
EZYTE04I lusername = user52
telnetd: krb name: user52, user: user52
EZYTE22I herald()
EZYTE22I herald() entered for /etc/banner

Figure 60. z/OS UNIX Telnet trace using -t -D all (Part 7 of 11)
38 EZYTE88E herald: stat error on /etc/banner
EDC5129I No such file or directory. rsn = 053B006C
EZYTE16I uid = 52, gid = 5
telnsave: mallocTelnetSave() rc=0
telnetd: doit() subcount=96
telnetd: doit() execvp()
EZYTU34I id 30002 pri 3 call catopen(tmsgs.cat,0) code 81 reason 053B006C
h_errno N/A
telnetd: main() -y getsubopt(tSave=2137884232)
telnsave: freeTelnetSave() rc=0
EZYT011I DInt: fffaf80780a2ff0fff88ba20301030003620304 ........0...........
EZYT011I DInt: 020f05030007621c00200404921a0a02080b0215 .................
EZYT011I DInt: 0c02170d02120e02160f02111002131102001202 .................
EZYT011I DInt: 00fff0fa200039630302c39963030fff0fff88ba0...........0..0..
EZYT011I DInt: 2700005653455201757365723532fff0 .................0

telnetd: doit(Second_pass=1)
EZYT011I GETPTY: open of /dev/ptyp EDC5114I Resource busy. rsn = 020A0155
EZYT011I GETPTY: open of /dev/ptyp EDC5114I Resource busy. rsn = 020A0155
EZYT011I GETPTY: slave fd = 9 , masterfd = 8
telnetd: doit() deferred_processing=1
39 EZYTS15I STATE:dooption:deferred receive DO ECHO
EZYTO09I options(1) = 3 ,
EZYTS15I STATE:dooption:deferred receive DO SUPPRESS GO AHEAD
EZYTO09I options(3) = 3 ,
EZYTS15I STATE:dooption:deferred receive DO STATUS
EZYTO09I options(5) = 3 ,
39 EZYTS16I STATE:willoption:deferred receive WILL TERMINAL TYPE
EZYTO09I options(24) = 12 ,
EZYTS16I STATE:willoption:deferred receive WILL NAWS
EZYTO09I options(31) = 12 ,
EZYTS16I STATE:willoption:deferred receive WILL TSPEED
EZYTO09I options(32) = 12 ,
EZYTS16I STATE:willoption:deferred receive WILL LLOW
EZYTO09I options(33) = 12 ,
EZYTS16I STATE:willoption:deferred receive WILL LINEMODE
EZYTU14I UTILITY: netwrite 13 chars.
EZYTU21I Ont: b7c2d637e5d127b4aeced4c4bd .BO.VJ....M.[
EZYTO09I options(34) = 12 ,
EZYTS16I STATE:willoption:deferred receive WILL AUTHENTICATION
EZYTO09I options(37) = 12 ,
EZYTS15I STATE:dooption:deferred receive DO ENCRYPT
EZYTO09I options(38) = 15 ,
EZYTS16I STATE:willoption:deferred receive WILL NEW-ENVIRON
EZYTO09I options(39) = 12 ,
telrcv() encrypt_output=0xA00C848
telrcv() decrypt_input =0xA00C6C0
EZYS18I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=7)
EZYTU17I UTILITY: receive suboption NAWS
0 120 (120)
0 50 (50)
EZYS18I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=52)

Figure 60. z/OS UNIX Telnet trace using -t -D all (Part 8 of 11)
EZYTU17I UTILITY: receive suboption
  LINEMODE
  SLC
  SYNCH
  DEFAULT
  0;
  IP
  VARIABLE
  |FLUSHIN|FLUSHOUT
  3;
  AO
  VARIABLE
  15;
  AYT
  DEFAULT
  0;
  ABORT
  VARIABLE
  |FLUSHIN|FLUSHOUT
  28;
  EOF
  VARIABLE
  4;
  SUSP
  VARIABLE
  |FLUSHIN
  26;
  EC
  VARIABLE
  8;
  EL
  VARIABLE
  21;
  EW
  VARIABLE
  23;
  RP
  VARIABLE
  18;
  LNEXT
  VARIABLE
  22;
  XON
  VARIABLE
  17;
  XOFF
  VARIABLE
  19;
  FORW1
  VARIABLE
  0;
  FORW2
  VARIABLE
  0;
EZYT518I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=13)
EZYTU17I UTILITY: receive suboption
  TERMINAL-SPEED
  IS 9600,9600
EZYT518I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0 chars.
utility: printsub(length=16)

Figure 60. z/OS UNIX Telnet trace using -t -D all (Part 9 of 11)
Figure 60. z/OS UNIX Telnet trace using -t -D all (Part 10 of 11)
Figure 60. z/OS UNIX Telnet trace using -t -D all (Part 11 of 11)

Following are short descriptions of the numbered items in the trace:

<table>
<thead>
<tr>
<th>ACK</th>
<th>FLUSHIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>EC</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td>8</td>
</tr>
<tr>
<td>EL</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td>21</td>
</tr>
<tr>
<td>EW</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td>23</td>
</tr>
<tr>
<td>RP</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td>18</td>
</tr>
<tr>
<td>LNEXT</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td>22</td>
</tr>
<tr>
<td>XON</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td>17</td>
</tr>
<tr>
<td>XOFF</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td>19</td>
</tr>
<tr>
<td>FORW1</td>
<td></td>
</tr>
<tr>
<td>ACK</td>
<td>0</td>
</tr>
<tr>
<td>FORW2</td>
<td>NOSUPPORT</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

EZYTUI4I UTILITY: netwrite 0 chars.
EYZTE66I PROTOCOL:lmodetype=4, linemode=0, uselinemode=0

| 40 | EZYTY08I argv_fsum(0) = fomtlinp |
| 40 | EZYTY08I argv_fsum(1) = *4OurhrEa)R0,H/h |
| 40 | EZYTY08I argv_fsum(2) =                               |
| 40 | EZYTY08I argv_fsum(3) = 0                               |
| 40 | EZYTY08I argv_fsum(4) = 8                               |
| 40 | EZYTY08I argv_fsum(5) = 9                               |
| 40 | EZYTY08I argv_fsum(6) = 0                               |
| 40 | EZYTY08I argv_fsum(7) = 0                               |
| 40 | EZYTY08I argv_fsum(8) = 6                               |
| 40 | EZYTY08I argv_fsum(9) = 80                              |
| 40 | EZYTY08I argv_fsum(10) = laph.raleigh.ibm.com           |
| 40 | EZYTY08I argv_fsum(11) = xterm                         |
| 40 | EZYTY08I argv_fsum(12) =                               |
| 40 | EZYTY08I argv_fsum(13) =                               |
| 40 | EZYTY08I argv_fsum(14) =                               |
| 40 | EZYTY08I argv_fsum(15) =                               |
| 40 | EZYTY08I argv_fsum(16) = 1                             |
| 40 | EZYTY08I inherit flag = 40000000                       |

EZYTY09I login_tty: spawnp fsumocl 33

| 41 | EZYTE67I S(nfd):socketfd..ibits=00000001 obits=00000000 ebits=00000000 |
| 41 | S(nfd) pty..ibits=00000000 obits=00000000 ebits=00000100 |

| 42 | EZYTE68I Ept: #bytes = 4 pkcontrol(cntl) 1003 |
| 42 | EZYTE69I PROTOCOL: cntl = 1003 |
| 42 | EZYTE65I PROTOCOL: send IAC Data Mark. DMARK |

Chapter 15. Diagnosing z/OS UNIX Telnet daemon (otelned) problems 501
EZYTE29I indicates the start of a new z/OS UNIX Telnet client session.

EZYTE05I indicates what options were specified in /etc/inetd.conf for z/OS UNIX Telnet.

EZYTE11I indicates the resolved host name (from the client).

EZYTE11I shows the IP address of the z/OS UNIX Telnet client.

EYZTS04I indicates otelnetd agrees to send and receive authentication information.

EZYTU14I traces netwrites (writes to the client terminal).

EZYTU21I traces data from parent to client; that is, z/OS UNIX Telnet to the client terminal.

EZYTU03I indicates the number of bytes read from the client by z/OS UNIX Telnet.

EZYTU20I traces data from the client to the parent (z/OS UNIX Telnet server).

EYZTS05I indicates the client agrees to send and receive authentication information.

EZYTU17I shows otelnetd requesting that the client send authentication information for Kerberos Version 5.

EYZTS10I indicates the client agrees to receive encrypted data.

EYZTS09I indicates otelnetd agrees to send encrypted data.

EYZTS05I indicates the client agrees to send encrypted data.

EYZTS04I indicates otelnetd agrees to receive encrypted data.

EZYTU17I shows which types of encryption otelnetd supports when receiving data.

EYZTS05I shows the terminal option negotiation the client has sent/received.

EYZTS05I shows the terminal option negotiation the client has sent/received.

EYZTS04I indicates the terminal negotiation options sent to the client by the z/OS UNIX Telnet server.

EZYTU17I shows the account name on otelnetd that the client wishes to be authorized to use.

EZYTU17I shows the client authentication information for Kerberos Version 5.

Shows the Kerberos Version 5 principal of the user logging in.

EZYTU17I shows the client requesting that otelnetd enable encryption as soon as the initialization is completed.

EZYTU17I shows which types of encryption the client supports when receiving data.

EZYTU17I shows otelnetd sending to the client the type of encryption to use for the data stream (otelnetd to client) and the initial encryption data.
EZYTU17I shows otelnetd receiving from the client the type of encryption to use for the data stream (client to otelnetd) and the initial encryption data.

EZYTU17I shows otelnetd acknowledging receipt of the initial encryption data from the client.

EZYTU17I shows the client acknowledging receipt of the initial encryption data from otelnetd.

EZYTU17I shows otelnetd verifying its keyids.

EZYTS08I shows the terminal option negotiation the client has sent/received.

EZYTU17I shows the client verifying its keyids.

EZYTU17I shows all data following this command in the data stream (otelnetd to client) are encrypted using the previously negotiated method of data encryption.

EZYTU17I shows all data following this command in the data stream (client to otelnetd) are encrypted via the previously negotiated method of data encryption.

EZYTU17I traces z/OS UNIX Telnet sending terminal negotiation suboptions to the client.

EZYTE10I traces the call to tgetent(), which determines client terminal type.

EZYTE88E indicates no /etc/otelnetd.banner file was found.

EZYTO04I shows the user name with which the telnet client logged in.

EZYTE88E indicates no /etc/banner file was found.

EYZTS15I and EZYTS16I show that a state change was processed due to options/responses received from the client.

EZYTY08I traces the parameters passed to the spawned/forked child address space where the OMVS shell runs.

EZYTE67I traces the socket sets to show whether input/ibits, output/obits, or exception/ebits data has been received.

EZYTE68I shows exception data received on the parent/child connection.

Cleaning up the utmp entries left from dead processes

Assuming that you have the suggested /etc/rc script, the utmpx file is cleaned up each time the S OMVS command is issued. The utmpx file should not normally need cleaning up, as each terminal slot should be reused the next time someone logs on with that terminal.

Although during normal processing the utmp entries are cleaned up, there are the occasional incidents where zombies are created, or the user might have terminated the session abnormally. When this occurs the utmp entry for that user remains in the /etc/utmpx file until it is cleared out. There is an associated tty reserved for every entry in the /etc/utmpx file including the zombie entries. For dead entries, these ttys are not available for reuse until someone under superuser erases the /etc/utmpx file.
Tip: If you erase the file while someone is logged on, the next logoff reports not finding the utmpx entry for the user. This can be seen with a waitpid failure during that user cleanup.
Chapter 16. Diagnosing Telnet problems

This topic describes how to diagnosis Telnet problems, and contains the following sections:

- “General TN3270E Telnet server information”
- “TN3270E Telnet server definitions”
- “Diagnosing TN3270E Telnet server problems” on page 506
- “General Telnet client information” on page 514
- “Telnet client definitions” on page 515
- “Diagnosing Telnet client problems” on page 515
- “Telnet client traces” on page 518

General TN3270E Telnet server information

The Telnet protocol provides a standardized interface, through which a program on one host (the Telnet client) can access the resources of another host (the TN3270E Telnet server) as though the client were a local terminal connected to the server host.

Telnet protocol is based on the concept of a Network Virtual Terminal (NVT) and the principle of negotiated options.

An NVT is an imaginary device, providing the necessary basic structures for a standard terminal. Each host client represents an imaginary device with certain terminal characteristics that the host server can support.

The principle of negotiated options is used by the Telnet protocol because many clients and hosts require additional services beyond the base services. Various options can be negotiated. Server and client use a set of conventions to establish operational characteristics for their Telnet connection by means of the DO, DON'T, WILL, WON'T mechanism that is discussed in “Telnet commands and options” on page 522.

Component event tracing is done under the SYSTCPIP component. A subset of trace options and a subset of IPCS commands are available to Telnet. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47 and Chapter 6, “IPCS subcommands for TCP/IP,” on page 191 for details.

TN3270E Telnet server definitions

Telnet LUs must be defined correctly to both VTAM and Telnet. A VTAM APPL definition statement is needed for each Telnet LU that is used. Model application definitions can also be used. Refer to the z/OS Communications Server: SNA Resource Definition Reference for detailed information about these definitions. A corresponding LU must be specified in the BEGINVTAM section of the PROFILE data set. Refer to the z/OS Communications Server: IP Configuration Reference for detailed information about these definitions.

Restriction: All default 3270 LOGMODE entries from the table of Telnet device name parameters in the z/OS Communications Server: IP Configuration Reference are
for non-SNA sessions. You must code device types and the needed LOGMODE entries for SNA sessions. All default 3270E LOGMODES are for SNA sessions.

Diagnosing TN3270E Telnet server problems

Problems with Telnet are generally reported under one of the following categories:
- **Abends**
- **Logon problems**
- **Session hangs**
- **Incorrect output**
- **Session outages**

Use the information provided in the following sections for problem determination and diagnosis of errors reported against Telnet.

**Abends (server)**

An abend during Telnet processing should result in messages and error-related information sent to the MVS system console. A dump of the error is needed unless the symptoms already match a known problem.

**Documentation**
Code a SYSMDUMP DD or SYSABEND DD statement in the PROC used to start Telnet to ensure that a useful dump is obtained in the event of an abend.

**Analysis**
Refer to [z/OS Problem Management](#) or see Chapter 3, “Diagnosing abends, loops, and hangs,” for debugging dumps produced during TCP/IP processing.

**Logon problems (server)**

Telnet logon problems are reported when clients are unable to connect to the host application. Generally, this type of problem is caused by an error in the configuration or definitions (either in VTAM or Telnet).

If the problem can be re-created, use the DEBUG DETAIL parameter to gather diagnostic messages or trace information. Refer to the [z/OS Communications Server: IP Configuration Guide](#) for details.

**Documentation**
The following documentation should be available for initial diagnosis of Telnet login problems:
- Console Log of error messages issued by both Telnet and VTAM
- PROFILE data set
- VTAM APPL definitions for Telnet LUs

More documentation that might be needed is discussed in the following analysis section.

**Steps for analyzing logon problems (server)**

Table 31 shows symptoms of login problems and refers to the steps needed for initial diagnosis of the error. The information following the chart and associated information can be used for extended diagnosis, if the problem persists.

<table>
<thead>
<tr>
<th>Login problem</th>
<th>Analysis steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>No LUs available</td>
<td>1, 2, 6, 10, 13</td>
</tr>
</tbody>
</table>
Table 31. Telnet login problems (continued)

<table>
<thead>
<tr>
<th>Login problem</th>
<th>Analysis steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN failure</td>
<td>1, 2, 3, 4, 5, 6, 7, 8, 9, 10</td>
</tr>
<tr>
<td>x-clock (Telnet solicitor panel)</td>
<td>1, 2, 3, 4, 5, 6, 7, 10</td>
</tr>
<tr>
<td>x-clock (blank screen)</td>
<td>1, 2, 3, 6, 7, 8, 10, 12</td>
</tr>
<tr>
<td>x-clock (application panel)</td>
<td>7, 8, 10</td>
</tr>
<tr>
<td>Incorrect USSMSG or DEFAULTAPPL</td>
<td>3, 4, 5, 6, 1, 13, 14</td>
</tr>
</tbody>
</table>

1. Have VTAM APPL definition statements been coded correctly?

   **Note:** There must be a VTAM definition statement or model application name for each LU coded in the PROFILE data set.

2. Is the VTAM node containing the Telnet LU definitions active?

3. Is there a DEFAULTAPPL coded in the PROFILE data set?

4. Is the host application (or DEFAULTAPPL) active?

5. Is there an ALLOWAPPL statement coded that includes the requested application?

6. Have comment delimiters been added or removed as needed in the BEGINVTAM section of the PROFILE data set?

7. Have correct LOGMODEs (or required overrides for SNA) been coded in the PROFILE data set?

8. Does the host application have BIND (session parameter) requirements that are not met by the specified LOGMODE?

9. Is the MSG07 parameter coded in the PROFILE.TCP data set?

   **Note:** MSG07 returns information to the end user indicating the reason for the failure.

10. Are any abends (in VTAM, host application, or TCP/IP) indicated on the MVS system console?

    **Note:** If an abend occurred, refer to the section on abends to continue investigation of the problem.

11. Check the PROFILE data set for the IP to LU mapping.

12. Is an SSL client attempting to connect to a basic port or is a basic client trying to connect to an SSL port?
13. Use the D TCPIP,,T,PROFILE,DETAIL command to view the active profile definitions.

14. Determine if USSTCP within the TCPIP PROFILE points at the correct USSTAB, because this could also cause an incorrect USSMSG to be displayed.

If the problem still occurs after following the preceding procedure and making any needed changes, obtain the following documentation:

- TELNET display of the LUNAME or CONN ID of affected client, for example, D TCPIP,,T,CONN,LUN=luname.
- VTAM DISPLAY of Telnet LU.
- VTAM DISPLAY of the target host application.
- Activate DEBUG DETAIL and review additional diagnostic information this function provides.

For information about the Telnet Display command options, refer to the [z/OS Communications Server: IP System Administrator’s Commands](http://www.ibm.com/systems/z/os/). The following documentation might also be needed in some cases, but it is suggested that your IBM Software Support Center be contacted before this documentation is obtained:

- TCP/IP packet trace and CTRACE with TELNET option filtered on the IP address of the failing client.
- VTAM buffer trace of the Telnet LU.
- VTAM INTERNAL TRACE (VIT) with options (API,PIU,MSG,PSS,NRM,SSCP).
- Dump of the Telnet address space. To capture the necessary areas of storage in the DUMP command, include:
  
  SDATA=(CSA,LSQA,PSA,RGN,SQA,SUM,SWA,TRT,LPA)

For information about obtaining VTAM traces, refer to [z/OS Communications Server: SNA Operation](http://www.ibm.com/systems/z/os/) or to [z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT](http://www.ibm.com/systems/z/os/) for your release. Instructions on obtaining a dump can be found in [z/OS MVS Diagnosis: Tools and Service Aids](http://www.ibm.com/systems/z/os/) for your release of MVS.

Session hangs (server)

This section discusses diagnosis of a hang after a session has been successfully connected. A hang would be indicated by the keyboard remaining locked on the client side of the session, with no data being sent to or received from the server host.

If a problem is recreatable, you can use the DEBUG TRACE parameter. Refer to the [z/OS Communications Server: IP Configuration Guide](http://www.ibm.com/systems/z/os/) for details.

Documentation

To determine the cause of a Telnet session hang, the following documentation is usually required:

- CTRACE specifying the TELNET option filtered on the IP address of the failing client.
- In some cases a VTAM buffer trace of the Telnet LU might be needed.
- Information about what was seen at the client screen.
Steps for analyzing session hangs (server)
The preceding traces are essential to finding the reason for the session hang. Data entered at the client terminal is sent to Telnet on the TCP/IP connection. The TCP/IP packet trace shows the data arriving at or leaving the stack. CTRACE with the option Telnet specified shows the data coming into and out of Telnet (from both the stack and VTAM). Some processing steps during this time are also included in the trace. The CTRACE with Telnet option shows what Telnet does with this data.

The VTAM buffer trace shows the data as received by VTAM to be forwarded to the host application. Following the data flow through the traces between VTAM, TCP/IP, and Telnet provides an indication of where the problem is occurring.

The following list suggests information to check in the traces. Refer to z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT or to SNA Network Product Formats for more information about VTAM buffer trace output.

1. Does the packet trace show data passed to TCP/IP? If not, the problem is in client or emulator code. If data is in the trace, continue with Step 2.

2. Does CTRACE with TELNET option show data passed to Telnet? The TELNET option shows data coming into Telnet from the stack and also going out to VTAM (the reverse for outbound data). If not, the error is in the TCP/IP platform code. Otherwise, continue with Step 3.

3. Does VTAM buffer trace show data passed from Telnet? If not, problem is in the Telnet code. Otherwise, continue with Step 4.

4. Does VTAM buffer trace show data passed to host application? If not, problem is in VTAM code. If buffer trace shows correct data, continue with Step 5.

5. Does the buffer trace show data coming from the host application? If not, the problem is in the host application. Contact your host application support center for these products. Otherwise, continue with Step 6.

6. Does the buffer trace show data sent back to the Telnet LU? If not, the problem is in VTAM. Otherwise, continue with Step 7.

7. Is the last data from the application seen in the CTRACE with TELNET option output? If not, the problem is in Telnet. Otherwise, continue with Step 8.

8. Does the packet trace show the data sent to the client? If not, the error is in TCP/IP platform. Otherwise, continue with Step 9.

9. Check the data in the packet trace output to see if unlock keyboard is set on in the data stream. If unlock is set in the output data, the problem is in the emulator or client code. Otherwise, continue with Step 10 on page 510.
10. Check the last data received by the Telnet LU in the VTAM buffer trace. If unlock is set in that data stream, or end bracket or change direction is set in the RH, the problem is in the Telnet code. If none is set, the host application did not allow for unlocking of the keyboard. Contact your host application software support.

If the preceding problem determination shows the error to be in the TCP/IP platform or Telnet code, a dump is needed to allow a more detailed investigation of the problem.

**Incorrect output (server)**

Problems with incorrect output are reported when the data sent to the client is not seen in its expected form. This could be garbled data that is unreadable on the screen, a blank screen when output is expected, or screen formatting problems. These problems are generally traced back to logmode issues. Ensure the primary and alternate screen sizes in the logmode used are correct for the TN3270 or TN3270E emulator that you are using. The logmode coded in the TCPIP profile is suggested to VTAM as the correct logmode for this device type. The VTAM PLU application determines the actual logmode that is used. Therefore this application must be configured correctly to use the appropriate logmode.

If a problem is recreatable, you can use the Telnet DEBUG features. Refer to the [z/OS Communications Server: IP Configuration Guide](https://www.ibm.com/support/docview.ws/docuid/pea-3d618e8) for details.

**Documentation**

Documentation needed to find the source of the error in an incorrect output problem would be:

- CTRACE with TELNET option and the FULLDATATRACE parm active in the profile
- VTAM buffer trace of the Telnet LU, with AMOUNT=FULL specified
- Client screen output information

**Steps for analyzing incorrect output (server)**

The main goal of diagnosing this type of problem is to determine if the data was sent incorrectly by the host application or corrupted by VTAM, TCP/IP, Telnet, or Telnet client code.

<table>
<thead>
<tr>
<th>Incorrect Output</th>
<th>Analysis Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blank screen</td>
<td>1, 6, 7</td>
</tr>
<tr>
<td>Garbled or unreadable characters on the screen</td>
<td>2, 3, 4, 5, 6, 7</td>
</tr>
<tr>
<td>Incorrectly formatted screen</td>
<td>6, 7</td>
</tr>
</tbody>
</table>

See [Table 32](#) to identify which of the following steps to use in determining the cause of the error.
1. Was the last output data seen in a SEND DATA to CLIENT CTRACE entry displayed at the terminal emulator? If not, the problem is in the client or emulator. Contact your emulator provider for this product. If the last output was seen at the terminal, go to step 9 on page 509 of the analysis procedure in Session hangs (server) and continue your diagnosis.

2. Was the TELNET command entered with TRANSLATE specified? If so, make sure the translate table is compatible with the capabilities of the client device. If compatible or no TRANSLATE was used, continue with Step 4.

3. The CTRACE with TELNET option entries show the data as it arrived from VTAM and again as it goes to the stack. FULLDATATRACE parameter should be specified in the profile when looking for a problem in the data stream. Examine the CTRACE and compare the DATA from VTAM entries to the DATA to CLIENT entries. If they are different, then Telnet altered the data stream. If not, the problem is with the TCP/IP platform code. Otherwise, continue with Step 4 on page 509.

4. In the data trace output, is the data stream sent by the server the same as received from VTAM? If not, the problem is with the Telnet code. Otherwise, continue with Step 5 on page 509.
   Tip: If the client is an ASCII device, these might be different due to EBCDIC-to-ASCII translation. Check the appropriate translate table for compatibility with the client device.

5. In the VTAM Buffer trace with the FULL option specified, is the data in the VTAM USER entry (data received by VTAM) the same as the data in the VTAM BUFF entry (data sent by VTAM)? If not, VTAM has corrupted the data. Otherwise, incorrect data was sent by the application. Contact the IBM Software Support Center for the host application.

6. Is the LOGMODE specified for the negotiated terminal type valid for the actual client device?
   Tip: A VTAM session display specifying the SID for the session shows the actual logmode selected by the SNA application.

7. Does the device characteristics information in the BIND sent by the host application match the device characteristics information in the specified LOGMODE entry, and are these characteristics appropriate for the emulator in use?
   This can be checked by comparing the specified LOGMODE entry (refer to z/OS Communications Server: SNA Customization) with the BIND in the buffer trace at logon to the selected application. Refer to the z/OS Communications Server: SNA Programming for information of the BIND RU as well as SNA Formats.
If the problem is not found after using the analysis steps, contact your IBM Software Support Center for additional diagnostic suggestions.

Session outages (server)

Session outages are reported as an unexpected termination of the TCP/IP connection or the Telnet-to-host application session. A session that has been disconnected or terminated results in the client being returned to the panel where the initial TELNET command was entered and message EZZ6034I is issued. Refer to z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM).

Telnet sessions can be terminated due to TELNETPARMS specified in the PROFILE data set. Telnet ends a session if there is no activity on the SNA side of the connection for the amount of time specified in the INACTIVE parameter. Telnet checks for dormant sessions on the IP side of the connection using the SCANINTERVAL/TIMEMARK parameters specified. When appropriate, the connection is terminated due to this processing. Refer to the SCANINTERVAL/TIMEMARK parameters in the z/OS Communications Server: IP Configuration Reference for additional information.

Documentation

The following documentation is needed for initial investigation of problems reported as session outages.

Abnormal connection terminations are reported using EZZ6034i message with appropriate reason code (RCODE). If DEBUG SUMMARY is coded in the Telnet profile, then normal connection terminations are also reported. If DEBUG DETAIL is coded, then additional diagnostic information is reported using EZZ6035I messages. These messages can be spooled to either the console or joblog. Examination of the RCODE carried in these messages is the first step to diagnosing this type of problem.

Steps for analyzing session outages (server)

The preceding output is needed to begin diagnosis of a session outage reported against Telnet. It is also helpful to know what kind of processing the Telnet user was doing at the time of the interrupted session.

Perform the following steps for initial investigation of a Telnet session outage:

1. If a timeout due to inactivity or termination due to TIMEMARK processing is suspected, check the values set in the PROFILE data set.

2. Additional messages are issued for session outages when the Telnet DEBUG features are active. Refer to the z/OS Communications Server: IP Configuration Guide for details of the Telnet DEBUG features.

3. Check the documentation listed in "Documentation" for indications of an error.
   • If the MVS system console indicates a VTAM error, continue diagnosis with your VTAM programmer.
   • If the console shows a Telnet or TCP/IP error, check z/OS Communications Server: IP Messages Volume 1 (EZA) or z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM) and follow the directions for system programmer response for the message.
   If messages are found that do not lead to an accurate diagnosis and resolution of the error, search the APAR data base, available at http://
4. If only one Telnet user session was affected, continue with step 5. Otherwise, go to step 7.

5. If the problem can be re-created by performing the same operation or processing, run the following traces:
   - TCP/IP packet trace filtered using the IP address of the failing client
   - Component Trace output (CTRACE) specifying the Telnet option
   - VTAM Internal Trace (VIT)
   - VTAM buffer trace output with AMOUNT=FULL specified.

   **Note:** Contact your IBM Software Support Center for information about options needed before running these traces.

6. If all Telnet user sessions were interrupted, do one of the following:
   - Check the MVS system console and LOGREC for abends.
   - Check for loss of network connectivity. Verify whether all the TELNET users come in through the same channel interface or through a common router.

7. If there are no messages or abends and all Telnet user sessions have been disconnected, the traces listed in Step 5 is needed during a recurrence of the failure.
   A dump of the TCP/IP address space including the TCP/IP dataspace or a dump of the Telnet address space should be taken at this time. To capture the necessary areas of storage in the DUMP command, include:
   \[SDATA=(CSA,LSQA,PSA,RGN,SQA,SUM,SWA,TRT,LPA)\]
   If Telnet is running in the TCP/IP address space, capture the trace dataspace by including:
   \[DSPNAME=('tcpip_procname'.TCPIPDS1)\]

   Instructions on obtaining a dump can be found in [z/OS MVS Diagnosis: Tools and Service Aids](https://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS) for your release of MVS.

---

**Special considerations when using SSL encryption support**

Because data flowing across the connection between the client and the server is encrypted, the data field in the packet trace is also encrypted after SSL handshaking is completed. If problem determination requires seeing Telnet handshake or user data, you also need to run Component Trace to see the decrypted data field. When starting Component Trace, specify `options=(TELNET)` and use IPCS to format the Component Trace. For more information on Component Trace, see Chapter 5, “TCP/IP services traces and IPCS support,” on page 47.

The Telnet Component Trace records contain the connection ID in the CID field. The connection ID in the trace corresponds to the connection ID output of the connection display command. Use this field to locate records related to the client in...
question. After an LUname has been assigned, the Component Trace User field shows the LUname, providing additional data for locating your client.

The following Component Trace records might be of interest:

**SKSCINIT Succeeded**
SSL handshaking completed and subsequent data on this connection is encrypted.

**Receive Data from Client**
The Data from Client field of this record contains the decrypted data coming from the client.

**Send Data to Client**
The Data to Client field of this record contains the decrypted data going to the client.

Following is a sample Send Data to Client Component Trace record:

```
MVS181 TELNET 7001004 12:49:06.354966 Send Data to Client
HASID..002A PASID..002A SASID..002A MODID...EZBTTSND
TCB....0000000 REG14...8937FA USER...TCPM1011 DUCB...0000000
CID....092552CE SEQ....000024BE
...<snip>...
ADDR...08167AB0 LEN....00000004 Number of Bytes Sent
+0000 0000002C
ADDR...7F687950 LEN....00000002C Data to Client
+0000 F5C1115D 7F1D4011 40401DC8 C9D2D1F5 5A.)"..HIKJ5
+0010 F5C1115D 7F1D4011 40401DC8 C9D2D1F5 5A.)"..HIKJ5
+0020 F6F7F0F0 C140C5D5 E3C5D940 E4E2C5D9 6700A ENTER USER
CID....092552CE SEQ....000024BE
```

**Telnet Component Trace data**

To help associate a Component Trace entry with a particular client, the following two Component Trace fields contain data unique to Telnet:

**CID**
The connection ID for the connection. This is equivalent to the connection ID output from the connection display command.

**USER**
The LUname associated with the client, after it has been assigned. Prior to LUname assignment, this field might be null or contain the TCP procedure name. The LUname is not set until after the completion of the Telnet handshake.

Use these fields in Component Trace formatting to limit the records to be displayed. For example, if you want Telnet records for a client connection ID X'021F' with the LUName TCPM1011, code the following IPCS command:

```
CTRACE COMP(SYSTCP/IP) SUB((proc_name)) FULL JOBLIST (TCPM1011)
   OPTIONS((TELNET,CID(X'0000021F'))) 
```

**Tip:** Some of the records pertinent to the connection are not shown when the output is restricted by the CID and USER options. However, it is often helpful to use the output produced by these filters as a starting point.

**General Telnet client information**

The Telnet client code runs under TSO in the TSO user’s address space. The Telnet client uses the VTAM interface, like other TSO applications, to send data out to the user’s terminal.
The Telnet client can run in line mode, when accessing an ASCII host, or run in full-screen mode, if the remote host provides 3270 full-screen support.

**Telnet client definitions**

The Telnet command must be authorized to be issued by TSO users. Refer to the z/OS MVS Initialization and Tuning Guide for information about making Telnet an authorized command. There are no other special definitions or setup requirements to run the Telnet client.

**Diagnosing Telnet client problems**

Problems that might involve the Telnet client are usually reported as one of the following types:

- Abends
- Session hangs
- Incorrect output

Use the information in the following sections for problem determination and diagnosis of errors reported in the Telnet client.

**Abends (client)**

An abend in the TELNET client should result in messages and error-related information being sent to the MVS system console. These abends should affect only the TSO user that was running Telnet. A dump of the error is needed unless the symptoms match a known problem.

**Documentation**

Code a SYSMDUMP DD or SYSABEND DD statement in the TSO PROC to ensure that a useful dump is obtained in the event of an abend. See Chapter 3, "Diagnosing abends, loops, and hangs," on page 25, for more information.

**Analysis**

Refer to z/OS Problem Management or see Chapter 3, "Diagnosing abends, loops, and hangs," on page 25 for more information about debugging dumps produced during TCP/IP processing.

**Session hangs (client)**

This section discusses diagnosis of a hang after a session has been successfully connected. A hang is indicated by the keyboard remaining locked after sending or receiving data from the remote host.

There are many components involved in the transfer of data from a locally attached device through a Telnet session. Any one of these might be the cause or a contributing factor to the hang. Each must be investigated to define the area responsible for the failure.

**Documentation**

To determine the cause of a Telnet client session hang, the following is needed:

- Information about what was seen at the client screen
- VTAM buffer trace of the local device LU
- VTAM internal trace (if the error appears to be in VTAM)
- VTAM TSO trace of the user ID issuing Telnet
- GTF trace of SVC93 and SVC94 (TGET/TPUT)
- Telnet client trace
• Dump of the TSO user’s address space
• TCP/IP packet trace and CTRACE with TELNET option on remote host (if possible)

The preceding list of documentation is a complete list that includes documentation needed to resolve most types of hangs. All of the indicated data might not be needed for each occurrence of a hang. The following analysis section provides information about what types of data might be needed through each diagnostic step.

Steps for analyzing session hangs (client)

To assist with diagnosis of a Telnet client hang, it is helpful to be familiar with the components involved and understand which ones interface directly with each other. In the case of a Telnet from an MVS client to a remote host, the following occurs:
• Data is entered by the user and then passed by VTAM to TSO.
• Data is passed from TSO to Telnet client code.
• Data is transferred across the TCP/IP connection to the remote host.
• The remote server sends data to the target application.

Note: It is suggested that a VTAM buffer trace and a Telnet client trace be run while recreating the problem for initial debugging purposes. A sample of the client trace output can be found in Figure 61 on page 519. Refer to z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures or to SNA Network Product Formats for more information about VTAM buffer trace output.

Perform the following steps for diagnosing a Telnet client hang, along with the documentation needed in each situation.

1. Does the hang affect other Telnet clients? If so, go to “Diagnosing TN3270E Telnet server problems” on page 506. Otherwise, continue with Step 2.

2. Was the last activity at the terminal input or output? If input, go to step 5. If output, continue with Step 3.

3. Check the data in the VTAM buffer trace to see if unlock keyboard is set on in the data stream. If unlock is set on in the data stream, the problem is in the emulator, control unit, or terminal device. If not, check the Telnet client trace to ensure the output data stream matches what is seen in the buffer trace. If the data streams match, the remote host application has not unlocked the keyboard. Contact your IBM Software Support Center for more help with the problem. If the data streams do not match, continue with Step 4.

4. The problem appears to be in the VTAM TSO area. Recreate the error while running the Telnet client trace, a GTF trace of SVC93 and SVC94, a VTAM TSO trace, and a VTAM buffer trace. Contact your IBM Software Support Center for assistance in interpreting the traces.

5. Check the VTAM buffer trace to ensure input data was received by VTAM and passed to TSO. If the last data entered at the terminal is not in the VTAM
buffer trace, the problem is in the PC emulation code or in the control unit. If input data is correct, continue with Step 6.

6. Is the entered data seen in client trace output? If not, the problem is in VTAM TSO. Follow the instructions in Step 4 on page 516. If data is in the client trace, the error needs to be diagnosed from the server host. See "Session hangs (server)" on page 508 and follow the path for “last activity at the terminal was input.”

Documentation listed earlier, but not referenced in the previous debugging steps, can be useful in the following situations:

- VTAM internal trace

  Note: Data is seen in “BUFF VTAM” VTAM buffer trace entry (entering VTAM from the terminal), but not in the “BUFF USER” VTAM buffer trace entry (passed from VTAM to TSO).

- Dump of TSO user’s address space

  Note: Data is seen in the “BUFF USER” VTAM buffer trace entry, but not in the VTAM TSO trace or Telnet client trace.

Contact the IBM Software Support Center for assistance with further diagnosis when data is obtained in these situations.

Tip: Information about starting and examining traces is discussed in “Step for starting Telnet client traces” on page 518.

Incorrect output (client)

Problems with incorrect output are reported when the data seen at the terminal is not in its expected form. This might be garbled data that is unreadable, a blank screen when output is expected, or screen formatting problems.

Documentation

Documentation needed to find the source of the error in an incorrect output problem is:

- VTAM buffer trace of the local device LU
- VTAM TSO trace of the user ID issuing Telnet
- GTF trace of SVC93 and SVC94
- Telnet client trace
- Client screen output information

Steps for analyzing incorrect output (client)

The main goal of diagnosing this type of problem is to determine if the data was sent incorrectly by the host application or was corrupted by the Telnet server, Telnet client, TSO, or VTAM code. The following analysis steps should allow quick determination of whether the problem is a Telnet client problem or must be addressed from the server host.

1. If new data sent to the screen cannot be read (garbled or formatted incorrectly), go to Step 4 on page 518. Otherwise, continue with Step 2 on page 518.
2. Was the last output data seen in the VTAM buffer trace displayed at the terminal? If not, the problem is in the emulator or device. Contact the appropriate IBM Software Support Center. Otherwise, continue with Step 3.

3. Does the last output data in the Telnet client trace match the data in the VTAM buffer trace? If not, contact your IBM Software Support Center with the client trace, a VTAM TSO trace, and a VTAM buffer trace of the error. Otherwise, this problem must be investigated from the Telnet server side. Continue with the investigation as a Telnet server session hang.

4. Was the TELNET command entered with TRANSLATE specified? If so, make sure the translate table is compatible with the capabilities of the output device. If the table is compatible or no TRANSLATE was used, continue with Step 5.

5. Check the Telnet client trace and VTAM buffer trace. If the data is different, contact your IBM Software Support Center with the client trace, a VTAM TSO trace, and a VTAM buffer trace. Otherwise, continue investigating as a Telnet server incorrect output problem.

6. If the data is formatted incorrectly for the screen size, check the defined session parameters for the negotiated device type for the Telnet server.

If the problem is not found after using the analysis steps, contact your IBM Software Support Center for more diagnostic suggestions.

Telnet client traces

The Telnet client trace shows data received from the remote server to be sent to the local device, and data from the device to be forwarded to the remote host. This includes attention interrupts and some negotiation data seen at the beginning of the session. Data from the initial Telnet negotiation is not seen, only an indication that it is negotiation data and the number of bytes received.

Step for starting Telnet client traces

Before issuing the Telnet command, the following command should be issued from the TSO “ready” prompt or command line to allocate the trace data set:

ALLOC F(DEBUGFIL) DA(data.set.name) NEW

Trace data is written to the data set indicated in the command.

The trace is invoked by issuing the Telnet command with the DEBUG option:

TELNET hostname (DEBUG

Trace example (client)

Figure 61 on page 519 is sample output from a Telnet client trace showing part of a Telnet login to a remote host.
Figure 61. Telnet client trace (Part 1 of 4)
Figure 61. Telnet client trace (Part 2 of 4)
Figure 61. Telnet client trace (Part 3 of 4)
Following are short descriptions of the numbered items in the trace:

1. This entry shows the data received from the Telnet server and indicates the number of bytes. The example here is during initial negotiation and does not include the actual data received.

2. This indicates the type of data received.

3. This entry indicates the data received from TCP (from the Telnet server).

4. The actual hexadecimal data received. This trace example is of a transparent mode session, so the data is in EBCDIC. In a line mode session, the data would be in ASCII, and there would be one character per line (like the input data later in the trace).

5. This is the translation of the previous hexadecimal data. All hexadecimal characters that translate into readable data are displayed.

6. This entry indicates data received from the terminal or PC.

7. Following this line is the actual input data. There is a single hexadecimal byte per line that is translated into its readable form.

8. This entry follows the input data and indicates the number of bytes received from the terminal.

9. This entry indicates the data from the host application (using the Telnet server) that is being sent to the terminal.

Telnet commands and options

For information about Telnet connection negotiations, refer to RFC 2355. Table 33 on page 523 describes the Telnet commands from RFC 854, when the codes and code sequences are preceded by an IAC. For more information about Telnet commands, refer to RFC 854.
### Table 33. Telnet commands from RFC 854

<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE</td>
<td>X'F0'</td>
<td>End of subnegotiation parameters.</td>
</tr>
<tr>
<td>NOP</td>
<td>X'F1'</td>
<td>No operation.</td>
</tr>
<tr>
<td>Data Mark</td>
<td>X'F2'</td>
<td>The data stream portion of a Synch. This should always be accompanied by a TCP Urgent notification.</td>
</tr>
<tr>
<td>Break</td>
<td>X'F3'</td>
<td>NVT character BRK.</td>
</tr>
<tr>
<td>Interrupt Process</td>
<td>X'F4'</td>
<td>The function IP.</td>
</tr>
<tr>
<td>Abort output</td>
<td>X'F5'</td>
<td>The function AO.</td>
</tr>
<tr>
<td>Are You There</td>
<td>X'F6'</td>
<td>The function AYT.</td>
</tr>
<tr>
<td>Erase character</td>
<td>X'F7'</td>
<td>The function EC.</td>
</tr>
<tr>
<td>Erase Line</td>
<td>X'F8'</td>
<td>The function EL.</td>
</tr>
<tr>
<td>Go ahead</td>
<td>X'F9'</td>
<td>The GA signal.</td>
</tr>
<tr>
<td>SB</td>
<td>X'FA'</td>
<td>Indicates that what follows is subnegotiation of the indicated option.</td>
</tr>
<tr>
<td>WILL (option code)</td>
<td>X'FB'</td>
<td>Indicates the desire to begin performing, or confirmation that you are now performing, the indicated option.</td>
</tr>
<tr>
<td>WON'T (option code)</td>
<td>X'FC'</td>
<td>Indicates the refusal to perform, or continue performing, the indicated option.</td>
</tr>
<tr>
<td>DO (option code)</td>
<td>X'FD'</td>
<td>Indicates the request that the other party perform, or confirmation that you are expecting the other party to perform, the indicated option.</td>
</tr>
<tr>
<td>DON'T (option code)</td>
<td>X'FE'</td>
<td>Indicates the demand that the other party stop performing, or confirmation that you are no longer expecting the other party to perform, the indicated option.</td>
</tr>
<tr>
<td>IAC</td>
<td>X'FF'</td>
<td>Data byte 255.</td>
</tr>
</tbody>
</table>

Table 34 lists the options available for Telnet commands from RFC 1060. For more information about Telnet protocols, refer to RFC 1060 and RFC 1011.

### Table 34. Telnet command options from RFC 1060

<table>
<thead>
<tr>
<th>Option</th>
<th>Option (Hex)</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Binary Transmission</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Echo</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Reconnection</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Suppress Go Ahead</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Approx Message Size Negotiation</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Status</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>Timing Mark</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>Remote Controlled Trans and Echo</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Output Line Width</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>Output Page Size</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>Output Carriage-Return Disposition</td>
</tr>
</tbody>
</table>
Table 34. Telnet command options from RFC 1060 (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Option (Hex)</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>B</td>
<td>Output Horizontal Tab Stops</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>Output Horizontal Tab Disposition</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>Output Formfeed Disposition</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>Output Vertical Tabstops</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>Output Vertical Tab Disposition</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Output Linefeed Disposition</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>Extended ASCII</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>Logout</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>Byte Macro</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Data Entry Terminal</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>SUPDUP</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>SUPDUP Output</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>Send Location</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>Terminal Type</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>End of Record</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>TACACS User Identification</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>Output Marking</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>Terminal Location Number</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>Telnet 3270 Regime</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>X.3 PAD</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>Negotiate About Window Size</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>Terminal Speed</td>
</tr>
<tr>
<td>33</td>
<td>21</td>
<td>Remote Flow Control</td>
</tr>
<tr>
<td>34</td>
<td>22</td>
<td>Linemode</td>
</tr>
<tr>
<td>35</td>
<td>23</td>
<td>X Display Location</td>
</tr>
<tr>
<td>255</td>
<td>FF</td>
<td>Extended-Options-List</td>
</tr>
</tbody>
</table>
Chapter 17. Diagnosing Simple Mail Transfer Protocol (SMTP) problems

The Simple Mail Transfer Protocol (SMTP) is used to transfer electronic mail reliably and efficiently. Recipients of the mail can be users on a local host, users on Network Job Entry (NJE), or users on remote TCP/IP hosts. The SMTPNOTE command is used to send mail to a local or remote host.

This topic describes how to diagnose problems with SMTP and contains the following sections:

- “Sender SMTP”
- “Receiver SMTP”
- “SMTP environment”
- “SMTP definitions” on page 526
- “Diagnosing SMTP problems” on page 526
- “ADDRBLOK data set” on page 531
- “SMTP RESOLVER trace” on page 533

For information about diagnosing problems with the other z/OS Communications Server mail application, z/OS UNIX sendmail, see Chapter 18, “Diagnosing z/OS UNIX sendmail and popper problems,” on page 537.

Sender SMTP

The sender SMTP performs the following functions:

- Receives notes from the SMTPNOTE CLIST by way of a TSO TRANSMIT command or through a batch job using IEBGENER
- Resolves the host name of recipients by way of the RESOLVER module
- Opens a TCP/IP connection with the SMTP server
- Returns mail to the sender, if mail is undeliverable

Receiver SMTP

The receiver SMTP performs the following functions:

- Accepts mail from remote TCP/IP hosts
- Delivers mail to the local user using TSO TRANSMIT to the spool for the local user
- Forwards mail to the next “hop”, if this is not the final destination
- Rejects mail for recipients who are not valid

SMTP environment

Figure 62 on page 526 shows the SMTP environment.
SMTP definitions

In order to run correctly, SMTP must be defined correctly for both TCP/IP and SMTP. The SMTP.CONFIG and TCPIP.DATA data sets contain the main sender and receiver parameters. The SMTPNOTE CLIST must be customized for your particular installation. The IEFSSNxx member of PARMLIB must be modified to include the following lines:

TNF, MVPSSI
VMCF, MVPXSSI, nodename (where nodename is the NJE node name)

For more information about restartable VMCF and TNF, refer to z/OS Communications Server: IP Configuration Guide.

Restrictions:
- The NJE node name, nodename, must be the same as the hostname and the smtpnode in the SMTPNOTE CLIST.
- SMTP can handle only one NJE node name.

Refer to the z/OS Communications Server: IP Configuration Reference for more information about configuring SMTP.

Diagnosing SMTP problems

SMTP problems are generally reported under one of the following categories:
- Abend
- Spooling
- SMTP does not deliver mail
- SMTP loop
- Mail item has incorrect output

Abends

An abend during SMTP processing should result in messages and error related information being sent to the system console. A dump of the error is needed unless the symptoms already match a known problem.

Documentation

The following documentation is needed for abends:
- Dump

Guideline: Code a SYSMDUMP DD or SYSABEND DD statement in the SMTP cataloged procedure to ensure that a useful dump is obtained in the event of an abend.
Output from the started SMTP procedure
SYSLOG and LOGREC output for the time of the error

Analysis
Refer to z/OS Problem Management or see Chapter 3, “Diagnosing abends, loops and hangs,” on page 25, for information about debugging dumps produced during SMTP processing.

Spooling problems
Spooling problems can occur when the VERB command is being used and the origination information is either missing or not valid. The VERB command requires the originator to have a valid JES user ID and node ID on the SMTP sending system. The originator information is taken from the TSO XMIT (Transmit) command headers.

For more information about the VERB command, refer to z/OS Communications Server: IP User's Guide and Commands.

SMTP does not deliver mail
This section discusses diagnosis of mail items that are not delivered to the recipient. Problems with mail not being forwarded can be divided into the following categories:
- Mail not forwarded to a local user
- Mail not forwarded to a user on another NJE host
- Mail not forwarded to remote TCP/IP host

Steps for undeliverable mail items
For all categories of this problem, perform the following steps:
1. Check whether an SMTP EXIT program is installed and activated for outbound mail.
2. Check the SMTP.CONFIG data set for the EXITDIRECTION BOTH statement.
3. If EXITDIRECTION BOTH is coded, activate DEBUG in SMTP.CONFIG data set.
4. Check SYSDEBUG log to see if SMTP exit program is rejecting the mail.
5. If yes, check the SMTP exit program.

Documentation
The following documentation should be available for initial problem diagnosis:
- TSO console log with the SMTPNOTE messages
- Job log output from the started SMTP procedure
- SMTP.CONFIG data set
- TCPIP.DATA data set

Other documentation that might be needed is discussed in the following section.
Steps for analyzing mail delivery problems
Perform the following steps to analyze the problem:

• If the problem is that mail was not forwarded to a local user:
  1. Was SMTPNOTE customized for your installation?
  2. Is the local user one that is coded as a restricted user in the SMTP:CONFIG data set?
  3. Are the JES node parameters coded correctly? This can be determined by issuing a TSO TRANSMIT of a data set to the user and node. If the transmission works, the JES node parameters are coded correctly.
  4. Activate DEBUG in SMTP:CONFIG data set. Check SYSDEBUG log to see if SMTP exit program is rejecting the mail. If yes, customer needs to check SMTP exit program.
  5. If TSO TRANSMIT fails with message INMX202I Node name SMTPNODE not defined to JES when testing customization of SMTPNOTE variables, check that the SMTPNODE variable used by SMTPNOTE is defined correctly in the JES2PARM data set as a node name. Also check that the SMTPJOB name used by SMTPNOTE is not defined as a node name to JES.

• If the problem is that a mail note was not forwarded to an NJE host:
  1. Follow the preceding steps for mail that was not forwarded to a local user.
  2. Is SMTP configured as an NJE gateway?
  3. Was SMTPNJE successfully run to create the NJE host table data set?
  4. Check whether the NJE node is in the NJE host table data set.

Refer to the z/OS Communications Server: IP Configuration Reference for information about SMTP configuration.

• If the problem is that mail was not forwarded to a remote TCP/IP host:
  1. Use the SMSG SMTP QUEUE command to see the status of the note.

Browse the ADDRBLOK data set for obvious errors. The ADDRBLOK data set is described in "ADDRBLOK data set" on page 531.

Restriction: You should stop SMTP in order to obtain the ADDRBLOK data set as it was sent, because the data set is updated during processing and deleted when the number of recipients equals 0.

  2. Has the host name been resolved to an IP address?

Run RESOLVER trace to see if the host name is resolved correctly. The RESOLVER trace is explained in "SMTP RESOLVER trace" on page 533.

Check if IPMAILERADDRESS and RESOLVERUSAGE NO is configured in SMTP:CONFIG data set. If yes, this causes all mail destination for IP networks to be sent to this IP address. Use packet trace to ensure that SMTP can connect to this IP address and that there is another remote SMTP mailer at this address. Check if IPMAILERNAME .... ALL is configured in SMTP:CONFIG data set. If yes, this causes all mail destination for IP networks to be sent to this IPMAILERNAME. Use RESOLVER traces to ensure IPMAILERNAME is resolved correctly. Use packet trace to ensure that SMTP can connect to the IP addresses associated with the IPMAILERNAME and that another remote SMTP mailer is at these addresses.

Check if IPMAILERNAME is configured. Is message EZA5647E generated? Or when resolver traces are active, is the following trace message generated
Potential loop IP mailer? If yes, HOME IP addresses in the IP list are associated with the IPMAILERNAME. Activate resolver traces in SMTP to understand how SMTP resolved the IPMAILERNAME. Either correct the IPMAILERNAME, or remove the HOME IP address from the list of addresses associated with the IPMAILERNAME in the DNS database or local host tables.

3. Is the remote TCP/IP/SMTP server running?
   Use the PING command to see if the remote TCP/IP is running, or try using Telnet to access the IP address of the remote mail server using port 25.
   **Guideline:** Options coded in the SMTP.CONFIG data set directly affect how and when names are resolved by name servers and how often mail delivery is attempted, if there is a problem in the network or the remote NAME server or if the SMTP server is not running.

If the problem still occurs after following this procedure and making any needed changes and corrections, obtain the following documentation and contact the IBM Software Support Center:
- SMTP.CONFIG data set
- TCPIP.DATA data set
- Output from SYSERR and SYSDEBUG of the started SMTP procedure with DEBUG turned on
- ADDRBLOCK data set

**SMTP Loop**
This section discusses diagnosis of the SMTP address space looping during processing.

**Documentation**
If SMTP is looping and printing out AMPX... messages to SYSERR, do the following:
- Examine the SYSERR output for AMPX... error messages and traceback information of called routines.
- Call the IBM Software Support Center with this information.
  **Tip:** Coding the NOSPIE run-time parameter in the SMTP cataloged procedure might help alleviate a Pascal error recovery loop. For example, code:
  ```
  //SMTP PROC MODULE=SMTP,DEBUG=,PARMS='NOSPIE',SYSERR=SYSERR
  ```


**Mail item has incorrect output**
Problems with incorrect output are reported when the recipient does not see the mail item in its expected form.

**Documentation**
Use the following documentation to confirm the source of the error:
- SMTP.CONFIG data set
- TCPIP.DATA data set
- Output from SYSERR and SYSDEBUG from the started SMTP procedure with DEBUG turned on
• A packet trace from TCP/IP and network trace facility output
  This documentation might be needed in cases where the actual data in the
  TCP/IP packets needs to be examined
• SMTPhlq.*.ADDRBLOK data set is a control file for SMTP processing.

  Note: You should stop SMTP in order to obtain the ADDRBLOK data set as it
  was sent, because the data set is updated during processing and deleted
  when the number of recipients equals zero.
• SMTPhlq.*.NOTE data set is the contents of the note being sent across the
  TCP/IP connection containing both headers and mail body.

**Steps for analyzing incorrect mail output**

**Before you begin:** The main goal in diagnosing an incorrect output problem is to
determine where the corruption occurs. Is the data corrupted in SMTP, TCP/IP, or
by something or someone on the network?

Perform the following steps to analyze the problem:

• If the problem is that the received mail item has incorrect output:
  1. Is the correct translation table being used or could it have been customized
to cause the error?
     Correct the translation error.
  2. Do TCP/IP and SMTP receive the correct output from the remote host?
     Obtain TCP/IP packet trace output or network trace facility output or both
to see the actual data in the packets from the remote host.
  3. Analyze the output from SMTP DEBUG for obvious errors. The body of the
     note (mail item) is not shown in this output.

• If the problem is that the sent mail item has incorrect output:
  1. Is the correct translation table being used, or could it have been customized
to cause the error?
     Correct the translation error.
  2. Was the correct data sent from SMTP or TCP/IP?
     Obtain a TCP/IP packet trace to see the actual data in the packets as they
     leave TCP/IP.
  3. Analyze the output from SMTP DEBUG for obvious errors. The body of the
     note (mail item) is not shown in this output.

If the problem cannot be corrected by this procedure, and you believe that the
problem is caused by either SMTP or TCP/IP, call the IBM Software Support
Center for further diagnosis.

**Forcing resolution of queued mail**

Normally, the SMTP server resolves the MX or A records of a piece of mail and
stores the mail in the data sets pointed to by the MAILFILEDSPREFIX keyword in
the SMTP configuration data set. If the mail cannot be delivered for some period of
time, the IP addresses in the mail can become old or obsolete. The data set names
for each piece of mail are:

`mailfiledsprefix.number.ADDRBLOK`
`mailfiledsprefix.number.NOTE`
There are two ways to force the SMTP server to resolve the addresses:

- The preferred method is to issue the SMSG SMTP EXPIRE command. Refer to z/OS Communications Server: IP User’s Guide and Commands and z/OS Communications Server: IP System Administrator’s Commands for more information about this command.
- An alternate method is to modify the ADDRBLOK data set for the piece of mail. For each recipient record (records three through the end of the data set), if the first character of the record is an S, then change the S to an E, for expired. This causes SMTP to resolve that record in the ADDRBLOK data set the next time the SMTP server is started. To modify the ADDRBLOK data set, the data set must be zapped, or a local utility program must be used. The data set cannot be modified using the ISPF editor or IEBUPDATE.

**ADDRBLOK data set**

An ADDRBLOK data set is the master control file for SMTP and is used for tracking the status of a mail item during mail delivery. One ADDRBLOK data set is allocated for each piece of mail and is built when the mail is received. The data set is allocated with a high-level qualifier of MAILFILEDSPREFIX from the SMTP.CONFIG data set. The data set is updated during mail processing and is deleted when the number of recipients equals zero.

**Guideline:** You might need to stop SMTP in order to obtain the ADDRBLOK data set as it was sent, because the data set is updated during processing and deleted when the number of recipients equals zero.

**Table 35** shows the format of Record 1 (the master control record) of an SMTP ADDRBLOK data set.

<table>
<thead>
<tr>
<th>Characters</th>
<th>Description</th>
<th>Length (in characters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–7</td>
<td>Total number of recipients</td>
<td>7</td>
</tr>
<tr>
<td>8–14</td>
<td>Number of unresolved recipients</td>
<td>7</td>
</tr>
<tr>
<td>15–21</td>
<td>Number of recipients left to send this mail item to</td>
<td>7</td>
</tr>
<tr>
<td>22</td>
<td>Unused</td>
<td>1</td>
</tr>
<tr>
<td>23–30</td>
<td>File name of note file</td>
<td>8</td>
</tr>
<tr>
<td>31</td>
<td>Unused</td>
<td>1</td>
</tr>
<tr>
<td>32–39</td>
<td>Date</td>
<td>8</td>
</tr>
<tr>
<td>40</td>
<td>Unused</td>
<td>1</td>
</tr>
<tr>
<td>41–48</td>
<td>Time</td>
<td>8</td>
</tr>
<tr>
<td>49</td>
<td>Unused</td>
<td>1</td>
</tr>
<tr>
<td>50–53</td>
<td>Unused</td>
<td>4</td>
</tr>
<tr>
<td>54–55</td>
<td>Unused</td>
<td>2</td>
</tr>
<tr>
<td>56</td>
<td>Key</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>BSMTP RPLY file</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>Spool file</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Spool file from Mailer</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>File from TCP</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Error file</td>
<td></td>
</tr>
</tbody>
</table>
Table 35. Format of Record 1 of an SMTP ADDRBLK data set (continued)

<table>
<thead>
<tr>
<th>Characters</th>
<th>Description</th>
<th>Length (in characters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>57–64</td>
<td>Tag user ID</td>
<td>8</td>
</tr>
<tr>
<td>65–72</td>
<td>Tag node ID</td>
<td>8</td>
</tr>
<tr>
<td>73–80</td>
<td>Spool ID on the current system</td>
<td>8</td>
</tr>
<tr>
<td>77–80</td>
<td>Spool ID of the file source</td>
<td>4</td>
</tr>
</tbody>
</table>

Note: Characters 57–80 are optional data used only when the key (Character 56) is “S” or “M.”

Table 36 shows the format of Record 2 (for an unresolved From record) of an SMTP ADDRBLK data set.

Table 36. Format of Record 2 (for an unresolved from record) of an SMTP ADDRBLK data set

<table>
<thead>
<tr>
<th>Characters</th>
<th>Description</th>
<th>Length (in characters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Key</td>
<td>1</td>
</tr>
<tr>
<td>U</td>
<td>Unresolved</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Sender path length (user host.domain)</td>
<td>1</td>
</tr>
<tr>
<td>3–4</td>
<td>Length of sender ID</td>
<td>2</td>
</tr>
<tr>
<td>5–(L1+4)</td>
<td>Sender ID (who sent the mail)</td>
<td>L1</td>
</tr>
<tr>
<td>(L1+5)–(L1+6)</td>
<td>Length of sender host.domain</td>
<td>2</td>
</tr>
<tr>
<td>(L1+7)–(L1+L2+6)</td>
<td>Sending host.domain</td>
<td>L2</td>
</tr>
<tr>
<td>(L1+L2+7)</td>
<td>Length of sender ID</td>
<td>1</td>
</tr>
<tr>
<td>(L1+L2+8)–(L1+L2+L3+7)</td>
<td>Sender ID (who sent the mail)</td>
<td>L3</td>
</tr>
</tbody>
</table>

Table 37 shows the format of Record 2 (for a resolved From record) of an SMTP ADDRBLK data set.

Table 37. Format of Record 2 (for a resolved from record) of an SMTP ADDRBLK data set

<table>
<thead>
<tr>
<th>Characters</th>
<th>Description</th>
<th>Length (in characters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Key</td>
<td>1</td>
</tr>
<tr>
<td>M</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Sender path length (user host.domain)</td>
<td>1</td>
</tr>
<tr>
<td>3–4</td>
<td>Length of sender ID</td>
<td>2</td>
</tr>
<tr>
<td>5–(L1+4)</td>
<td>Sender ID (who sent the mail)</td>
<td>L1</td>
</tr>
<tr>
<td>(L1+5)–(L1+6)</td>
<td>Length of sender host.domain</td>
<td>2</td>
</tr>
<tr>
<td>(L1+7)–(L1+L2+6)</td>
<td>Sending host.domain</td>
<td>(L1+L2+7)</td>
</tr>
<tr>
<td>(L1+L2+8)</td>
<td>Length of sender ID</td>
<td>1</td>
</tr>
<tr>
<td>(L1+L2+L3+9)</td>
<td>Sender ID (who sent the mail)</td>
<td>L3</td>
</tr>
<tr>
<td>(L1+L2+L3+10)</td>
<td>Length of encoded return path</td>
<td>1</td>
</tr>
<tr>
<td>(L1+L2+L3+10)–(L1+L2+L3+L4+9)</td>
<td>Encoded return path</td>
<td>L4</td>
</tr>
</tbody>
</table>
Table 38 shows the format of Records 3–n of an SMTP ADDRBLOK data set.

Table 38. Format of Record 3 (for an unresolved from record) of an SMTP ADDRBLOK data set

<table>
<thead>
<tr>
<th>Characters</th>
<th>Description</th>
<th>Length (in characters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Key Value</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>U Unresolved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>M Resolved</td>
<td></td>
</tr>
<tr>
<td>2–5</td>
<td>Time-to-Live (TTL)</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Length of return path</td>
<td>1</td>
</tr>
<tr>
<td>7–8</td>
<td>Length of recipient user ID</td>
<td>2</td>
</tr>
<tr>
<td>9–(L1+8)</td>
<td>Recipient user ID</td>
<td>L1</td>
</tr>
<tr>
<td>(L1+9)–(L1+11)</td>
<td>Length of recipient host.domain</td>
<td>2</td>
</tr>
<tr>
<td>(L1+12)–(L1+L2+11)</td>
<td>Recipient’s host.domain</td>
<td>L2</td>
</tr>
<tr>
<td>(L1+L2+12)</td>
<td>Length of recipient path</td>
<td>1</td>
</tr>
<tr>
<td>(L1+L2+13)–(L1+L2+L3+12)</td>
<td>Recipient path</td>
<td>L3</td>
</tr>
<tr>
<td>(L1+L2+L3+13)</td>
<td>Number of IP addresses</td>
<td>1</td>
</tr>
<tr>
<td>(L1+L2+L3+14)–(L1+L2+L3+17)</td>
<td>IP address 1</td>
<td>4</td>
</tr>
</tbody>
</table>

**Note:** There can be up to 16 IP addresses listed.

**SMTP RESOLVER trace**

The RESOLVER trace shows requests and responses sent to and received from name servers. It also shows if local hosts tables are used for name resolution. This trace helps you diagnose problems with host name resolution.

RESOLVER trace output from SMTP is included in the job log output from the started SMTP procedure.

Figure 63 on page 534 shows an example of RESOLVER trace output. Short descriptions of the numbered items in the trace follow the figure.
Following are short descriptions of the numbered items in the trace.

1. Address of the name server being used for name resolution. The address is pulled from the TCPIP .DATA data set.

2. Identification number of the query. This is also returned in the response and should be used to match queries to responses.

3. Bits set to determine the type of query and response. (Refer to RFC 1035.) There are 16 bits (0–15) set in the parameter field of DNS message.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Operation: 0=query, 1=response</td>
</tr>
<tr>
<td>1-4</td>
<td>Query type: 0=standard, 1=inverse</td>
</tr>
<tr>
<td>5</td>
<td>Set if the answer is authoritative</td>
</tr>
<tr>
<td>6</td>
<td>Set if the message is truncated</td>
</tr>
<tr>
<td>7</td>
<td>Set if recursion is desired</td>
</tr>
<tr>
<td>8</td>
<td>Set if recursion is available</td>
</tr>
<tr>
<td>9-11</td>
<td>Reserved</td>
</tr>
<tr>
<td>12-15</td>
<td>Response type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
</tr>
</tbody>
</table>
1 Format error in query
2 Server failure
3 Name does not exist
4 Actual question sent to the name server
5 IP address of the name server being queried
6 The response has arrived (UDP in this case)
7 Length of the record
8 Answer to the question
Chapter 18. Diagnosing z/OS UNIX sendmail and popper problems

This topic describes how to diagnose problems with z/OS UNIX sendmail, an electronic mail-transport agent and server, and with z/OS UNIX popper, a mail-delivery agent.

The following sections are in this topic:
- “Diagnostic aids for sendmail”
- “Debugging switches”
- “Additional diagnostic aids” on page 539
- “Diagnostic aids for IPv6 support” on page 541
- “Diagnostic aids for AT-TLS support” on page 542
- “Diagnostic aids for mail filter support” on page 542
- “Hints and troubleshooting sendmail message submission program (MSP) file submit.cf” on page 543
- “Diagnostic aids for popper” on page 544

Diagnostic aids for sendmail

The following sections describe various tools and techniques available for diagnosing problems with z/OS UNIX sendmail. For a comprehensive discussion of sendmail, refer to the industry-accepted publication *sendmail* by O'Reilly & Associates, Inc. (ISBN 1-56592-839-3). That publication is known throughout the industry as the *bat book*, because of the fruit bat depicted on the cover. This topic consistently refers to the *bat book* for further information.

You can also find more information about sendmail at the [http://www.sendmail.org](http://www.sendmail.org) web site.

For information about diagnosing problems with the other z/OS Communications Server mail application, Simple Mail Transfer Protocol (SMTP), see [Chapter 17, “Diagnosing Simple Mail Transfer Protocol (SMTP) problems,” on page 525](#).

Debugging switches

Table 39 shows a complete list of debugging switches in sendmail. Some of these switches create long and complex output. Each switch that is especially useful for debugging mail problems is marked “X” in the third column.

<table>
<thead>
<tr>
<th>Category</th>
<th>Bat book reference</th>
<th>Useful for mail problems</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-d0.1</td>
<td>16.6.1</td>
<td>X</td>
<td>Print version, compilation, and interface information</td>
</tr>
<tr>
<td>-d0.4</td>
<td>16.6.2</td>
<td>X</td>
<td>Our name and aliases</td>
</tr>
<tr>
<td>-d0.10</td>
<td>16.6.3</td>
<td></td>
<td>Operating System defines</td>
</tr>
<tr>
<td>-d0.12</td>
<td>16.6.4</td>
<td>X</td>
<td>Print library (libsm) defines</td>
</tr>
<tr>
<td>-d0.13</td>
<td>16.6.5</td>
<td>X</td>
<td>FFR Defines: _FFR_MILTER_PERDAEMON</td>
</tr>
<tr>
<td>Category</td>
<td>Bat book reference</td>
<td>Useful for mail problems</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>--------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>-d0.22</td>
<td>16.6.6</td>
<td></td>
<td>Dump delivery agents</td>
</tr>
<tr>
<td>-d0.40</td>
<td>16.6.7</td>
<td></td>
<td>Print network address of each interface</td>
</tr>
<tr>
<td>-d0.44</td>
<td>16.6.8</td>
<td></td>
<td>End with finis()</td>
</tr>
<tr>
<td>-d2.9</td>
<td>16.6.9</td>
<td></td>
<td>Show file descriptors with <code>dumpfd()</code></td>
</tr>
<tr>
<td>-d1.1</td>
<td>16.6.10</td>
<td></td>
<td>Trace <code>enoughspace()</code></td>
</tr>
<tr>
<td>-d1.5</td>
<td>16.6.11</td>
<td></td>
<td>Show failed mail</td>
</tr>
<tr>
<td>-d2.1</td>
<td>16.6.12</td>
<td></td>
<td>DNS name resolution</td>
</tr>
<tr>
<td>-d2.9</td>
<td>16.6.13</td>
<td></td>
<td>Call to <code>getcanonname(3)</code></td>
</tr>
<tr>
<td>-d3.1</td>
<td>16.6.14</td>
<td></td>
<td>Trace dropped local hostnames</td>
</tr>
<tr>
<td>-d3.5</td>
<td>16.6.15</td>
<td></td>
<td>Hostname being tried in <code>getcanonname(3)</code></td>
</tr>
<tr>
<td>-d3.15</td>
<td>16.6.16</td>
<td></td>
<td>Yes/no response to <code>-d8.5</code></td>
</tr>
<tr>
<td>-d3.20</td>
<td>16.6.17</td>
<td></td>
<td>Resolver debugging</td>
</tr>
<tr>
<td>-d3.30</td>
<td>16.6.18</td>
<td></td>
<td>Trace delivery</td>
</tr>
<tr>
<td>-d11.2</td>
<td>16.6.19</td>
<td>X</td>
<td>Show the user-id running as during delivery</td>
</tr>
<tr>
<td>-d12.1</td>
<td>16.6.20</td>
<td></td>
<td>Show mapping of relative host</td>
</tr>
<tr>
<td>-d13.1</td>
<td>16.6.21</td>
<td></td>
<td>Show delivery</td>
</tr>
<tr>
<td>-d20.1</td>
<td>16.6.22</td>
<td></td>
<td>Show resolving delivery agent:parseaddr()</td>
</tr>
<tr>
<td>-d21.1</td>
<td>16.6.23</td>
<td>X</td>
<td>Trace rewriting rules</td>
</tr>
<tr>
<td>-d21.2</td>
<td>16.6.24</td>
<td></td>
<td>Trace <code>$&amp;macros</code></td>
</tr>
<tr>
<td>-d22.1</td>
<td>16.6.25</td>
<td></td>
<td>Trace tokenizing an address: prescan()</td>
</tr>
<tr>
<td>-d22.11</td>
<td>16.6.26</td>
<td></td>
<td>Show address before prescan</td>
</tr>
<tr>
<td>-d22.12</td>
<td>16.6.27</td>
<td></td>
<td>Show address after prescan</td>
</tr>
<tr>
<td>-d25.1</td>
<td>16.6.28</td>
<td></td>
<td>Trace &quot;sendlist&quot;</td>
</tr>
<tr>
<td>-d26.1</td>
<td>16.6.29</td>
<td></td>
<td>Trace recipient queueing</td>
</tr>
<tr>
<td>-d27.1</td>
<td>16.6.30</td>
<td>X</td>
<td>Trace aliasing</td>
</tr>
<tr>
<td>-d27.2</td>
<td>16.6.31</td>
<td>X</td>
<td>Include file, self-reference, error on home</td>
</tr>
<tr>
<td>-d27.3</td>
<td>16.6.32</td>
<td>X</td>
<td>Forwarding path and alias wait</td>
</tr>
<tr>
<td>-d27.4</td>
<td>16.6.33</td>
<td>X</td>
<td>Print not safe</td>
</tr>
<tr>
<td>-d27.5</td>
<td>16.6.34</td>
<td>X</td>
<td>Trace aliasing with printaddr[]</td>
</tr>
<tr>
<td>-d27.8</td>
<td>16.6.35</td>
<td>X</td>
<td>Show setting up an alias map</td>
</tr>
<tr>
<td>-d27.9</td>
<td>16.6.36</td>
<td></td>
<td>Show user-id/group-id changes with:Include:reads</td>
</tr>
<tr>
<td>-d28.1</td>
<td>16.6.37</td>
<td></td>
<td>Trace user database transactions</td>
</tr>
<tr>
<td>-d29.1</td>
<td>16.6.38</td>
<td></td>
<td>Special rewrite of local recipient</td>
</tr>
<tr>
<td>-d29.4</td>
<td>16.6.39</td>
<td></td>
<td>Trace fuzzy matching</td>
</tr>
<tr>
<td>-d31.2</td>
<td>16.6.40</td>
<td></td>
<td>Trace processing of headers</td>
</tr>
<tr>
<td>-d34.1</td>
<td>16.6.41</td>
<td></td>
<td>Watch header assembly for output</td>
</tr>
<tr>
<td>-d34.11</td>
<td>16.6.42</td>
<td></td>
<td>Trace header generation and skipping</td>
</tr>
<tr>
<td>-d35.9</td>
<td>16.6.43</td>
<td></td>
<td>Macro values defined</td>
</tr>
<tr>
<td>-d37.1</td>
<td>16.6.44</td>
<td>X</td>
<td>Trace settings of options</td>
</tr>
</tbody>
</table>
### Table 39. Debugging switches by category (continued)

<table>
<thead>
<tr>
<th>Category</th>
<th>Bat book reference</th>
<th>Useful for mail problems</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-d37.8</td>
<td>16.6.45</td>
<td>X</td>
<td>Trace adding of words to a class</td>
</tr>
<tr>
<td>-d38.2</td>
<td>16.6.46</td>
<td></td>
<td>Show database map opens and failures</td>
</tr>
<tr>
<td>-d38.3</td>
<td>16.6.47</td>
<td>X</td>
<td>Show passes</td>
</tr>
<tr>
<td>-d38.4</td>
<td>16.6.48</td>
<td>X</td>
<td>Show result of database map open</td>
</tr>
<tr>
<td>-d38.9</td>
<td>16.6.49</td>
<td></td>
<td>Trace database map closing and append</td>
</tr>
</tbody>
</table>
| -d38.10  | 16.6.50           |                          | Trace NIS search for @:
| -d38.12  | 16.6.51           |                          | Trace database map stores |
| -d38.19  | 16.6.52           |                          | Trace switched map finds |
| -d38.20  | 16.6.53           |                          | Trace database map lookups |
| -d41.1   | 16.6.54           |                          | Trace queue ordering |
| -d44.4   | 16.6.55           |                          | Trace safefile() |
| -d44.5   | 16.6.56           |                          | Trace writable() |
| -d48.2   | 16.6.57           |                          | Trace calls to the check_rules set |
| -d49.1   | 16.6.58           |                          | Trace checkcompat() |
| -d52.1   | 16.6.59           |                          | Show disconnect from controlling TTY |
| -d52.100 | 16.6.60           |                          | Prevent disconnect from controlling TTY |
| -d60.1   | 16.6.61           |                          | Trace database map lookups inside rewrite() |
| -d99.100 | 16.6.62           |                          | Prevent backgrounding including the daemon |
| -d96.9   | NA                | X                        | Trace SSL (gsk_xxx) calls |

### Additional diagnostic aids

In addition to debugging switches, you can use the following z/OS UNIX sendmail diagnostic aids:

- syslog.log provides more information. The following sample shows a z/OS UNIX sendmail syslog.log message:

  Dec 28 02:13:30 MVS186 sendmail[67108947]: EZZ7514I: sendmail starting.

  Dec 28 02:13:30 MVS186 sendmail[67108947]: starting daemon (8.12.1): SMTP


- Use the -v (verbose) command-line switch to print a complete description of all the steps required to deliver a mail message. For details, refer to [sendmail, 3rd Edition](https://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.zos.ezzслушate.doc/zh-cn/zmssp_sendmail_02.htm).

- Use the -X (trace log) command-line switch to record all input, output, SMTP traffic, and other significant transactions into the specified trace file. For details, refer to [sendmail, 3rd Edition](https://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.zos.ezzслушate.doc/zh-cn/zmssp_sendmail_02.htm).

- Check the qf file for queueing concerns. z/OS UNIX sendmail stores undeliverable messages in the QueueDirectory that is specified in the configuration file. The QueueDirectory contains data files (df files) named dfxxxxxxxx and matching queue-control files (qf files) named qfxxxxxxxx. A df
file contains the body of a queued message. A qf file holds all the information that is needed to deliver the message. Each queued message has a corresponding df and qf file.

The qf file is line-oriented, containing one item of information per line. The single uppercase character (the code letter) specifies the contents of the line. The complete list of qf code letters is shown in Table 40.

<table>
<thead>
<tr>
<th>Code</th>
<th>Reference</th>
<th>Meaning</th>
<th>How Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Bat book 11.11.1</td>
<td>AUTH=parameter</td>
<td>At most, one</td>
</tr>
<tr>
<td>B</td>
<td>Bat book 11.11.2</td>
<td>Message body type</td>
<td>At most, one</td>
</tr>
<tr>
<td>C</td>
<td>Bat book 11.11.3</td>
<td>Set controlling user</td>
<td>At most, one per R line</td>
</tr>
<tr>
<td>d</td>
<td>Bat book 11.11.4</td>
<td>Data file directory</td>
<td>Exactly one</td>
</tr>
<tr>
<td>D</td>
<td>Bat book 11.11.5</td>
<td>Data file name</td>
<td>Exactly one</td>
</tr>
<tr>
<td>E</td>
<td>Bat book 11.11.6</td>
<td>Send errors to</td>
<td>Many</td>
</tr>
<tr>
<td>F</td>
<td>Bat book 11.11.7</td>
<td>Save flagged bits</td>
<td>Exactly one</td>
</tr>
<tr>
<td>H</td>
<td>Bat book 11.11.8</td>
<td>Header line</td>
<td>Many</td>
</tr>
<tr>
<td>I</td>
<td>Bat book 11.11.9</td>
<td>Mode and device information for the df file</td>
<td>Exactly one</td>
</tr>
<tr>
<td>K</td>
<td>Bat book 11.11.10</td>
<td>Time last processed</td>
<td>Exactly one</td>
</tr>
<tr>
<td>M</td>
<td>Bat book 11.11.11</td>
<td>Message (why Manyqueued)</td>
<td>At most one</td>
</tr>
<tr>
<td>N</td>
<td>Bat book 11.11.12</td>
<td>Number times tried</td>
<td>At most, one</td>
</tr>
<tr>
<td>P</td>
<td>Bat book 11.11.13</td>
<td>Priority (current)</td>
<td>At most, one</td>
</tr>
<tr>
<td>Q</td>
<td>Bat book 11.11.14</td>
<td>The DSN ORCPT address</td>
<td>At most, one per R ine</td>
</tr>
<tr>
<td>r</td>
<td>Bat book 11.11.15</td>
<td>Final recipient</td>
<td>At most, one</td>
</tr>
<tr>
<td>R</td>
<td>Bat book 11.11.16</td>
<td>Recipient address</td>
<td>Man y</td>
</tr>
<tr>
<td>S</td>
<td>Bat book 11.11.17</td>
<td>Sender address</td>
<td>Exactly one</td>
</tr>
<tr>
<td>T</td>
<td>Bat book 11.11.18</td>
<td>Time created</td>
<td>Exactly one</td>
</tr>
<tr>
<td>V</td>
<td>Bat book 11.11.19</td>
<td>Version</td>
<td>Exactly one</td>
</tr>
<tr>
<td>Z</td>
<td>Bat book 11.11.20</td>
<td>DSN envelope ID</td>
<td>At most, one</td>
</tr>
</tbody>
</table>
Table 40. qf File code letters (continued)

<table>
<thead>
<tr>
<th>Code</th>
<th>Reference</th>
<th>Meaning</th>
<th>How Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>Bat book</td>
<td>Delivery by specification</td>
<td>At most, one</td>
</tr>
<tr>
<td></td>
<td>11.11.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>Bat book</td>
<td>Restore macro value</td>
<td>At most, one</td>
</tr>
<tr>
<td></td>
<td>11.11.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>Bat book</td>
<td>End of qf file</td>
<td>Exactly one</td>
</tr>
<tr>
<td></td>
<td>11.11.23</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


**Diagnostic aids for IPv6 support**

For information about configuring an IPv6 Daemon, refer to `z/OS Communications Server: IP Configuration Guide`.

In addition, to handle network variation, the following are useful.

- **Failed to open socket.**
  When invoking sendmail, if it fails to open a socket, the following log message is displayed:
  
  `opendaemonsocket: daemon <MTA_name>: cannot create server SMTP socket``
  `opendaemonsocket: daemon <MTA_name>: problem creating SMTP socket``

  Consider the following to solve this problem:
  
  - Is the TCP/IP stack enabled for IPv4 or IPv6?
  - Is the DaemonPortOption in sendmail configuration file (sendmail.cf) properly set? (Remember that an IPv6 daemon option cannot run on a IPv4-only stack.)

- **DNS support.**
  When sendmail runs as a IPv6-enable daemon, it needs to do two things:
  
  - Receive mails with long-type address
  - Make AAAA type queries with DNS

  In some database files (for example, aliases, relay-domains, or access), if mail which is targeted to a legal IPv6 site always fails to be sent, check whether name server supports IPv6 (AAAA type queries).

  If DNS queries are failing, see “RESOLVER trace (SYSTCPRE)” on page 188 for information about how to run a resolver trace.

  To determine whether the name server is IPv6-capable, issue the following:
  
  `dig @<address_of_name_server> <host_name_of_target> aaaa`

  If this does not return an IPv6 address, either the name server is not IPv6-capable or the name server is not configured properly.

  In order to determine if the name server is IPv6-capable and the name server is a bind-based name server (not Microsoft®), issue the following:
  
  `dig @<address_of_name_server> version.bind chaos any`

  If the version of bind returned is 9.0 or greater, the name server is IPv6-capable, so it is likely not configured properly for IPv6. DNS administrators can restrict the name server from giving out its bind version, but if any type of an answer is
received other than a failed query response, the name server is IPv6-capable. If the query fails, the name server cannot support IPv6.

Diagnostic aids for AT-TLS support

Before you begin: You need to know that a packet trace can be taken to ensure that mail is encrypted before being sent. If packet traces show that encryption has occurred, but a specific packet is suspected of being unencrypted, set confLOG_LEVEL to a value greater than 9 and re-create the packet. If there were any errors in encryption, they are sent to syslog with LOG_ERR. After investigating a single packet, if you want to investigate whether SSL function calls were in error, use -d96.9 debug to check all return codes to gsk_xxx calls.

To analyze the reason individual System SSL function calls are in error, follow these steps:

1. Set the /etc/mail/zOS.cf file GskTraceFile parameter to a file name to receive the System SSL trace.

2. Rerun the command.

3. Use the System SSL gsktrace command to create a readable copy of the trace information.

When you are done, you can use this trace information to analyze reasons individual System SSL function calls might be in error. For additional information, see z/OS Cryptographic Services System SSL Programming.

Diagnostic aids for mail filter support

The debug message of a mail filter can be divided into two parts:

- Milter API
  These messages are provided to allow programmers to develop a mail filter. These messages are written into the log file defined in filter program. The following section gives more detail of these messages.

- Filter program
  The Milter API messages are mainly function error and input error. A function error means that a function call fails. It occurs when using an incompatible function or allocating invalid system resource, for example. These messages can be as follows:

  EZZ9963I filtername: malloc(size) failed for typestorage (ret reason)
  strerror(ret) {abort | try again}*

  EZZ9971I filtername: pthread_create() failed (ret reason), strerror(ret)

  These errors cannot be resolved easily. Report them to the program developer or the system administrator.

  An input error means that a user has given an invalid parameter and caused the program to terminate. The mail filter reads socket type and port number from users.
Socket type has the following types:

- `inet4` (for IPv4)
- `inet6` (for IPv6)
- UNIX domain socket

The following list describes the error operation and messages:

* **EZZ9951I SampleFilter: unknown socket type inet5**
  You gave an invalid socket type `inet5`. Select a valid socket type.

* **EZZ9961I filtername: Unable to bind (ret reason) to port string: strerror(ret)**
  The file path does not exist when using UNIX domain socket. Check that the file path exists before using UNIX domain socket.

* **EZZ9952I filtername: UNIX socket name string longer than max**
  A UNIX domain socket name cannot be defined over 108 characters in length. Rename A.B to less than 108 characters.

* **EZZ9955I SampleFilter: unknown port name abc**
  You gave an invalid port number.

* **EZZ9961I filtername: Unable to bind (ret reason) to port string: strerror(ret)**
  Do not give a port number that has been reserved, for example 21 (default for FTP). Obtain the reserved filter port number from the system administrator.

* **EZZ9965I SampleFilter: Unable to create listening socket on conn inet:21**
  Some error occurred when creating, binding, setting or listening a socket. Detailed error message should already be displayed before this message.

Sendmail daemon provides some information for connecting and talking to mail filters, you can change the log level defined in sendmail configuration file. The default log level is the same with sendmail log level:

```
O Milter.LogLevel=20
```

Check if the sendmail daemon works correctly with mail filters by log messages in sendmail's log file.

```
O Milter.LogLevel=20
```

Check if sendmail daemon works correctly with mail filters in the sendmail log file.

If mail is lost between the sendmail daemon and the filter program, see “Packet trace (SYSSTCPDA) for TCP/IP stacks” on page 94 to run a packet trace to determine where, and if, packets are being lost.

---

**Hints and troubleshooting sendmail message submission program (MSP) file submit.cf**

When feature msp is specified, FEATURE('msp'), the option conf RunAsUser is set to smmsp. This user must have the group smmspgrp, for example, the same group as the clientmqueue directory. If you specify a user whose primary group is not the same as that of the clientmqueue directory, then you should explicitly set the group as follows:

```
FEATURE('msp')
define('confRUN_AS_USER', 'mailmsp:smmspgrp')
```

The SEZASAMP(EZARACF) file shows sample commands to add the smmsp user and group.
ADDGROUP SMSPGRP OMVS(GID(25))
ADDGROUP SNDMGRP OMVS(GID(26))
ADDUSER MAILNULL DFLTGRP(SNDMGRP) NOPASSWORD OMVS(UID(26) HOME('/'))
ADDUSER SENDMAIL DFLTGRP(SNDMGRP) NOPASSWORD OMVS(UID(0) HOME('/'))
ADDUSER SMSP DFLTGRP(SMSPGRP) NOPASSWORD OMVS(UID(25) HOME('/'))

In addition, there are security concerns for programs that change user ID without prompting for a password. Program control is the Security Server facility used to manage programs that change user IDs without prompting for a password. By having an installation use program control, applications not permitted to the facility are not allowed to change user IDs without prompting for a password. The commands are:

PERMIT BXP.DAEMON CLASS(FACILITY) ID(SENDMAIL) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH

For more information on Security Server commands used to allow sendmail access to the program control facility, refer to SEZAINST(EZARACF). For complete information on the program control facility, refer to z/OS Security Server RACF Security Administrator’s Guide.

In a program control environment, use /bin/sendmail to create mail as a Mail Submission Agent (MSA) and /usr/sbin/sendmail as a Mail Transfer Agent (MTA).

If a program control environment is defined for your installation and an end user invokes sendmail and gets EZZ9895I, the installation has not configured the MSA completely.

/bin/sendmail must be owned by the same user ID as the confRUN_AS_USER (smssp uid 25 default) set in /etc/mail/submit.cf. To do this enter the following two commands:

chown 25:25 /bin/sendmail
chmod 6755 /bin/sendmail

Diagnostic aids for popper

Diagnostic aids for popper are found in the SYSLOGD log information. Following is a sample z/OS UNIX popper log message:

Apr 20 14:19:36 MVSW popper[16777240]: Received: "quit"

Use the -t trace option to direct all popper message logging to the specified file. The POP server copies the user’s entire maildrop to /tmp and then operates on that copy. If the maildrop is particularly large, or inadequate space is available in /tmp, then the server refuses to continue and terminate the connection.

To test popper, you can mimic a popper client by TELNETing into a popper port (110) and issuing the popper commands documented in RFC 1725. Following are a few of the commands used to verify that popper is listening on port 110:

user name
   Specifies the mailbox.

pass string
   Specifies a server/mailbox-specific password.

list [msg]
   Lists all message numbers and size or information about a specific message.
**retr msg**
Retrieves the specific message to the screen.

**quit**  Closes the connection to popper.

Following is an example of a TELNET exchange:

```plaintext
> telnet <host name/ip addr> 110
OK POP (version 2.53) at MVSW.tcp.raleigh.ibm.com starting.

> user user163
OK Password required for USER163

> pass tcpxyz
OK USER163 has 6 messages (4273 octets)

> list
OK 6 messages (4273 octets)
1 346
2 371
3 333
4 347
5 2541
6 335
.

> retr 3
OK 333 octets
Received: 98PXROOT@local host by mvsw.tcp.raleigh.ibm.com (8.8.7/8.8.1) id PAA83
886099 for user163; Tue, 10 Mar 1998 15:36:57 -0500
Date: Tue, 10 Mar 1998 15:36:57 -0500
from USER163 <USER163>USER163
Message-ID: <199803102036.PAA83886099@mvsw.tcp.raleigh.ibm.com>
X-UIDL: 4569e8e12631e857eed8d8b0ca493
Status: 0

hello
.
```
Chapter 19. Diagnosing SNALINK LU0 problems

The TCP/IP host is implemented with the SNALINK LU0 function. This function allows the use of an SNA backbone to transfer TCP/IP protocols. A TCP/IP host with SNALINK LU0 can be an originator, destination, or router for TCP/IP data. To use the SNALINK LU0 function of TCP/IP, each connected host must have VTAM and TCP/IP installed. The SNALINK LU0 application runs in its own address space and is defined as a VTAM application. There are two types of SNALINK implementations:

- SNALINK LU0, which uses VTAM LU0 protocol
- SNALINK LU6.2, which uses VTAM LU6.2 protocol

This topic describes how to diagnosis problems with the SNALINK LU0 function and contains the following sections:

- "Definitions"
- "Problem diagnosis"
- "Traces" on page 551

SNALINK LU6.2 diagnosis is discussed in Chapter 20, "Diagnosing SNALINK LU6.2 problems," on page 555.

SNALINK LU0 is a very convenient way to connect to TCP/IP hosts using an existing SNA backbone. An IP datagram destined for a remote host that is connected using SNALINK LU0 is passed to the SNALINK LU0 address space by TCP/IP. The data is packaged into an SDLC frame and transmitted to the remote host using SNA LU0 protocol. Two SNALINK LU0 applications can be configured to connect using a single, bidirectional session or with two separate sessions (one dedicated to send data in each direction).

Definitions

The following are required to define a SNALINK LU0:

- Device and link definitions in the TCPIP profile
- Home address and routing information
- VTAM application definitions
- Parameters on the PROC used to start SNALINK LU0

For more information about these required definitions, refer to the z/OS Communications Server: IP Configuration Reference.

Problem diagnosis

SNALINK LU0 problems are normally reported as one of the following:

- Abends
- Session hung terminals
- Session outages

Use the information in the following sections for problem determination and diagnosis of errors reported against SNALINK LU0.
When contacting the IBM Software Support Center for any type of SNALINK LU0 problem, have the VTAM application definitions for SNALINK LU0 and the DEVICE and LINK information from the hlq.PROFILE.TCPIP data set for SNALINK LU0.

**Abends**

An abend for the SNALINK LU0 application should result in messages or error-related information on the MVS system console. Since SNALINK LU0 is a VTAM application, some abends might be generated or first detected by VTAM. These messages indicate that VTAM is abending or a dump is being taken for the SNALINK LU0 application.

In the case of a VTAM error caused by SNALINK LU0, refer to **z/OS Communications Server: SNA Messages** and **z/OS Communications Server: IP and SNA Codes** for initial problem determination.

If SNALINK LU0 fails to initialize with an 0C4 abend, there is probably an installation problem. Check the program properties table (PPT) entries for errors. Some levels of MVS do not flag PPT syntax errors properly. For more information about PPT configuration, refer to **z/OS MVS Initialization and Tuning Reference**.

**Documentation**

Code a SYSMDUMP DD or SYSABEND DD statement in the SNALINK cataloged procedure.

There are two MVS abends commonly seen during the initialization and startup of the SNALINK LU0 application: X'0C2' and X'0F8'. Both can be caused by the SNALINK LU0 application processing in TCB mode. The VTAM application definition statement for SNALINK LU0 must have the SRBEXIT=YES parameter coded. This should ensure that VTAM passes control to SNALINK LU0 in SRB mode. SNALINK LU0 code has processing that is not allowed in TCB mode. If the SRBEXIT parameter is coded incorrectly or allowed to default, either abend X'0C2' or X'0F8' will occur.

**Guideline:** Some networking optimizing packages change the defined mode for VTAM applications for performance purposes. It is suggested that this type of program not be used for the SNALINK LU0 application.

**Analysis**

For more information about debugging abends, refer to Chapter 3, “Diagnosing abends, loops, and hangs,” on page 25.

An abend or unexpected termination of the SNALINK LU0 application does not terminate the TCP/IP address space. If there is no alternate route to the remote host, IP datagrams for TCP/IP Services components (such as TELNET and FTP) are not transmitted until the application is restarted, either manually or using TCP/IP autolog.

**Session hangs**

This section discusses diagnosis of a hung terminal after a session has been successfully connected. A hang might be detected by TCP/IP users who are connected to the remote system by means of SNALINK LU0 (this could be FTP, TELNET, or other applications).
The SNALINK LU0 application detects a hung terminal if there is no response to data sent. After waiting 30 seconds for a response, SNALINK LU0 ends the session and tries to reestablish the LU-to-LU session with its partner SNALINK LU0 application. This processing is shown on the SNALINK LU0 log or MVS console log.

**Documentation**

To determine the cause of an SNALINK LU0 session hung terminal, the following might be needed:

- SNALINK LU0 log or MVS console log
- NETSTAT DEVLINKS display output
- VTAM display application status output
- SNALINK LU0 DEBUG trace output
- VTAM buffer trace of the SNALINK LU0 applications
- VTAM internal trace


This list of documentation includes documentation needed to resolve most types of hung terminals. All of the indicated data might not be needed for each occurrence of a hung terminal. The following section provides information on the types of data that might be needed for each diagnostic step.

**Steps for analyzing session hangs**

*Before you begin:* The first step in analysis is to determine if the SNALINK LU0 is actually hung or if one of the sessions using SNALINK LU0 to transfer data is hung. When the SNALINK LU0 is the only connection between two hosts, an actual hang in the SNALINK LU0 application impacts all data flowing for TCP/IP. This can include TELNET, FTP, and any other application.

Perform the following steps to determine the cause of the reported SNALINK LU0 hung terminal:

1. **Does all traffic across the SNALINK LU0 stop?** A VTAM buffer trace of the SNALINK LU0 application can be used to see if any data is being passed. If data is still flowing on the session, the SNALINK LU0 is not hung. You need to determine which TCP/IP application or component is failing. If there is no data traffic, continue with Step 2.

   You can also check SNALINK LU0 traffic by doing multiple VTAM displays of the SNALINK LU0 application. The SEND and RECEIVE data count should increase for an active session. Often, using the VTAM display to obtain the status of the TRLE might provide useful information.

2. **Issue NETSTAT DEVLINKS to determine the status of the SNALINK LU0 TCP/IP device.** If the NETSTAT output shows that the application is trying to connect, check the VTAM and SNALINK LU0 consoles for information about a previous error or abend. If NETSTAT indicates “negotiating,” verify the session type. You might require a session_type of SINGLE; refer to [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com/support/knowledgecenter/en/SS5W8A_9.1.1/com.ibm.zos.v9r1.hac_smp0573.pdf) for information on configuring session types. If NETSTAT indicates “connected” or “sending,” continue with Step 3.
3. At this point, you should determine the last SNALINK LU0 activity or processing. This is best accomplished with the debug trace. Contact your IBM Software Support Center with information about the last activity from the SNALINK LU0 console and debug trace.

Information on starting and examining the trace data is discussed in "Starting SNALINK LU0 DEBUG trace" on page 551.

---

Session outages

A session outage is an unexpected abend or termination of the task. Session outages are usually seen only when an unrecoverable error is detected. The error could be a SNALINK LU0 abend or an error return code from a VTAM request. A session outage should not occur without an indication of its cause, either on the SNALINK LU0 or the VTAM console. Since SNALINK LU0 abends have already been discussed separately, this section describes other types of session outages.

For an example of a successful session setup between two SNALINK LU0 applications, refer to the z/OS Communications Server: IP Configuration Reference.

Documentation

The following documentation might be needed to determine the source of the error for a session outage problem:

- SNALINK LU0 log
- MVS console log
- VTAM log
- NETSTAT DEVLINKS display output
- VTAM display application status output
- SNALINK LU0 DEBUG trace output
- VTAM buffer trace of the SNALINK
- LU0 applications
- VTAM Internal Trace (VIT)

Note: For information on VTAM traces, refer to z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures and z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT.

Analysis

When a SNALINK LU0 outage occurs, there should be messages and indicators of the reason for the outage. These appear in the SNALINK LU0 log, or on the VTAM console, or both. If an abend has been recorded, continue diagnosis using the section on abends.

The following is an example of a session outage problem. The message EZA5797E Rejecting bind from xxxxx-no DLC found, along with VTAM error message IST663I Bind fail request received, SENSE=800A0000, was displayed on the MVS system console.

Cause: Large packet size sent in a PIU is rejected by the NCP with sense 800A0000 (PIU too long).

Resolution: Reduce the MTU size on this route using the GATEWAY statement.
Traces

The following are useful:

- Use VTAM buffer trace to trace the data sent and received from the VTAM.
- Use the TCPIP PKTTRACE LINKNAME=link_name to trace the data sent and received from TCP/IP.

Using IP packet trace

The IP packet trace facility is used to trace the flow of IP packets. It is useful when tracking the cause of packet loss or corruption. If the LINKNAME parameter of the IP packet trace facility is specified, only packets transferred along the given link are traced. Specifying this parameter is recommended to avoid tracing a large number of unrelated packets. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47 for details about how to use the IP packet trace facility.

SNALINK LU0 DEBUG trace

The SNALINK LU0 DEBUG trace output is written to an internal buffer. The trace can be seen only if a dump of the SNALINK LU0 address space is taken. The trace wraps when the buffer is full (a pointer in the trace header points to the most current entry).

The trace contains information on SNALINK LU0 processing. This includes communication with VTAM and TCP/IP, showing VTAM macro requests and DLC requests.

Starting SNALINK LU0 DEBUG trace

To run the SNALINK LU0 DEBUG trace, SNALINK LU0 must be started with DEBUG listed as the first parameter of the PARM parameter on the EXEC statement of the SNALINK cataloged procedure. For information about this parameter, refer to z/OS Communications Server: IP Configuration Reference.

DEBUG trace example

Figure 64 on page 552 shows part of an internal SNALINK LU0 trace obtained from a dump. As shown in the example, the trace can be located by searching for the characters TRCTBL in the dump of the SNALINK LU0 address space. Following the eyecatcher is the address of the next entry to be written, the starting address of the trace table, and the ending address of the trace table.

Use the information following the trace to interpret the entry types and their meaning.
The layout of a SNALINK trace table entry is shown in Table 41.

Table 41. Format of a SNALINK trace table entry

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>00–07</td>
<td>TOD time stamp</td>
</tr>
<tr>
<td>08–0F</td>
<td>LU name, if any</td>
</tr>
</tbody>
</table>

Figure 64. Example of a SNALINK LU0 DEBUG trace

The layout of a SNALINK trace table entry is shown in Table 41.
Table 41. Format of a SNALINK trace table entry (continued)

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Entry Type</td>
</tr>
<tr>
<td></td>
<td><strong>Value</strong></td>
</tr>
<tr>
<td></td>
<td>01 DLC Accept</td>
</tr>
<tr>
<td></td>
<td>02 DLC Send</td>
</tr>
<tr>
<td></td>
<td>03 DLC Receive</td>
</tr>
<tr>
<td></td>
<td>04 DLC Sever</td>
</tr>
<tr>
<td></td>
<td>05 DLC Msg Pend Queue Request</td>
</tr>
<tr>
<td></td>
<td>06 DLC Msg Pend D-Queue Request</td>
</tr>
<tr>
<td></td>
<td>0E MVS DLC emulation</td>
</tr>
<tr>
<td></td>
<td>0F DLC Interrupt</td>
</tr>
<tr>
<td></td>
<td>10 VTAM Request</td>
</tr>
<tr>
<td></td>
<td>17 VTAM OPNDS Exit</td>
</tr>
<tr>
<td></td>
<td>1F VTAM CLSDST Exit</td>
</tr>
<tr>
<td></td>
<td>22 VTAM SEND Exit</td>
</tr>
<tr>
<td></td>
<td>23 VTAM Receive Exit</td>
</tr>
<tr>
<td></td>
<td>25 VTAM SESSIONC Exit</td>
</tr>
<tr>
<td></td>
<td>2A VTAM OPNSEC Exit</td>
</tr>
<tr>
<td></td>
<td>2C VTAM TERMSESS Exit</td>
</tr>
<tr>
<td></td>
<td>31 VTAM SCIP Exit</td>
</tr>
<tr>
<td></td>
<td>32 VTAM LOSTERM Exit</td>
</tr>
<tr>
<td></td>
<td>33 VTAM NSEXIT Exit</td>
</tr>
<tr>
<td></td>
<td>34 VTAM TPEND Exit</td>
</tr>
<tr>
<td></td>
<td>35 VTAM LOGON Exit</td>
</tr>
<tr>
<td></td>
<td>40 SNALINK Internal Message Routine Call</td>
</tr>
<tr>
<td>11</td>
<td>DLC Interrupt Code/VTAM RPL REQ Code/ VTAM Receive Exit</td>
</tr>
<tr>
<td></td>
<td>Chain field</td>
</tr>
<tr>
<td>12</td>
<td>VTAM CMD: R15/VTAM Exit: RTNCD</td>
</tr>
<tr>
<td>13</td>
<td>VTAM CMD: R0 /VTAM Exit: FDB2/DLC IPRCODE</td>
</tr>
<tr>
<td>14-17</td>
<td>RPL Address/DLC MSG ID/TPEND reason code/Internal Message ID</td>
</tr>
<tr>
<td>18–1B</td>
<td>VTAM Send/Receive/DLC buffer address/Number of Arguments Passed to Internal Message routine</td>
</tr>
<tr>
<td>1C–1F</td>
<td>VTAM Send/Receive/DLC buffer length/Internal Message Routine caller’s return address</td>
</tr>
</tbody>
</table>
Chapter 20. Diagnosing SNALINK LU6.2 problems

This topic describes how to diagnose problems with the SNALINK LU6.2 function and contains the following sections:

- “Steps for setting up a SNALINK LU6.2 network” on page 556
- “Common configuration mistakes” on page 557
- “Diagnosing problems” on page 558
- “Documentation references for problem diagnosis” on page 568
- “Traces” on page 573
- “Finding abend and sense code documentation” on page 578
- “Finding error message documentation” on page 578

The SNALINK LU6.2 interface uses the LU type 6.2 protocol to establish a point-to-point connection across a SNA network. SNALINK LU6.2 is capable of establishing a connection with any system that runs TCP/IP and uses the LU type 6.2 protocol.

The SNALINK LU6.2 interface is similar to the SNALINK LU0 and X.25 NPSI interfaces with the connection involving several subsystems. The components of the SNALINK LU6.2 network are shown in Figure 65.

Following is a brief description of the component interaction and data flow that occurs when data is transferred over a SNALINK LU6.2 network. Each component is cross-referenced to the figure.

1. Data is generated and encapsulated on the TCP/IP address space and is passed to the SNALINK LU6.2 address space through a DLC connection.
2. The SNALINK LU6.2 address space handles all establishment, aging, and termination of SNA network connections in a manner transparent to the TCP/IP address space. The data is then sent to the local system SNA subsystem. In the case of MVS hosts, this subsystem is VTAM.
3. VTAM APPC routines are used to pass the data to the SNA network.

Figure 65. Components of a SNALINK LU6.2 connection on MVS
VTAM routines on the destination system receive the data and pass it through to the SNALINK LU6.2 address space.

The SNALINK LU6.2 address space sends the data to the TCP/IP address space using a DLC connection.

The data is unencapsulated and processed by the TCP/IP address space.

Steps for setting up a SNALINK LU6.2 network

Complete the following steps to establish the system described in Figure 65 on page 555.

This list of steps can be used to diagnose problems in starting components by identifying the prerequisites. For details about how to complete the steps, refer to the appropriate documentation.

1. Configure the SNALINK LU6.2 network on both the local and remote network hosts. This is fully described in the z/OS Communications Server: IP Configuration Reference in the section about configuring and operating the SNALINK LU6.2 interface. The process can be condensed into the following steps:
   a. Specify SNALINK LU6.2 DEVICE and LINK statements in the hlq.PROFILE.TCPIP data set.
   b. Copy the sample SNALINK LU6.2 cataloged procedure to an authorized data set and update according to your system.
   c. Define a SNALINK LU6.2 application LU to VTAM.
   d. Customize a SNALINK LU6.2 configuration data set.

2. Vary the SNALINK LU6.2 VTAM application LUs active on both the local and remote network hosts.

3. Start both the local and remote network TCP/IP address spaces.

4. Start both the local and remote network SNALINK LU6.2 address spaces, if they have not been autologged by the TCP/IP address space.

5. Verify that the network connection has been established between the local host and the remote host. See “Using the SNALINK LU6.2 subcommand” on page 569 for details about how to verify SNALINK LU6.2 connections.

The example in Figure 66 on page 557 shows the messages that are expected when the SNALINK LU6.2 address space is started and a network connection is established.
The following list explains the MVS system console messages on SNALINK LU6.2 address space startup as shown in Figure 66.

1. The SNAL621A address space has been started.
2. The SNALINK LU6.2 configuration data set for the SNAL621A address space has been successfully parsed.
3. The SNAL621A address space displays its local VTAM application LU and the TCP/IP address space name to which it connects.
4. The SNAL621A address space establishes a network connection through the VTAM API.
5. The SNAL621A address space establishes a DLC connection with its TCP/IP address space.

Common configuration mistakes

Following is a list of common configuration mistakes:

- The SNALINK LU6.2 configuration data set contains a syntax error.
- The SYSTCPD or LU62CFG ddnames in the SNALINK LU6.2 cataloged procedure have been assigned to a data set that is not valid.
- The SNALINK LU6.2 VTAM application LU has not been activated.
- The SNALINK LU6.2 VTAM application LU definition has the option SRBEXIT=YES.
- The SNALINK LU6.2 VTAM application LU definition does not have the option APPC=YES.
- The SNALINK LU6.2 VTAM application LU definition specifies a logon mode table in the MODETAB parameter that does not contain the log mode entry specified in the LOGMODE parameter on the LINK statement in the SNALINK LU6.2 configuration data set. The logon mode entry options used for the local host must be the same as for the remote host.
- The hlq.PROFILE.LINK data set contains syntax errors in the SNALINK LU6.2 DEVICE, LINK, HOME, GATEWAY, or START statements.
- The maximum buffer size in the SNALINK LU6.2 configuration data set does not match the maximum packet size in the GATEWAY statement of the hlq.PROFILE.TCPIP data set.
- The link name in the SNALINK LU6.2 configuration data set does not match the link name on the LINK statement in the hlq.PROFILE.TCPIP data set.
- The SNALINK LU6.2 device has not been started by a START statement in the hlq.PROFILE.TCPIP data set.
- The user ID assigned to the SNALINK LU6.2 start procedure has not had an OMVS Segment assigned to it using RACF or similar security manager.
Diagnosing problems

SNALINK LU6.2 problems are normally reported under one of the following categories:

- Problems starting the SNALINK LU6.2 address space
- DLC connection
- Network connection establishment
- Network connection loss
- Data loss
- Data corruption

Use the information in the following sections to help you diagnose SNALINK LU6.2 problems.

Quick checklist for common problems

The following list summarizes some initial checks that can be made quickly.

Use the following checklist to identify problem areas:

__ 1. Is the TCP/IP SNALINK LU6.2 network active?
   PING the remote TCP/IP host from the local TCP/IP host to verify that the SNALINK LU6.2 network is active. If the SNALINK LU6.2 network is not active, continue through this list to identify the problem.
   If the PING still fails after working through this list, see “Network connection establishment problems” on page 563 for a detailed list of network connection problems and their solutions.

__ 2. Have you completed all the required definitions?
   See “Steps for setting up a SNALINK LU6.2 network” on page 556 for the list of definitions and configurations required. Continue through this list if connection problems persist.

__ 3. Have the VTAM major node and application LU used by the SNALINK LU6.2 address space been varied active?
   See “Useful VTAM operations” on page 570 for details on how to use the VTAM DISPLAY command to identify the status of the VTAM major node and application LU.
   If the VTAM application LU is not in a CONCT state, see “Useful VTAM operations” on page 570 for details about how to vary the VTAM application LU active.

__ 4. Are the TCP/IP and SNALINK LU6.2 devices started and active on the local and remote host?
   Check to see if the TCP/IP and SNALINK LU6.2 devices are active and running. The MVS SDSF facility can be used to view the active address space list for MVS hosts.
   If the SNALINK LU6.2 address space does not start, see “Problems starting the SNALINK LU6.2 address space” on page 559 for a detailed list of startup problems and their solutions.

__ 5. Did the SNALINK LU6.2 address space list any configuration errors to the SYSPRINT data set?
   Use the JCL DD statement in the SNALINK LU6.2 cataloged procedure to identify the destination of the SYSPRINT output and check for errors. If errors occur, see “Finding error message documentation” on page 578.
determine the reason for the configuration errors. Text in the message
documentation specifies the action required to fix the problem.

6. Have the TCP/IP-to-SNALINK LU6.2 DLC connections been established?

See “Using NETSTAT” on page 569 for details about how to use the
NETSTAT command to identify the status of the DLC connection.
If the status of the DLC connection is not “Connected,” see “DLC connection
problems” on page 561 for a detailed list of SNALINK LU6.2 DLC
connection problems and their solutions.

7. Does the MVS system console contain VTAM error messages?

Refer to z/OS Communications Server: SNA Messages and z/OS Communications
Server: IP and SNA Codes for detailed descriptions of the VTAM error
messages and sense codes. These messages might indicate a network
configuration or hardware error.

Problems starting the SNALINK LU6.2 address space

Generally, if there is a startup problem, error messages are displayed on the MVS
system console during the starting of the SNALINK LU6.2 address space. The
address space then terminates.

Documentation

To isolate a SNALINK LU6.2 address space starting problem, note any error
messages or abend codes that are displayed on the MVS system console.

Analysis

Table 42 shows some of the common SNALINK LU6.2 address space startup
problems.

<table>
<thead>
<tr>
<th>If this is displayed. . .</th>
<th>Then this might have occurred . .</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The message Errors Detected - Address Space will Terminate has been displayed on the MVS system console with no other error messages.</td>
<td>This error message indicates that an error has occurred with the SNALINK LU6.2 configuration data set</td>
<td>Check the SNALINK LU6.2 SYSPRINT output for messages that tell what kind of syntax error might have occurred. If a syntax error has occurred in the configuration data set, correct it and restart the SNALINK LU6.2 address space. Refer to the z/OS Communications Server: IP Configuration Reference for details about the SNALINK LU6.2 configuration data set statement syntax.</td>
</tr>
<tr>
<td>The message Error in open of LU62CFG - no data will be read has been displayed on the MVS system console.</td>
<td>The SNALINK LU6.2 address space cannot access a SNALINK LU6.2 configuration data set. The LU62CFG ddname might have been omitted from the SNALINK LU6.2 cataloged procedure.</td>
<td>Check the SNALINK LU6.2 cataloged procedure. Ensure that the LU62CFG ddname is assigned a valid SNALINK LU6.2 configuration data set. Refer to the z/OS Communications Server: IP Configuration Reference for an example of a SNALINK LU6.2 cataloged procedure.</td>
</tr>
<tr>
<td>If this is displayed...</td>
<td>Then this might have occurred...</td>
<td>Resolution</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>The message Address Space Already Active - this Address Space will Terminate has been displayed on the MVS system console.</td>
<td>An address space with the same name as the SNALINK LU6.2 address space is already active.</td>
<td>Check to see if the address space with the same name is no longer required before stopping it, or rename the SNALINK LU6.2 address space. Restart the SNALINK LU6.2 address space.</td>
</tr>
</tbody>
</table>
| The messages Error 0000005A in VTAM OPEN and Errors detected in VTAM Initialization - Address Space will terminate have been displayed on the MVS system console. | The SNALINK LU6.2 address space has not been able to find the VTAM application LU that has been defined in the VTAM statement of the SNALINK LU6.2 configuration data set. | This problem might be resolved by one or both of the following solutions:  
  - Check the status of the SNALINK LU6.2 VTAM application LU and its VTAM major node. If it is not in a CONCT state, the VTAM major node and then the VTAM application LU must be activated. See “Useful VTAM operations” on page 570 for a detailed description of the VTAM operations that display the status of VTAM application LUs and activate them.  
  - Check the VTAM application LU specified in the VTAM statement of the SNALINK LU6.2 configuration data set. Ensure that it exists and is not duplicated within the domain in which the SNALINK LU6.2 application program resides. Refer to the z/OS Communications Server: IP Configuration Reference for details about the SNALINK LU6.2 VTAM statement syntax and the SNALINK LU6.2 VTAM application LU definition. |
| The messages Error 00000024 in VTAM OPEN and Errors detected in VTAM Initialization - Address Space will terminate have been displayed on the MVS system console. | VTAM security is not allowing the SNALINK LU6.2 address space to access the VTAM application LU. | Check to see if the SNALINK LU6.2 configuration data set VTAM statement password matches the password set in the VTAM application LU definition and correct it, if necessary.  
Refer to the z/OS Communications Server: IP Configuration Reference for details about the SNALINK LU6.2 VTAM statement syntax and the SNALINK LU6.2 VTAM application LU definition. |
| The SNALINK LU6.2 address space abends with a system abend code of 300 after the SNALINK LU6.2 address space STARTED message. | The abend code of 300 indicates that there is insufficient storage for the SNALINK LU6.2 address space. | Either increase the value of the REGION parameter for the address space or reduce the number of buffers specified in the SNALINK LU6.2 configuration data set. Refer to z/OS Communications Server: SNA Messages and z/OS Communications Server: IP and SNA Codes for detailed SNALU6.2 abend code descriptions. |
Table 42. Common SNALINK LU6.2 address space startup problems (continued)

<table>
<thead>
<tr>
<th>If this is displayed. . .</th>
<th>Then this might have occurred . . .</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The SNALINK LU6.2 address space abends with an abend code of S0F8 after the Initialization Complete... message.</td>
<td>The MVS S0F8 abend code indicates that an SVC was issued in SRB mode. SNALINK LU6.2 is not designed to run with VTAM in SRB mode.</td>
<td>The SRBEXIT option in the VTAM application LU definition has been set to “Yes.” Correct the VTAM application LU definition. Refer to the z/OS Communications Server: IP Configuration Reference for details about the SNALINK LU6.2 VTAM application LU definition.</td>
</tr>
</tbody>
</table>

Table 42. Common SNALINK LU6.2 address space startup problems (continued)

If, after investigation, you do not find the SNALINK LU6.2 startup problem, obtain a description of all abend codes and errors written to the SYSPRINT data set and MVS system console. Most solutions to SNALINK LU6.2 address space starting problems can be solved by reading the error message or abend code descriptions.

See “Finding abend and sense code documentation” on page 578 and “Finding error message documentation” on page 58 for a list of references that contain SNALINK LU6.2 error message and abend code documentation.

DLC connection problems

These problems are related to the TCP/IP DLC connection between the TCP/IP address space and the SNALINK LU6.2 address space.

The DLC connection between the TCP/IP and SNALINK LU6.2 address spaces is established during the SNALINK LU6.2 address space startup after the SNALINK LU6.2 configuration data set has been parsed. This DLC connection can be established independently of the SNA LU type 6.2 connection between two or more SNALINK LU6.2 address spaces. The fundamental requirements of the DLC connection are an active, configured SNALINK LU6.2 address space and an active, configured TCP/IP address space. The DLC connection is initiated by a START statement in hlq.PROFILE.TCPIP.

Steps for checking DLC connection status

Perform the following steps to check the status of the DLC connection.

1. Note the SNALINK LU6.2 address space startup messages displayed on the MVS system console.

2. Issue a NETSTAT DEVLINKS command to obtain the status of the DLC connection.

See “Using NETSTAT” on page 569 for details about how to use the NETSTAT command to identify the status of the DLC connection.

If the DLC connection status is not Connected, check the list of common DLC connection problems in the next section.

Analysis

Table 43 on page 562 lists some of the common DLC connection problems between the SNALINK LU6.2 address space and the TCP/IP address space.
<table>
<thead>
<tr>
<th>If this is displayed...</th>
<th>Then this might have occurred...</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The message Error in DLC connect... has been displayed on the MVS system console and the NETSTAT DEVLINKS output shows that the DLC connection status is either Issued Connect or Will retry connect.</td>
<td></td>
<td>This problem can be due to one of the following situations:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. The SNALINK LU6.2 address space might be rejecting the connect attempt from the TCP/IP address space because it has the wrong TCP/IP ID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. The SNALINK LU6.2 address space might be rejecting the connect attempt from the TCP/IP address space because of a SNALINK LU6.2 link name that is incorrectly defined.</td>
</tr>
</tbody>
</table>

- Check the SNALINK LU6.2 SYSPRINT output for the “Rejecting DLC path for the link_name, wrong TCP/IP id tcpip_addr_space” error message. If this error message is displayed, check whether a valid TCPIP.DATA data set was specified as the SYSTCPD ddbname in the SNALINK LU6.2 cataloged procedure and correct it, if necessary.

**Note:** SYSTCPD can be overridden by the global TCPIP.DATA file.

Refer to the [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.2.7/com.ibm.zos.v2r11.srvcoms.doc/u9600100.html) for an example of a SNALINK LU6.2 cataloged procedure and for the search order for the TCPIP.DATA data set.

If a valid TCPIP.DATA data set has been used, check the TCP/IP address space specified in the TCPIPJOBNAME statement within it.

Refer to the [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.2.7/com.ibm.zos.v2r11.srvcoms.doc/u9600100.html) for a detailed description of the TCPIPJOBNAME statement in the TCPIP.DATA.

- Check the SNALINK LU6.2 SYSPRINT output for the “Rejecting DLC path for link_name, not configured” error message.

If this error message is displayed, check to see if the link name specified in the LINK statement of the SNALINK LU6.2 configuration data set matches the link name specified in the LINK statement associated with the SNALINK LU6.2 device defined in hlq.PROFILE.TCPIP.

Refer to the [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.2.7/com.ibm.zos.v2r11.srvcoms.doc/u9600100.html) for details about the SNALINK LU6.2 LINK statement syntax and the TCPIP LINK statement syntax.
Table 43. Common DLC connection problems (continued)

<table>
<thead>
<tr>
<th>If this is displayed. . .</th>
<th>Then this might have occurred . . .</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The SNALINK LU6.2 address space has been started but the Link opened message has not been displayed and the NETSTAT DEVLINKS output shows that the DLC connection is Inactive.</td>
<td>The DLC connection to the SNALINK LU6.2 device associated with the SNALINK LU6.2 address space might not have been started by the TCP/IP address space.</td>
<td>Check the START statements in hlq.PROFILE.TCPIP.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the SNALINK LU6.2 device has not been started, use the VARY TCPIP,procname,START,device_name for the SNALINK LU6.2 device or include the START statement in the hlq.PROFILE.TCPIP and restart the TCP/IP address space.</td>
</tr>
</tbody>
</table>

Refer to the z/OS Communications Server: IP Configuration Reference for a detailed description of the START statement in the hlq.PROFILE.TCPIP.

Network connection establishment problems

These problems are related to the establishment of the SNA LU type 6.2 connection between two or more SNALINK LU6.2 devices.

The SNA LU type 6.2 connection can be established independently of the TCP/IP address space and the DLC link. The fundamental requirements for establishing the LU type 6.2 connection are two active, configured SNALINK LU6.2 devices that have an active SNA network connection between them.

Initiate the establishment of a network connection in one of the following ways:

- Connections with the INIT parameter specified on the DEST statement in the SNALINK LU6.2 configuration data set are established when the SNALINK LU6.2 address space is started.
- Connections with the DATA parameter specified on the DEST statement in the SNALINK LU6.2 configuration data set or connections that have timed out or been terminated are established when a request is made to the SNALINK LU6.2 address space to transfer data across the link.
- Connections can be established using the SNALINK LU6.2 RESTART MODIFY subcommand.

Steps for checking network connection problems

To check the status of the LU type 6.2 connection, issue the following MODIFY subcommands to the MVS SNALINK LU6.2 address space.)

1. **MODIFY addr_sp_name,LIST,LU=dest_lu_name**
   
   where addr_sp_name is the MVS SNALINK LU6.2 address space name and dest_lu_name is the SNA destination LU name of the remote SNALINK LU6.2 device.
   
   See "Using the SNALINK LU6.2 subcommand" on page 569 for more information about issuing this command and reading the output.
   
   If the connection status is not “Allocated,” continue with the following commands.

2. **MODIFY addr_sp_name,RESTART,LU=dest_lu_name**
This command attempts to establish the LU type 6.2 connection between the SNALINK LU6.2 devices. During connection establishment, any problems causes error messages to be output to the MVS system console.

3. **MODIFY addr_sp_name,LIST,LU=dest_lu_name**

If the connection status is still not “Allocated,” note the messages in the SYSPRINT data set and on the MVS system console and continue with the following analysis.

**Analysis**

Table 44 lists some of the common SNALINK LU6.2 address space network establishment problems.

<table>
<thead>
<tr>
<th>If this is displayed...</th>
<th>Then this might have occurred...</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The SNALINK LU6.2 address space issued error message: Unable to allocate send conversation.</td>
<td>This problem can be due to one of the following situations: 1. The local VTAM application LU might not be enabled for LU type 6.2 conversations. The name of this LU is specified on the VTAM statement in the SNALINK LU6.2 configuration data set. 2. The remote VTAM application LU names might not identify an LU that is reachable or that can establish an LU type 6.2 conversation over the SNA network. The remote VTAM application LU name is specified in the DEST statement of the SNALINK LU6.2 configuration data set. For dependent LUs, both the SEND and RECV LU names must be able to establish LU type 6.2 conversations.</td>
<td>1. The APPC option in the VTAM application LU definition must be set to YES to enable LU type 6.2 conversations. 2. The first step is to check to see if the remote SNALINK LU6.2 device is active. If the remote SNALINK LU6.2 is using VTAM to access the SNA network, see “Useful VTAM operations” on page 570 to check the active status of the remote LU. If the remote SNALINK LU6.2 device is active, use the VTAM error messages to determine why the LU type 6.2 conversation cannot be established with the destination LU. The VTAM error messages are written to the MVS system console immediately before the Unable to allocate send conversation message. VTAM sense code documentation can be found in z/OS Communications Server: SNA Messages and z/OS Communications Server: IP and SNA Codes. These messages might indicate a network configuration or hardware error.</td>
</tr>
</tbody>
</table>

| VTAM error message output to the MVS system console: REQUIRED LOGMODE NAME UNDEFINED. | To allocate LU type 6.2 conversations over an SNA network, both sides must specify matching log modes. The VTAM log modes are defined in log mode tables. The log mode configured for use with this connection cannot be found in the log mode table specified on the VTAM application LU definition. | The log mode entry name specified as the LOGMODE parameter on the LINK statement in the SNALINK LU6.2 configuration file must exist in the log mode table specified on the MODETAB statement in the VTAM application LU definition. |
The following list contains some of the common SNALINK LU6.2 address space network establishment problems. Each error symptom is listed with possible causes and resolutions.

**Network connection loss problems**

SNA network connection loss can be either expected or unexpected. This section deals with unexpected connection problems. The definitions of expected and unexpected losses are discussed before continuing with the analysis for unexpected loss.

Connections for the SNALINK LU6.2 address space can be configured to be normally active or normally inactive. The normally inactive configuration is used when there is a cost involved with the network connection time. Normally inactive connections are expected to experience connection establishment and loss regularly with use. Because of this, the SNALINK LU6.2 address space does not write messages to the MVS system console for connection loss. Connection loss for a normally active connection is unexpected. In this case, the SNALINK LU6.2 address space writes connection loss messages to the MVS system log.

When a connection is configured with the INIT parameter on the DEST statement and a timeout value of zero on the LINK statement in the SNALINK LU6.2 configuration data set, the connection is a normally active connection.

When a connection is configured with the DATA parameter on the DEST statement and a nonzero timeout value on the LINK statement in the SNALINK LU6.2 configuration data set, the connection is a normally inactive connection.

Check the connection experiencing the loss to ensure the loss is unexpected. If the connection loss experienced is specifically caused by errors, the loss is unexpected regardless of the connection configuration.

**Documentation**

Unexpected connection loss occurs if the SNALINK LU6.2 address space encounters errors that compromise the connection. In this case, error messages are written to the data set specified on the SYSPRINT DD statement in the SNALINK LU6.2 cataloged procedure.

To check the status of the SNA LU type 6.2 connection, issue the LIST MODIFY subcommand to the MVS SNALINK LU6.2 address space. See “Using the SNALINK LU6.2 subcommand” on page 569 for more information about issuing this command and reading the output.

**Analysis**

Use the error messages in the SNALINK LU6.2 SYSPRINT data set to identify the cause of the loss. See “Finding error message documentation” on page 578 for details on finding the documentation for these messages. Text in the message documentation specifies the action required to fix the problem.

Table 45 on page 566 lists an example of an outage problem.
**Table 45. Outage problem**

<table>
<thead>
<tr>
<th>If this is displayed...</th>
<th>Then this might have occurred...</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The message EZA5797E Rejecting bind from xxxxx-no DLC found, along with VTAM error message IST663I Bind fail request received, SENSE=800A0000, is displayed on the MVS system console.</td>
<td>Large packet size sent in a PIU is rejected by the NCP with sense 800A0000 (PIU too large).</td>
<td>The PIU includes the TH, RH, and RU. SNALINK attempts to send data up to the MAXRU size. The total size of the PIU includes the RU portion and the additional 29 bytes for the TH and RH. If this exceeds the maximum size, NCP issues a negative response with sense 800A0000 (PIU too large), which results in the SNA session being taken down between SNALINK and the NCSTLU. When the DLC connection is reestablished, the NCP sends a Bind RU which is then rejected with sense 800A0000. The definitions used in the NCP and SNALINK must be such that MAXRU is at least 29 bytes less than MAXDATA. Refer to <a href="https://www.ibm.com">z/OS Communications Server: SNA Network Implementation Guide</a> for more information on defining the MAXDATA, MAXBFRU, and UNITSZ operands.</td>
</tr>
</tbody>
</table>

---

**Data loss problems**

These problems are related to data transfer over the SNALINK LU6.2 network. The first step is to determine the point in the network where the data is being lost. The following information is mainly concerned with determining the actual place of loss.

**Steps for documenting data loss problems**

**Before you begin:** To determine where the data packets are being lost, use the LIST MODIFY command for the SNALINK LU6.2 address space. See “Using the SNALINK LU6.2 subcommand” on page 569 for details. When listing the connection status, the number of packets sent and received over the connection since establishment is displayed in the report.

Perform the following steps to help you determine the source of the data loss.

1. Record the current packet count for the SNALINK LU6.2 devices in the network that support the LIST MODIFY command.

2. Issue the PING command on one end of the connection. In a correctly functioning network, PING sends a data packet to the other end of the connection, which then sends a response data packet back to the PING command.

3. Use the updated packet counts to determine how far the packet went.

4. Issue the PING command from the other end of the connection.
5. Use the updated packet counts to determine how far the packet got.

**Tip:** IP packet trace, as described in “Using IP packet trace” on page 576, can also be used to trace and validate the IP data packets as they enter and leave the SNALINK LU6.2 address space.

**Analysis**

Table 46 lists some of the common SNALINK LU6.2 data loss problems.

<table>
<thead>
<tr>
<th>If this is displayed . . .</th>
<th>Then this might have occurred . . .</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data packets are lost between the TCP/IP and the SNALINK LU6.2 address space (either end).</td>
<td>This problem can be due to one of the following situations: 1. The DLC link between the TCP/IP address space and the SNALINK LU6.2 address space might not be active. 2. The SNALINK LU6.2 address space might be discarding packets.</td>
<td>1. See “DLC connection problems” on page 561 to diagnose the DLC link problem. 2. When a condition occurs in the SNALINK LU6.2 address space that causes data to be lost, “discarding datagram” messages are written to the data set specified by the SYSPRINT DD statement in the SNALINK LU6.2 cataloged procedure. See “Finding error message documentation” on page 578 for details on finding the documentation for these messages. Text in the message documentation specifies the action required to fix the problem.</td>
</tr>
<tr>
<td>Data packets are actually not lost but the protocol (PING) times out.</td>
<td>The SNALINK LU6.2 device might be establishing the LU type 6.2 connection to transfer the data packets. The delay in establishing the connection might be causing the protocol to time out.</td>
<td>If the DATA parameter is specified on the DEST statement for the connection in the SNALINK LU6.2 configuration data set, the connection is not established until data is to be transferred over the connection. In this case, after the first data transfer, further data packets are transferred successfully. If the TIMEOUT parameter is specified on the LINK statement for the connection in the SNALINK LU6.2 configuration data set, the connection can be timing out too often, causing the connection to be reestablished for each data transfer. In this case, the protocol timeout value or the connection timeout value should be increased.</td>
</tr>
<tr>
<td>Data packets are lost between the SNALINK LU6.2 devices.</td>
<td>The network is failing.</td>
<td>Check for VTAM error messages on the MVS system console. See “VTAM buffer traces” on page 578 for more details about using VTAM traces to diagnose the SNA network.</td>
</tr>
</tbody>
</table>
Data corruption problems

To determine the source of corruption for the data packets, use the IP packet tracing facility. This facility traces and validates the IP data packets as they enter and leave the SNALINK LU6.2 address space. Using this facility, the source of corruption can be identified as either the SNA network or the TCP/IP system.

Documentation

Set up the network conditions that are experiencing the data corruption. Start component trace in the SNALINK LU6.2 address space. Use the appropriate amount of data and time to ensure the corruption occurs.

Guideline: Allocate the MVS GTF trace data set (usually SYS1.TRACE) large enough to hold the expected trace output. This trace data set wraps back to the start of the data set when full, overwriting trace information. When tracing, this option does not collect all the data, which means the corruption could be missed. When formatting, this option turns off some of the IP packet validation processing.

Analysis

The IP packet trace facility analyzes the data corruption problem automatically. After the trace is collected, the trace data is passed through a formatter, which presents the data packets in an easy-to-read report and validates the contents of the packets against the RFC requirements. Every byte of the data packet is validated including reserved fields. The checksums are also recalculated and verified. If any of the data packets traced are corrupted, the formatter writes messages in the formatted report.

You can use this method, possibly together with TCP/IP internal traces, network level traces, or both, to identify the source and type of corruption.

For details on how to use the IP packet trace facility, see "Using IP packet trace" on page 576.

Documentation references for problem diagnosis

This section contains the information and documentation references required to gather and decode diagnostic information about the SNALINK LU6.2 network connection.

The main tools used for problem diagnosis are the NETSTAT utility, the SNALINK LU6.2 LIST subcommand, VTAM status display operations, the SNALINK LU6.2 internal trace facility, and the IP packet trace facility. The use of these tools is explained in the following sections. An explanation of how to interpret the output from each of these tools is also provided and referenced against the sample output.

For TCP/IP internal tracing or VTAM buffer tracing, you are referred to the appropriate diagnosis documentation.

Two cross-reference sections are provided at the end of this section that list all of the types of abend codes, sense codes, and error messages that can be issued from the SNALINK LU6.2 network connection. For each type of abend code, sense code, or error message, you are referred to the documentation that provides a complete description.
Using **NETSTAT**

This section describes how to use NETSTAT to query the state of TCP/IP devices. This command can be used to quickly verify the status of the SNALINK LU6.2 device and link with relation to the TCP/IP address space.

The NETSTAT DEVLINKS command output displays only information that is known to TCP/IP.

**Restriction:** The TCP/IP address space must be started before the NETSTAT command can query the connection status.

The command NETSTAT DEVLINKS displays the devices and links that have been defined to the main TCP/IP address space and the status of these devices (whether active or inactive).

*Figure 67* shows a sample of output from the NETSTAT DEVLINKS command.

```
DevName: SNALU621  DevType: SNALU62
DevStatus: Ready
LnkName: SNALU62L  LnkType: SNALU62  LnkStatus: Ready
  NetNum: 1  QueSize: 0
  BytesIn: 0  BytesOut: 0
  ActMTU: 32764
BSD Routing Parameters:
  MTU Size: 00000  Metric: 00
  DestAddr: 0.0.0.0  SubnetMask: 255.0.0.0
Multicast Specific:
  Multicast Capability: No
```

*Figure 67. NETSTAT DEVLINKS output example*

The example shows four SNALINK LU6.2 devices and associated links known to TCP/IP.

The most significant field for diagnosing DLC connection problems is the DevStatus field. Refer to *z/OS Communications Server: IP Configuration Reference* for detailed interpretation of the device status and its importance in the SNALINK LU6.2 DLC connection.

Using the **SNALINK LU6.2 subcommand**

This section details how to use the LIST MODIFY subcommand for the MVS SNALINK LU6.2 address space. The SNALINK LU6.2 address space has interactive commands to control the operation and list the status of the active address space. The LIST MODIFY subcommand writes a report to the MVS system console giving the status of the specified connections.

The connection status listed by the LIST subcommand can be requested for a particular remote VTAM application LU name or destination IP address. The following is an example using the LU parameter:

```
MODIFY procname,LIST LU=lu_name
```

In this example, *procname* is the member name of the cataloged procedure used to start the local SNALINK LU6.2 address space and *lu_name* is the remote VTAM application LU name of the connection for which you are requesting the status.
Figure 68 shows a sample output from the subcommand.

f snal621a,list lu=snal622a
EZA5971I LIST ACCEPTED; RANGE = SINGLE CONNECTION
EZA5967I 9.67.22.2 (Connected on 01.051 at 15:44:32)
EZA5968I Connected via: DATA Trace Level: ON
EZA5969I SEND:- Status: Allocated Packets Out: 1
EZA5970I RECE:- Status: Allocated Packets In: 1
EZA5974I LIST COMPLETED

Figure 68. LIST MODIFY subcommand output example

An active connection displays the EZA5968I Connected message with the "Allocated" status for both the send and receive conversations.

The SNALINK LU6.2 connection allocates two LU type 6.2 conversations: one for sending data to the remote device and one for receiving data. For independent LUs, the remote LU name is the same for both conversations. For dependent LUs, a remote LU name is specified for both the send and receive conversations.

The Packets In and Packets Out fields are decimal counters that record the number of data packets received from the remote SNALINK LU6.2 and the number of data packets sent to the remote SNALINK LU6.2, respectively. These fields can be used to identify configuration errors that cause data packets to be lost or discarded. For example, the packet counters can be used to track how far a PING packet travels around the network circuit before it gets lost. Each counter incremented means the packet made it past that point.

For more information about the contents of the messages from the LIST MODIFY subcommand, see the message documentation referenced in "Finding error message documentation" on page 578. Refer to the z/OS Communications Server: IP System Administrator's Commands for more explanation of the LIST MODIFY subcommand.

Useful VTAM operations

This section describes how to use the VTAM DISPLAY and VARY commands to activate an LU, change an LU definition, and to check the status of an application LU.

VTAM application LUs are defined with VTAM macros in a member of the SYS1.VTAMLST data set. The data set member, called the major node, can contain many application LU definitions, called minor nodes. The application LU names (minor node names) are specified on the VTAM and DEST statements in the SNALINK LU6.2 configuration data set.

Activating an LU

To activate an LU, the major node containing the LU definition must be activated first. If there are no definition errors, all the minor nodes defined in the major node are activated when the major node is activated. If a minor node becomes inactive, it can be activated individually. The following is an example of a VTAM VARY subcommand to activate a major or minor node:

VARY NET,ACT,ID=node_name

In this example, node_name is the major or minor node name to activate.

See "Displaying the status of an LU" on page 571 for an explanation of the active states for a minor node.
Changing an LU definition
To change an LU (minor node) definition, the major node containing the LU
definition must be deactivated and then reactivated to force VTAM to read the new
definition. The following is an example of a VTAM VARY subcommand to
deactivate a major node:

VARY NET, INACT, ID=majnode_name

In this example, majnode_name is the major node name to deactivate.

See “Activating an LU” on page 570 for the major node activation subcommand.

Displaying the status of an LU
To display the status of an LU definition, use the following command:

DISPLAY NET, ID=node_name, E

In this example, node_name is the major or minor node name for which you want
to display the status.

Displaying the status of a major node lists all of the minor nodes defined to the
major node and their STATUS field. For complete information on status of a minor
node, specify the actual minor node name in the command.

The STATUS field for a successfully activated LU definition is set to “CONCT,”
which means connectable. An LU in this state is waiting for the SNALINK LU6.2
address space to be started. An LU in the CONCT state cannot establish an LU
type 6.2 conversation.

Figure 69 on page 572 shows a sample of the output from an LU in connectable
state.
After the SNALINK LU6.2 address space has successfully started, the STATUS field is set to ACTIV, which means in use by an address space.

Figure 70 shows a sample of the output from a DISPLAY command for an LU in active state.

After the SNALINK LU6.2 address space has successfully started, the STATUS field is set to ACTIV, which means in use by an address space.

This example shows that the SNALINK LU6.2 address space (SNAL621A) has been started successfully and has its local LU (SNAL621A) in use with three sessions active to a remote LU (SNAL622A).
For each SNALINK LU6.2 connection, VTAM establishes three sessions between the application LUs. The first is the control session, which is the middle session in the example. The other two sessions are established for the LU type 6.2 conversations allocated for the connection, one for sending data and one for receiving data.

Refer to [z/OS Communications Server: SNA Operation](#) for more information about the DISPLAY command.

### Traces

Use the following traces to obtain information about the data flows and actions of the SNALINK LU6.2 network connection:

- SNALINK LU6.2 internal trace
- IP packet trace
- TCP/IP internal trace
- VTAM buffer trace

The SNALINK LU6.2 internal trace is the most useful for determining the state of the SNALINK LU6.2 address space. The IP packet trace facility is the most helpful trace facility for monitoring IP packets transferred across the SNALINK LU6.2 network. The TCP/IP internal traces can be used to diagnose problems with the DLC link between TCP/IP and SNALINK LU6.2. The VTAM buffer trace is used to monitor data transactions through the VTAM API interface.

#### Using SNALINK LU6.2 internal traces

The SNALINK LU6.2 internal traces are written to the location specified by the SYSPRINT statement in the SNALINK LU6.2 cataloged procedure. These traces provide information on the internals of the SNALINK LU6.2 address space.

SNALINK LU6.2 internal tracing is enabled by specifying the following statement in the SNALINK LU6.2 configuration data set:

```
TRACE DETAIL ALL
```

The SNALINK LU6.2 internal trace can also be started by passing a MODIFY console command to the SNALINK LU6.2 interface. The following MODIFY command starts internal tracing:

```
MODIFY procname,TRACE DETAIL ALL
```

In this example, `procname` is the member name of the cataloged procedure used to start the local SNALINK LU6.2 address space.

Refer to the [z/OS Communications Server: IP Configuration Reference](#) for detailed descriptions of the TRACE statement parameters and the TRACE subcommand parameters.
EZA5925I  TCPIP ADDRESS SPACE NAME SET TO TCPCS
EZA5926I  LU62CFG : STARTING PASS 1 OF 2
EZA5926I  LU62CFG : STARTING PASS 2 OF 2
EZA5927I  LU62CFG : NO ERRORS DETECTED - INITIALIZATION WILL CONTINUE
EZA5931I  INITIALIZATION COMPLETE - APPLID: SNAL621A TCP/IP: TCPCS
EZA5997I  CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
EZA5994I  LU= SNAL621A, DLU= SNAL622A, IP ADDRESS= 9.67.22.2
EZA6029E  OPRCNOS ERR. R15 00000000 R0 0000000B RTNCD 00000000 FDBK2 0000000B
EZA6030E  OPRCNOS ERR. RCPR= 0008, RCSEC= 0001
EZA6031E  SENSE CODE RECEIVED: 08570003
EZA6023E  UNABLE TO COMPLETE CNOS ON LU SNAL622A FOR 9.67.22.2
EZA6011E  UNABLE TO ALLOCATE SEND CONVERSATION FOR 9.67.22.2
EZA6009W  CONVERSATIONS FOR 9.67.22.2 TERMINATED
EZA5933I  LINK SNALU62L OPENED
EZA5997I  CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
EZA5986I  VTAM CONVERSATION ALLOCATED; CONVID= 01000012, SID= EAABEEC3D9B90C3F
EZA5984I  OLU= SNAL622A, DLU= SNAL621A, IP ADDRESS= 9.67.22.2
EZA5997I  CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
EZA5999I  IP DATAGRAM ADDED TO THE VTAM SEND QUEUE, LENGTH= 276, QUEUE COUNT = 1
EZA5995I  NUMBER OF IP PACKETS SENT ON 9.67.22.2 = 1
EZA5999I  450001140315000040014D4C09431601094301020800D58BEFBF74
EZA5999I  0AB56D2E96EF805CA03EFDD80
EZA5999I  78B6FE8035FE858058EE968017DA0B015B4AE800992C55B03DA1468027187D807CC5
EZA5999I  EB0745025B0
EZA5999I  39CF6B000F24D8023BEE80B12A9F5805434A6804C9FD8062498EB05779C5807B955
EZA5999I  6800961E80
EZA5999I  7CF6E8000EFF958060680080E7AB8001C2F8E0597B65803444B680829D808050B
EZA5999I  CB08352D3580
EZA5999I  2B1366B006A5B0217C7EB0745505807DB16B060492D8064A2EB04232D580175E
EZA5999I  CB04F93FD80
EZA5999I  6B31DBE802206A580625B760653C0CD8085FF3EB06F1075B801922680001D9D80664F3
EZA5999I  EB05C045B8
EZA5999I  3C2D6B060C906D80E4EEB075C615801FA86B083953D8011D49EB01DF1E5807412
EZA5999I  368000000000
EZA5999I  7C2F6E8000F24D8023BE0E8012A9F5805434A6804C9FD8062498EB05779C5807B955
EZA5999I  6800961E80
EZA5999I  7CF6E8000EFF958060680080E7AB8001C2F8E0597B65803444B680829D808050B
EZA5999I  CB08352D3580
EZA5999I  2B1366B006A5B0217C7EB0745505807DB16B060492D8064A2EB04232D580175E
EZA5999I  CB04F93FD80
EZA5999I  6B31DBE802206A580625B760653C0CD8085FF3EB06F1075B801922680001D9D80664F3
EZA5999I  EB05C045B8
EZA5999I  3C2D6B060C906D80E4EEB075C615801FA86B083953D8011D49EB01DF1E5807412
EZA5999I  368000000000
EZA5999I  7C2F6E8000F24D8023BE0E8012A9F5805434A6804C9FD8062498EB05779C5807B955
EZA5999I  6800961E80
EZA5999I  7CF6E8000EFF958060680080E7AB8001C2F8E0597B65803444B680829D808050B
EZA5999I  CB08352D3580
EZA5999I  2B1366B006A5B0217C7EB0745505807DB16B060492D8064A2EB04232D580175E
EZA5999I  CB04F93FD80
EZA5999I  6B31DBE802206A580625B760653C0CD8085FF3EB06F1075B801922680001D9D80664F3
EZA5999I  EB05C045B8

Figure 71. SNALINK LU6.2 internal trace output (Part 1 of 2)
Following are brief explanations of the numbered items in the output:

1. Messages written to the MVS system console.
2. VTAM send and receive conversation status.
3. Information about the VTAM API interface data flow.
   The VTAM interface information contains the LU type 6.2 conversation ID (Convid), the VTAM session ID (SID), length of the VTAM logical record, the origin and destination VTAM application LUs, and the home IP address.
   The VTAM logical record length should be four greater than the length of the TCP/IP datagram packet to account for the VTAM logical record header.
4. Information about data received from the TCP/IP DLC connection.
   Datagrams received from the SNALINK LU6.2 DLC connection are unpacked from the DLC message and added to the appropriate VTAM send queue for transmission.
5. Information about data received from the VTAM API interface.
   Datagrams received from the VTAM API are packed into a DLC message buffer.
6. Hexadecimal display of data passed through the SNALINK LU6.2 address space.
   There should be a hexadecimal display for every 4 and 5 event.
7. Change number of sessions (CNOS) data.

Figure 71. SNALINK LU6.2 internal trace output (Part 2 of 2)
Refer to the *z/OS Communications Server: SNA Programmer’s LU 6.2 Guide* for more information about CNOS processing.

**Using IP packet trace**

Trace on the SNALU62 LINKNAME using the TCPIP PKTTRACE command or on the SNA LU name using the VTAM buffer trace command.

If the LINKNAME parameter of the IP packet trace facility is specified, only packets transferred along the given link are traced. Specifying this parameter is recommended to avoid tracing a large number of unrelated packets.

See [Chapter 5, “TCP/IP services traces and IPCS support,” on page 47](#) for details about how to use the IP packet trace facility.

[Figure 72 on page 577](#) shows an example of a CTRACE formatted packet trace record.
TCP/IP internal traces

The TCP/IP internal traces are written to the data set specified on the TCP/IP address space SYSDEBUG ddname statement. These traces provide information on the internals of the TCP/IP address space that can be used to diagnose problems in establishing the DLC link between the TCP/IP address space and the SNALINK LU6.2 address space.
VTAM buffer traces

The VTAM buffer traces provide information on the contents of the VTAM API buffers. This information can be used to follow the data through the VTAM API interface. For details about VTAM buffer tracing and reading the trace reports, refer to z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures.

Finding abend and sense code documentation

The following list refers to the appropriate abend and sense code documentation for all abend and sense codes expected in the SNALINK LU6.2 network connection:

- Refer to z/OS Communications Server: IP Messages Volume 1 (EZA) and z/OS Communications Server: IP and SNA Codes for detailed SNALINK LU6.2 abend code descriptions.
- Sense codes in SNALINK LU6.2 error messages are generated by VTAM. Refer to z/OS Communications Server: SNA Messages and z/OS Communications Server: IP and SNA Codes for detailed sense code descriptions.

Finding error message documentation

The following list refers to the appropriate error message documentation for all error messages expected when using SNALINK LU6.2:

- Error messages from SNALINK LU6.2 are written to the SNALINK LU6.2 SYSPRINT data set and the MVS system console. Refer to z/OS Communications Server: IP Messages Volume 1 (EZA) and z/OS Communications Server: IP and SNA Codes for descriptions of the SNALINK LU6.2 error messages.
- Error messages from TCP/IP are written to the TCPIP SYSERROR data set. Refer to z/OS Communications Server: IP Messages Volume 1 (EZA) and z/OS Communications Server: IP and SNA Codes for descriptions of the error messages in these data sets.
- Error messages from VTAM are written to the MVS system console. Refer to z/OS Communications Server: SNA Messages and z/OS Communications Server: IP and SNA Codes for descriptions of the VTAM error messages written to the MVS system console.
Chapter 21. Diagnosing name server problems

This topic describes how to diagnose problems involving the BIND-based dynamic domain name server (DNS) and contains the following section:

- "Identifying name server problems"

Problem diagnosis involving connection optimization is also described.

For additional information about diagnosing problems with a BIND-based name server, see the latest version of DNS and BIND, Paul Albitz and Cricket Liu, available from the O'Reilly Online Catalog.

If, after reading this topic and DNS and BIND, you are unable to solve a DNS-related problem and you require the services of the IBM Software Support Center, gather the output from the debug log file with debug level of 10 or higher. You might be able to tailor the debug level to a more specific value by referring to Table 48 on page 582. See BIND 9 name server configuration logging statement for logging options. BIND 9 dynamic or static debug level might be set up to 90 for more detailed information, though it might affect server performance.

Identifying name server problems

The following methods are available for identifying name server problems:

- "Checking messages sent to the operator’s console"
- "Checking the log messages” on page 580
- "Tools for querying the name server” on page 581
- "Using the debug option with the name server” on page 582
- "Debugging with a resolver directive” on page 583
- "Using the remote name daemon control (rndc) program” on page 584
- "Using name server signals” on page 584
- "Statistics file for the Bind 9 name server” on page 585
- "Using the nsupdate command” on page 585
- "Using component trace” on page 585

These methods are discussed in the following sections.

Checking messages sent to the operator’s console

Messages that display automatically on the operator’s console indicate the status of your name server. Check console messages regularly to identify problems.

Messages fall into the following four categories:

- Name server initialization
- Name server initialization failure
- Name server initialization complete (such as EZZ9130I NAMED, BIND 9.2.0 IS RUNNING)
- Name server termination

For explanations of console messages, refer to z/OS Communications Server: IP Messages Volume 4 (EZ9, SNM)
Checking the log messages

For the BIND 9 name server, it is important to have the syslog daemon running before the name server is started. The BIND 9 name server logging files are not initialized until after the name server configuration files have been read and processed. Therefore, any messages issued as a result of syntax or semantic errors in the BIND 9 configuration files only appear in the syslogd output files. A MVS console message is issued indicating that the name server ended with a unrecoverable error.

Error messages can also be sent to a log file. You specify the name and location of this file in the syslog configuration file /etc/syslog.conf. Be sure to start syslogd before you start the named daemon.

For descriptions of the syslog file and the syslogd daemon, refer to the z/OS Communications Server: IP Configuration Guide. For information about syslog messages, refer to z/OS Communications Server: IP Messages Volume 3 (EZV).

BIND 9

The following applies only to the BIND 9 name server.

- Named debug trace (up to level 99)
  The debug trace can be directed to any file using the logging options in named.conf file. Log files are available to log important events.

- Logging can also be directed to the syslog file but severity is then limited to info and higher (does not include debug levels).

- Logging can be filtered by severity (critical, error, warning, notice, info, debug [level], dynamic).

The following is an example of the logging {} section of a named.conf file:

```plaintext
logging {
    channel main_log {
        file "'/tmp/named_main.log'" versions 2 size 5M;
        severity debug 10;
    };
    (blank line)
    channel query_log {
        file "'/tmp/named_query.log'" versions 2 size 5M;
        severity debug 10;
        category security { query_log;
        main_log; }
        category queries { query_log;
        main_log; }
        category default { main_log; }
    };

    Guideline: This example defines 2 arbitrarily named logging channels. All debug categories are logged to the main channel so that all events can be displayed together. The security and queries categories are also sent to the query channel for faster identification of these events. The latter categories could also be pulled off the main channel altogether. Up to 30 M of disk space can be used by the 2 channels in a round robin scheme of 3 times 5 M per channel as defined

    The events are categorized, and different categories can be logged to individual files if desired. The logging categories as shown in Table 47 on page 581
```
### Table 47. Logging categories

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Defines the logging options for those categories where no specific configuration has been defined.</td>
</tr>
<tr>
<td>general</td>
<td>Any items not otherwise categorized.</td>
</tr>
<tr>
<td>queries</td>
<td>Queries the server is receiving (not logged through default category).</td>
</tr>
<tr>
<td>database</td>
<td>Messages relating to the databases used internally by the name server to store zone and cache data.</td>
</tr>
<tr>
<td>security</td>
<td>Configuration file parsing and processing.</td>
</tr>
<tr>
<td>config</td>
<td>Configuration file parsing and processing.</td>
</tr>
<tr>
<td>resolver</td>
<td>DNS resolution, such as the recursive lookups performed on behalf of clients by a caching name server.</td>
</tr>
<tr>
<td>unmatched</td>
<td>Messages that named was unable to determine the class of or for which there was no matching view. A one line summary is also logged to the client category. This category is best sent to a file or stderr; by default it is sent to the null channel.</td>
</tr>
<tr>
<td>xfer-in</td>
<td>Zone transfers the server is receiving.</td>
</tr>
<tr>
<td>xfer-out</td>
<td>Zone transfers the server is sending.</td>
</tr>
<tr>
<td>notify</td>
<td>The NOTIFY protocol.</td>
</tr>
<tr>
<td>client</td>
<td>Processing of client requests.</td>
</tr>
<tr>
<td>network</td>
<td>Network operations.</td>
</tr>
<tr>
<td>update</td>
<td>Dynamic updates.</td>
</tr>
<tr>
<td>dispatch</td>
<td>Dispatching of incoming packets to the server modules where they are to be processed.</td>
</tr>
<tr>
<td>dnssec</td>
<td>DNSSEC and TSIG protocol processing.</td>
</tr>
<tr>
<td>lame-servers</td>
<td>Lame servers. These are misconfigurations in remote servers, discovered by BIND 9 when trying to query those servers during resolution.</td>
</tr>
</tbody>
</table>

### Tools for querying the name server

The **onslookup** and NSLOOKUP commands are helpful in diagnosing resolution of name problems in the z/OS UNIX and TSO environments, respectively. The z/OS UNIX **dig** command or the TSO DIG command can also be used to query name servers for problem diagnosis.

To turn on debugging information from nslookup, enter the following commands from the z/OS UNIX shell:

```
onslookup
set debug
```

To turn the debugging information off, enter the **set nodebug** command.
You can turn on more detailed tracing for nslookup by entering the following command:

```
onslookup
set d2
```

To turn d2 off, enter `set nod2`. Debug and d2 can be turned on and off independently. Using debug does not turn on resolver tracing, but instead adds more query question and response information. Using d2 adds some code flow traces.

For more information about the `dig`, TSO DIG, onslookup and NSLOOKUP commands, refer to the `z/OS Communications Server: IP System Administrator's Commands`.

**Tip:** The onslookup command messages do not give a message ID for debugging and are not documented in the z/OS Communications Server library.

### Using the debug option with the name server

You specify debugging in the JCL start procedure for the named server. Alternatively, you can specify debugging with the `-d` option on the `named` command or dynamically turn on debugging while the name server is running. For the BIND 9 name server, the `rndc` command can be used to dynamically turn on tracing. Valid levels for BIND 9 are in the range of 1-99. The location of the debug file for BIND 9 is specified by the logging[] statement in named.conf.

BIND 9 debugging log levels in the range of 1-99. The most useful information is contained in the messages in levels up to about 60. Specifying a level of 99 is a simple way to ensure the logging of any helpful information. High debug level logging should be utilized sparingly on high activity servers. Different logging categories can be directed to different channels, and therefore, might utilize different logging severities. Limit log size in server configuration file to avoid running out of disk space because of active and archived BIND 9 logs. With BIND 9, named.run is the default debug logging channel, which only works if defined or default logging categories are using it and if the name server `-d` start option has been used.

The `rndc` command can also be helpful in dynamically changing the BIND 9 server debug level. There are `rndc` commands to increment the debug level, set it to the desired level, or reset the level to 0 (no debug information). It does not affect the debug level of logging files specified in channel statements in the `named.conf` file unless the severity level of the logging channel is dynamic. The `rndc trace/notrace` command affects all logging channels with a default or specified debug level of dynamic. This applies to `named.run` or logging channels defined in `named.conf` file. The default debug logging channel, and therefore, the `named.run` file in the name server's working directory (specified by the directory statement in the `named.conf` file). See “Using the remote name daemon control (rndc) program” on page 584 for more information.

For BIND 9, Table 48 lists the types of debug information that are captured at the following debug levels.

<table>
<thead>
<tr>
<th>BIND 9 debug level</th>
<th>Debugging information</th>
</tr>
</thead>
</table>

Table 48. BIND 9 debug information
## Table 48. BIND 9 debug information (continued)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basic name server operation. This includes received queries, NOTIFYs from master name servers, the loading of zones, maintenance operations (including zone transfers, SOA queries by slaves, cache cleaning, and zone expirations), and task dispatching of some of the higher level functions.</td>
</tr>
<tr>
<td>2</td>
<td>Multicast requests.</td>
</tr>
<tr>
<td>3</td>
<td>Journal activity when dynamic update is enabled, DNSSEC and TSIG validation (if configured), and lower level task creation operations.</td>
</tr>
<tr>
<td>4</td>
<td>Incidents when a master name server has to resort to using AXFR (complete zone transfers) instead of IXFR (incremental zone transfer) because of the unavailability of journal files.</td>
</tr>
<tr>
<td>5</td>
<td>Captures the view being used in order to answer a request.</td>
</tr>
<tr>
<td>6</td>
<td>Some outgoing zone transfer requests including the query that initiates the transfer.</td>
</tr>
<tr>
<td>7</td>
<td>The additions and deletions to journal files, and the number of bytes returned on zone transfers.</td>
</tr>
<tr>
<td>8</td>
<td>Most dynamic update activity and more detailed information on zone transfers.</td>
</tr>
<tr>
<td>10</td>
<td>Timer activity for zones.</td>
</tr>
<tr>
<td>20</td>
<td>Zone refresh timer updates.</td>
</tr>
<tr>
<td>90</td>
<td>Detailed information about task dispatching and operations.</td>
</tr>
</tbody>
</table>

With BIND 9, zone transfer logging mostly depends on the transfer logging category, which can be directed to a common or unique logging channel (file) specified by the user.

For BIND 9, `-d` does not entail working in the foreground. The latter depends on separate start options (`-f` or `-g`).

For details on the `named` command and logging, refer to the "z/OS Communications Server: IP System Administrator’s Commands".

### Debugging with a resolver directive

Programs that query name servers are called `resolvers`. To debug resolution of name problems, you can specify the debug option in the file `/etc/resolv.conf` (using the options debug directive) or in the TCP/IP configuration file. For additional methods to specify resolver trace, refer to APAR II13398. The resolver trace is sent directly into the output stream for the command using the resolver (for example, `nslookup`).
For more information, see Chapter 39, “Diagnosing resolver problems,” on page 853.

Using the remote name daemon control (rndc) program

The rndc program can be used to collect diagnostic information for BIND 9 name servers.

Configuration is required in order to use the rndc utility. Refer to the z/OS Communications Server: IP System Administrator’s Commands and the z/OS Communications Server: IP Configuration Reference for additional information.

The following rndc commands can be used to provide diagnostic data for the BIND 9 name server.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dumpdb</td>
<td>Dump the current contents of the cache (or caches if there are multiple views) into the file named by the dump-file option (by default, named_dump.db).</td>
</tr>
<tr>
<td>trace</td>
<td>Increment the server’s debugging level by one.</td>
</tr>
<tr>
<td>trace level</td>
<td>Increment the server’s debugging level to an explicit value.</td>
</tr>
<tr>
<td>notrace</td>
<td>Sets the server’s debugging level to 0.</td>
</tr>
<tr>
<td>flush</td>
<td>Flushes the server’s cache.</td>
</tr>
<tr>
<td>status</td>
<td>Displays the status of the server.</td>
</tr>
<tr>
<td>stats</td>
<td>Writer server statistics to the statistics file.</td>
</tr>
<tr>
<td>querylog</td>
<td>Toggle query log</td>
</tr>
</tbody>
</table>

Refer to the z/OS Communications Server: IP System Administrator’s Commands for additional information about the rndc command.

Using name server signals

You can use z/OS UNIX signals to send messages to the named daemon.

There are three signals that currently can be used with the BIND 9 name server. Table 50 lists the BIND 9 name server signals:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGHUP</td>
<td>Causes the server to read named.conf and reload the database.</td>
</tr>
<tr>
<td>SIGTERM</td>
<td>Causes the server to clean up and exit.</td>
</tr>
<tr>
<td>SIGINT</td>
<td>Causes the server to clean up and exit.</td>
</tr>
</tbody>
</table>

A sample MVS start procedure is included in the samples directory that lets you issue these signals to the name server from the MVS operator’s console. The name of the sample is nssig. It has one parameter, sig. If the sample procedure is unaltered, a typical invocation from the operator’s console would be the following:
Statistics file for the Bind 9 name server

Bind 9 statistics are written to a file when you issue the `rndc stats` command. The statistics dump begins with the line `+++ Statistics Dump +++ (973798949)`, where the number in parentheses is a standard UNIX-style timestamp, measured as seconds since January 1, 1970. Following that line are a series of lines containing a counter type, the value of the counter, optionally a zone name, and optionally a view name. The lines without view and zone listed are global statistics for the entire server. Lines with a zone and view name are for the given view and zone (the view name is omitted for the default view). The statistics dump ends with the line `--- Statistics Dump --- (973798949)`, where the number is identical to the number in the beginning line.

The following statistics are maintained:

- **success**: The number of successful queries made to the server or zone. A successful query is defined as query that returns a NOERROR response other than a referral response.
- **referral**: The number of queries that resulted in referral responses.
- **nxrset**: The number of queries that resulted in NOERROR responses with no data.
- **nxdomain**: The number of queries that resulted in NXDOMAIN responses.
- **recursion**: The number of queries that caused the server to perform recursion in order to find the final answer.
- **failure**: The number of queries that resulted in a failure response other than those above.

Using the `nsupdate` command

The `nsupdate` command creates and executes Domain Name System (DNS) update operations on a host record. For nsupdate Bind 9, the -d option turns on debugging. The -d option must be specified on the `nsupdate` command line, as there is no interactive command to turn on debugging after nsupdate Bind 9 has been started. For details on the `nsupdate` command, refer to the "z/OS Communications Server: IP System Administrator’s Commands".

Using component trace

You can use the component trace function to trace data at the TCP/IP layer. In particular, the Resolver component trace might be beneficial. This information can be helpful in resolving naming problems. For detailed information on the component trace function, see Chapter 5, “TCP/IP services traces and IPCS support,” on page 47.

For more information about Resolver CTRACE, see Chapter 39, “Diagnosing resolver problems,” on page 853.
Chapter 22. Diagnosing REXEC, REXECD, and RSH problems

This topic contains diagnosis information about the classic (non-z/OS UNIX) Remote Execution Protocol (REXEC), the Remote Execution Protocol Daemon (REXECD), and the remote shell client (RSH). See “General information about REXEC and RSH” for information about REXEC and RSH and “General information about REXECD” on page 588 for information about REXECD.

The following sections are included:

- “General information about REXEC and RSH”
- “General information about REXECD” on page 588

General information about REXEC and RSH

REXEC and RSH are remote execution clients that allow you to execute a command on a remote host and receive the results on the local host. REXEC and RSH commands can be executed from the TSO command line or as a batch program.

Refer to the z/OS Communications Server: IP Configuration Reference for information about defining the remote execution server.

Figure 73 shows the principle behind REXECD.

Documentation for REXEC problem diagnosis

The following kinds of information might be required to diagnose a REXEC problem:
• REXEC console log
• REXEC debug trace

**TSO console log**

The TSO console log should be saved and made available, particularly if there are any error messages displayed at the console.

**Activating the REXEC debug trace**

To activate the REXEC debug trace, use the REXEC -d command.

Refer to [z/OS Communications Server: IP User's Guide and Commands](#) for more information about REXEC commands.

**REXEC trace example and explanation**

Figure 74 shows an example of an REXEC trace. Short descriptions of the numbered items in the trace follow the figure.

REXEC trace output is sent to the TSO console from which the command was submitted.

```plaintext
rexec -d -l debfox -p mypwd norway time
Established affinity with TCPCS
EZA4801I MVS TCP/IP REXEC CS V1R5
EZA47751 Calling function rexec_af with the following:
Host: norway user: debfox cmd: time port: 512
EZA47741 rexec invoked;
Data socket = 1 Control socket = 3
IKJ56650I TIME-01:22:00 PM. CPU-00:00:00 SERVICE-5982 SESSION-00:00:01 March 24, 2003
EZA47891 rexec complete
```

Figure 74. Example of an REXEC trace

**RSH trace example and explanation**

Figure 75 shows an example of an RSH trace. Short descriptions of numbered items in the trace follow the figure.

RSH trace output is sent to the RSH console.

```plaintext
rsh -d -l user1/tcpsup norway time
Established affinity with TCPCS
EZA5025I Calling function rcmd_af with the following:
Host: norway user: user1 cmd: time port: 514
EZA5046I rsh invoked;
Data socket = 1 Control socket = 3
IKJ56650I TIME-02:30:30 PM. CPU-00:00:00 SERVICE-6454 SESSION-00:00:00 March 24, 2003
EZA5048I rsh complete
```

Figure 75. Example of an RSH trace

**General information about REXEC**

The remote execution server allows execution of a TSO batch command that has been received from a remote host. REXEC supports both the remote execution command (REXEC) and remote shell (RSH) client protocols.

**Note:** When the REXEC server is active, it has outstanding listens on Ports 512 and 514. If you want to have a concurrent server for the z/OS UNIX REXECD or RSHD daemons, then configure them to use different ports.
Documentation for REXECD problem diagnosis

The following kinds of information might be required to diagnose a REXECD problem:
- REXECD console log
- REXECD traces

MVS system console log

The MVS system console log should be saved and made available, particularly if there are any error messages displayed at the console.

Starting REXECD server traces

To run the REXECD trace, REXECD must be started with one or more of the following options on the TRACE parameter in the PROC statement:

- **LOG**
  Specifies to write trace records to the SYSPRINT data set.
- **SEND**
  Specifies to send trace records to the REXEC or RSH client.
- **CLIENT**
  Specifies a specific client host for which trace records are to be produced.
- **ALLCLIENTS**
  Specifies that host records are to be produced for all clients.

Refer to the [z/OS Communications Server: IP Configuration Reference](#) for more information about the options. Refer to the [z/OS MVS JCL Reference](#) for information about the length limit of the PARM= parameter on the exec statement in the start procedure. REXECD trace output is included in the job output log.

**Restriction**: If more than one trace option is selected, the options must be enclosed within parentheses.

Example of an REXECD trace of a client using the SEND command

Figure 76 on page 590 shows a portion of an example of an REXECD trace of a client using a SEND command. Short descriptions of numbered items in the trace follow the figure.
Following are short descriptions of the numbered items in the trace:

1. JOB00043 is the JES job number. The number 40 indicates that the job is waiting for execution. This means that the remote execution server has processed the REXEC client request, created a JES job, and has submitted the JOB to JES. The server continues to check the status.

   **Guidelines:** If the status does not change from 40, this could indicate one of the following problems:
   - A JES initiator has not started to process the submitted job class.
   - Other jobs might be running in this class that are inhibiting this job from starting.

   Make a note of the job number and check JES activity, or check any other jobs that might be running at the same time.

2. The number 80 indicates that the job is currently active. This means that the remote execution server has checked with JES on the job status and was informed that the job is running.

   **Guideline:** If the status does not change from 80, this could indicate that the job is taking too long to run. REXEC was not intended to be used for long running jobs. Long running jobs should be submitted using FTP JES processing.

   **Tip:** The FTP JES interface provides an alternate method of submitting a batch job remotely and retrieving the results.

3. The number 20 indicates that the job is on the output queue. This means that when the server checked with JES on the job status it discovered that the JES job has completed, and it has been placed on the output queue. The command output should be sent back to the client soon. See [z/OS Communications Server: IP Messages Volume 1 (EZA)] for more information about the individual messages in the trace.

4. This line shows the return code from the dynamic allocation of the JES data sent back to the client.

5. Actual command output sent to the client.

6. This line displays the return code that is expected when the process has completed.
**Example trace of an RSH client using the SEND command**

Figure 77 shows a portion of an example of a trace of an RSH client using a SEND command. Short descriptions of numbered items in the trace follow the figure.

1. EZA4438I SSCSARAY[0] JOB00043 20 COMPLETED
   EZA4385I SSSORT(CTRL): 00000000
   EZA4389I SSSORT(init): 00000004
2. EZA4437I SSCSARAY[1] JOB00044 40 WAITING
   EZA4438I SSCSARAY[0] JOB00043 20 COMPLETED
   EZA4385I SSSORT(CTRL): 00000000
   EZA4389I SSSORT(init): 00000004
3. EZA4436I SSCSARAY[1] JOB00044 80 ACTIVE
   EZA4438I SSCSARAY[0] JOB00043 20 COMPLETED
   EZA4385I SSSORT(CTRL): 00000000
   EZA4389I SSSORT(init): 00000004
4. EZA4438I SSCSARAY[1] JOB00044 20 COMPLETED
   EZA4385I SSSORT(CTRL): 00000000
6. TIME-12:07:02 PM. CPU-00:00:00 SERVICE-1134 SESSION-00:00:00 APRIL 1,2008
   EZA4393I S99ret: 00000000
7. EZA4421 SSSORT(next): 00000004 NO MORE OUTPUT ON JES SPOOL

Figure 77. Example of a trace of an RSH client using a SEND command

Following are short descriptions of numbered items in the trace:

1. JOB00043 is a previous job that has completed.
2. The 40 indicates that job JOB00044 is waiting for execution. This means that the remote execution server has processed the RSH client request, created a JES job, and has submitted the JOB to JES. The server continues to check the status.

   **Guidelines:** If the status does not change from 40, this could indicate one of the following problems:
   - No JES initiator started to process the submitted job class.
   - Other jobs might be running in this class that are inhibiting this job from starting.

   Make a note of the job number and check JES activity, or check any other jobs that might be running at the same time.

3. The 80 indicates that job JOB00044 is currently active. This means that the remote execution server has checked with JES on the job status and was informed that the job is executing.

   **Note:** If the status does not change from 80 this could indicate that the job is taking too long to run. RSH was not intended to be used for long running jobs. Long running jobs should be submitted using FTP JES processing.

4. The 20 indicates that job JOB00044 is on the output queue. This means that when the server checked with JES on the job status it discovered that the
JES job has completed, and it has been placed on the output queue. The command output should be sent back to the client soon.

Note: Refer to z/OS Communications Server: IP Messages Volume 1 (EZA) for more information about the individual messages in the trace.

5 This line shows the return code from the dynamic allocation of the JES data sent back to the client.

6 Actual command output sent to the client

7 This is the return code expected when there is no more work to do.

Example trace to the JES spool file of the server

Figure 78 on page 593 shows a trace log as it appears on the server. Short descriptions of numbered items in the trace follow the figure.
Following are short descriptions of numbered items in the trace:

**Note:** Trace messages are preceded by the identification of which socket that trace entry applies too.

The listening socket is socket 0. The EZA4381I message for this socket identifies the socket the request is processed on and the IP address and port that the request was received on. Port 512 is for REXEC requests and port 514 is for RSH requests.

1. Indicates an REXEC request was received on socket 2 for the specified IP address. Subsequent entries beginning with socket 2 indicates activity occurring during the processing of this request. Message EZA441I is issued when this socket is closed.
EZA4382I identifies the socket 3 and the IP address and port the server is connecting back to the client on. This port (1255) is in the packet that the client sent to the server. Message EZA4442I is issued when this socket is closed. Common messages are EZA4382I and EZA4442I for this socket.

This line shows the return code from the dynamic allocation of the JES data sent back to the client.

Indicates the request is completed and the socket is closed.

Indicates the error socket is closed.
Chapter 23. Diagnosing z/OS UNIX REXEC, RSH, REXECD, and RSHD problems

This topic contains diagnosis information about the z/OS UNIX remote execution protocol (REXEC), remote shell protocol client (RSH), remote execution protocol daemon client (REXECD), and remote shell daemon (RSHD).

Setting up the inetd configuration file

The inetd program is a generic listener program used by such servers as z/OS UNIX TELNETD and z/OS UNIX REXECD. Other servers such as z/OS UNIX FTPD have their own listener program and do not use inetd.

The inetd.conf file is an example of the user’s configuration file. It is stored in the /etc directory. Upon startup, the servers for z/OS UNIX TELNETD, rshell, rlogin, and rexec are initiated if they have been defined in /etc/inetd.conf. If it does not include z/OS UNIX TCP/IP applications, add the information shown in Figure 79:

Guideline: For IPv6 support, specify tcp6 for the protocol.

When nowait is specified, the inet daemon issues an accept when a connect request is received on a stream socket. You can specify nowait.max, where max is the maximum number of users allowed to request service in a 60-second interval. The default is 40. If maximum is exceeded, the service’s port is shut down. If you expect more than 40 users per minute requesting service, specify the maximum that you expect.

To establish a relationship between the servers defined in the /etc/inetd.conf file and specific port numbers in the z/OS UNIX environment, ensure that statements have been added to ETC.SERVICES for each of these servers. See the sample ETC.SERVICES installed in the /usr/lpp/tcpip/samples/services directory for how to specify ETC.SERVICES statements for these servers.

Guideline: It is important that the service name in /etc/inetd.conf (login in Figure 79) matches the service name in /etc/services:

login 513/tcp

The traces for both the z/OS UNIX REXECD server and the z/OS UNIX RSHD server are enabled by options in the inetd configuration file (/etc/inetd.conf). See Figure 80 on page 596.
The traces are turned on for both servers by passing a -d argument to the server programs. \texttt{/SF5800002/SF590000} is the RSHD server and \texttt{/SF5800003/SF590000} is the REXECD server. All commands executed after the debug flags have been turned on in the inetd configuration file and after the inetd server has reread the file produces trace output.

The trace is written in formatted form to the syslogd facility name daemon with a priority of debug. The trace data can be routed to a file in your Hierarchical File System by specifying the following definition in your syslogd configuration file (/etc/syslogd.conf):

```
# All ftp, rexecd, rshd
# debug messages (and above
# priority messages) go
# to server.debug.a
#
daemon.debug /tmp/syslogd/daemon.debug.a
```

In this example, the trace data is written to \texttt{/tmp/syslogd/daemon.debug.a} in your hierarchical file system. Refer to the \texttt{z/OS Communications Server: IP Configuration Reference} for more information about syslogd.

For more information about inetd, refer to \texttt{z/OS UNIX System Services Planning}.

**Diagnosing z/OS UNIX REXEC**

The following kinds of information can help you diagnose a z/OS UNIX REXEC problem:

- A message beginning with EZYRC
- A code
- An z/OS UNIX REXEC debug trace
- A REXECD debug trace from the foreign host

**Activating the z/OS UNIX REXEC debug trace**

To activate the z/OS UNIX REXEC debug trace, specify the -d option.

**z/OS UNIX REXEC trace example and explanation**

The z/OS UNIX REXEC can be invoked using either \texttt{rexec} or \texttt{orexec}. Enter one of the following commands with either an IP address or a host name:

**IPv4**

```
orexec -d -l debfox -p mypwd -s 1512 197.22.190.1 ls -al
```

**IPv6**

```
orexec -d -l debfox -p mypwd -s 1512 fec0:0:0:12BE::1 ls -al
```

The following are examples of the trace output:

**IPv4**
IPv6
EZYRC02I Host: 197.22.190.1, user debfox, cmd ls -al, port 1512
EZYRC01I Calling function rexec.af with the following:
EZYRC02I Host: fec0:0:0:12BE::1, user debfox, cmd ls -al, port 1512
EZYRC19I Data socket = 4, Control socket = 6.

EZYRC01I shows that the z/OS UNIX REXEC function has been called in the run-time libraries. EZYRC02I shows the parameters that have been passed to the REXEC() function in the run-time library. EZYRC19I shows the socket descriptor being used for the data connection and the control (or standard error) connection.

### Diagnosing z/OS UNIX RSH

The following kinds of information can help you diagnose a z/OS UNIX REXEC problem:
- A code
- A z/OS UNIX RSH debug trace
- An RSHD debug trace from the foreign host

#### Step for activating the z/OS UNIX RSH debug trace

Perform the following step to activate the z/OS UNIX RSH debug trace.
- Specify the -d option.

#### Step for invoking z/OS UNIX RSH trace

The z/OS UNIX RSH can be invoked using either rsh or orsh.

Enter one of the following commands with either an IP address or a host name.

**IPv4**
```
orsh -d -l debfox/mypwd -s 1514 197.22.190.1 date
```

**IPv6**
```
orsh -d -l debfox/mypwd -s 1514 fec0:0:0:12BE::1 date
```

The following are examples of the trace output:

**IPv4**
```
EZYRC31I Calling function rcmd.af with the following:
EZYRC02I Host: 197.22.190.1, user debfox, cmd date, port 1514
EZYRC19I Data socket = 4, Control socket = 6.
Thu Apr 3 15:44:11 2003
```

**IPv6**
```
EZYRC31I Calling function rcmd.af with the following:
EZYRC02I Host: fec0:0:0:12BE::1, user debfox, cmd date, port 1514
EZYRC19I Data socket = 4, Control socket = 6.
Thu Apr 3 15:41:11 2003
```

EZYRC31I shows that the local rcmd.af() function has been called. EZYRC02I shows the parameters that have been passed to the rcmd.af() function. EZYRC19I shows the socket descriptor being used for the data connection and the control (or standard error) connection.
Diagnosing z/OS UNIX REXECD

The following kinds of information can help you diagnose a z/OS UNIX REXECD problem:

- A message beginning with EZYRD
- A code
- A z/OS UNIX REXECD debug trace
- A trace from the z/OS UNIX REXECD client

Activating the z/OS UNIX REXECD debug trace

The z/OS UNIX REXEC can be invoked using either rexec or orexec. To activate the z/OS UNIX REXECD debug trace, specify the -d option in the /etc/inetd.conf file.

z/OS UNIX REXECD trace example and explanation

These examples are in the file specified in syslogd.conf.

Note: Ensure syslogd is running before collecting these traces and that the file has been properly specified.

Jun 12 13:31:47 reexec.851981.: EZYRD31I MVS OE REXECD BASE

The entry is stamped with the date, time, the name of the daemon and the order number of the daemon, the message number (EZAYRD31I), and related information, as shown in the following example.

Jun 12 13:31:49 reexec.851981.: EZYRD03I Remote address = 9.67.113.61
Jun 12 13:31:49 reexec.851981.: EZYRD05I cliseports = 1029
Jun 12 13:31:49 reexec.851981.: EZYRD08I User is: user21
Jun 12 13:31:49 reexec.851981.: EZYRD09I Command is: ls -l
Jun 12 13:31:49 reexec.851981.: EZYRD12I Name is: USER21, user is user21
Jun 12 13:31:49 reexec.851981.: EZYRD13I dir is:/u/user21
Jun 12 13:31:49 reexec.851981.: EZYRD14I uid is: 21, gid is 0

For an explanation of the messages, refer to z/OS Communications Server: IP Messages Volume 1 (EZA).

Diagnosing z/OS UNIX RSHD

The following kinds of information can help you diagnose a z/OS UNIX RSHD problem:

- A message beginning with EZYRS
- A code
- A z/OS UNIX RSHD debug trace
- A trace from the RSH client

Step for activating the z/OS UNIX RSHD debug trace

The z/OS UNIX RSHD can be invoked using either rshd or orshd.

Perform the following step to activate the z/OS UNIX RSHD debug trace.

- Specify the -d option in the /etc/inetd.conf file.
z/OS UNIX RSHD trace example and explanation

These examples are from the file specified in syslogd.conf.

Restriction: Ensure syslogd is running before collecting these traces and that the file exists and has been properly specified.

Jun 9 12:10:04 rshd.4653080.: EZYRS01I MVS OE RSHD BASE

The entry is stamped with the date, time, name of daemon and the order number of the daemon, the message number (EZYRS01I), and related information, as shown in the following example.

Jun 9 12:10:06 rshd.4653080.: EZYRS12I C1isecport = 1020
Jun 9 12:10:06 rshd.4653080.: EZYRS21I Remote user is: OS2USER
Jun 9 12:10:06 rshd.4653080.: EZYRS22I Local user is: user21
Jun 9 12:10:06 rshd.4653080.: EZYRS23I Command is: ls -l

For an explanation of the messages, refer to z/OS Communications Server: IP Messages Volume 3 (EZY).

If the -A option is specified in /etc/inetd.conf, the z/OS UNIX RSHD server does not execute a command when the client host IP address cannot be resolved to a host name.

Resolving garbage errors

There are a few situations where the z/OS UNIX RSHD server might encounter an error so early in the processing of a command that the server has not yet established a proper EBCDIC-to-ASCII translation. In such a situation, the client end user might see garbage data returned to his or her terminal. A packet trace reveals that the response is in fact returned in EBCDIC, which is the reason for the garbage look on an ASCII workstation. This can happen if the z/OS UNIX name resolution has not been configured correctly, so the z/OS UNIX RSHD server, for example, was not able to resolve IP addresses and host names correctly. If your RSH clients encounter such a problem, go back and check your name resolution setup. If you are using a local hosts table, make sure that the syntax of the entries in your hosts file is correct.
Chapter 24. Diagnosing X Window System and Motif problems

This topic describes environment variable XWTRACE that might be useful when diagnosing X Window System and Motif problems. The environment variable, XWTRACE, controls the generation of traces of the socket level communication between Xlib and the X Window System Server.

The following sections are included:
- “Trace output when XWTRACE=2”
- “Trace output when XWTRACELC=2” on page 602
- XWTRACE undefined or zero — No trace generated.
- XWTRACE=1 — Error messages.
- XWTRACE>=2 — API function tracing for TRANS functions.

Another environment variable, XWTRACELC, causes a trace of various locale-sensitive routines. If XWTRACELC is defined, a routine flow trace is generated. If XWTRACELC=2, more detailed information is provided.

**Guideline:** There are no special post-install activities for GDDMXD in z/OS Communications Server. GDDM® APAR (PN77391) eliminated these activities for TC/IP Version 3 Release 1. However, if you have an old GDDMXD load library (tcpip.v3r1.SEZALNKG) in your LNKLSTxx member in SYSx.PARMLIB, you need to remove that library from the MVS link list, because it is no longer needed.

Following are some examples of X Window System traces.

**Trace output when XWTRACE=2**

[Figure 81 on page 602](#) shows a typical stream of socket level activity that is generated when an X application running on z/OS UNIX MVS exchanges information with an X Server.
Each line of the trace provides:

- The name of the function involved from x11trans
- Values of the parameters passed to the function

---

**Trace output when XWTRACE=2**

Figure 81 on page 603 is a partial trace showing typical types of information displayed by locale-sensitive routines.
Figure 82. Example of X Application trace output when XWTRACELC=2 (Part 1 of 2)
Each line of trace provides:
- The name of the locale routine.
- The function invoked within that locale routine.
- Where pertinent, charset name or encoding information, or charset name and encoding information.
- If exiting the invoked function, the trace statement indicates that the function is returning.

Figure 82. Example of X Application trace output when XWTRACELC=2 (Part 2 of 2)
Chapter 25. Diagnosing Simple Network Management Protocol (SNMP) problems

This topic explains SNMP-related concepts and terms, including information about how to diagnose SNMP problems and contains the following sections:

- "Overview"
- "Definitions" on page 607
- "Diagnosing SNMP problems" on page 609
- "SNMP traces" on page 627

Overview

The SNMP protocol provides a standardized interface, through which a program on one host (running an SNMP manager) can monitor the resources of another host (running an SNMP agent).

Management information base (MIB)

The information maintained at each agent is defined by a set of variables known as the management information base, or MIB. In addition to the architected list of variables that must be supported by each SNMP agent, an SNMP agent can also support user-defined variables. These user-defined variables that are not part of the architected MIB are known as enterprise-specific MIB variables.

On z/OS Communications Server, the majority of the MIB variables are maintained outside the SNMP agent address space by programs known as SNMP subagents. The subagent program for the TCP/IP-related MIB variables executes in the TCP/IP address space. The subagent program for OMPROUTE-related MIB variables run as part of OMPROUTE, not as a separate application. The subagent program for SLA-related MIB variables runs as a separate application. The subagent program for TN3270E Telnet server MIB variables executes as a separate subtask in the TN3270E Telnet server address space. For a list of all the MIB objects supported by the agent and subagents shipped as part of z/OS Communications Server, refer to the z/OS Communications Server: IP System Administrator’s Commands.

In addition, user-written subagent programs can also exist. All subagent programs, whether provided by z/OS Communications Server or user-written, communicate with the SNMP agent over an architected interface known as the Distributed Protocol Interface, or DPI®.

When the SNMP agent receives and authenticates a request, it passes the request to the DPI subagent that has registered as the target of the request. You can see this exchange by enabling DPI tracing within the agent.

PDUs

The SNMP protocol is based on the exchange of protocol data units, or PDUs, between the SNMP manager and the SNMP agent.

SNMP has seven types of PDUs:
**GetRequest-PDU**
Sent from the manager to request information from the agent.

**GetNextRequest-PDU**
Requests the next variable in the MIB tree.

**GetBulkRequest-PDU**
Requests the next variable in the MIB tree and can also be used to specify multiple successors.

**GetResponse-PDU**
Sent from the agent to return information to the manager.

**SetRequest-PDU**
Sent from the manager to alter information at the agent.

**Trap-PDU**
Sent from the agent to report network events to the manager. A trap is an unconfirmed notification.

**Inform-PDU**
Sent from an agent to a manager or from a manager to another manager to report a network event. Attempts to confirm delivery are made for Inform-PDUs, not Trap-PDUs.

### Functional components
The following sections provide detailed descriptions of the SNMP functional components.

**Managers**
A manager is a client application that requests management data. z/OS Communications Server provides two management applications, the z/OS UNIX `snmp` command (the `osnmp` command is a synonym for the `snmp` command), and the NetView SNMP command. The `snmp` command is a management application used from the z/OS UNIX shell to monitor and control network elements. The NetView SNMP command provides the same type of functions from the NetView environment.

The `snmp` command runs in a user address space that is created and removed as `snmp` is issued and completed. The NetView SNMP client requires the following started tasks:
- SNMPIUCV subtask of NetView, which runs in the NetView address space and provides the operator interface to SNMP.
- SNMP query stack address space, which provides the protocol support for the SNMP PDUs.

The SNMPIUCV subtask in the NetView address space and the SNMP query stack address space communicate over an IUCV connection.

**Agents**
An agent is the server that responds to requests from managers. The agent maintains the MIB. z/OS Communications Server supports a tri-lingual SNMP agent which can understand SNMPv1, SNMPv2c, and SNMPv3 versions of the SNMP protocol. It also communicates with the subagents using DPI1.1 and DPI2.0 protocols.
Subagents
Subagents help the agent by managing a part of the MIB. z/OS Communications Server supports the following subagents:

- TCP/IP subagent that manages TCP/IP-related standard MIB objects and several enterprise-specific MIBs
- OMPROUTE subagent that manages the ospf MIB
- Network SLAPM2 subagent that manages the Network SLAPM2 MIB
- TN3270E Telnet subagent that manages the Enterprise-specific TN3270 Telnet MIB

These subagents communicate with the SNMP agent using the DPI 2.0 protocol.

Trap forwarder daemon
The Trap Forwarder daemon on z/OS Communications Server listens for SNMP traps on a specified port and forwards them to other configured ports. This eliminates the port contention problem when multiple managers want to receive notifications at the same well-known port (162) at the same IP address.

Definitions

The SNMP agent, subagents and clients must be configured to TCP/IP before use. If the NetView SNMP client is used, Netview configuration is also required.

Though the SNMP Agent can be started with no configuration files, to implement settings or security other than the defaults, several configuration data sets are required. Most of the configuration data can be configured in several places. Details on the syntax of the statements in the files and the search orders for the files are in the z/OS Communications Server: IP Configuration Reference. In the text that follows, uppercase file names (such as OSNMPCONF) indicate the generic name for the file, which can be any of the places in the search order for the file.

TCP/IP configuration files for SNMP are summarized below. For use of the NetView SNMP command, changes are required for the NetView start procedure and the DSIDMN and DSICMD NCCFLST members of the NetView DSIPARM data set. For additional information, refer to the z/OS Communications Server: IP Configuration Guide.

z/OS UNIX snmp command
To use snmp, the following files might be needed:

- OSNMPCONF
  Defines configuration data for sending SNMPv1, SNMPv2, and SNMPv3 requests to SNMP agents. You can name this file as either a z/OS UNIX file system file or an MVS data set (partitioned or sequential).

- MIBS.DATA
  Defines textual names for user variables not included in the compiled MIB shipped with the product. You can name this file as either a z/OS UNIX file system file or an MVS data set (partitioned or sequential).

SNMP agent
The SNMP agent (snmpd) uses the following configuration data sets:
OSNMPD.DATA
Defines initial settings for some MIB variables supported by the agent.

PW.SRC
Defines community names, if the SNMPD.CONF file is not being used. Note that community name is a mixed-case, case-sensitive field.

SNMPD.BOOTS
Defines SNMPv3 initialization parameters to the SNMP agent if SNMPv3 security is used.

SNMPD.CONF
Defines security configurations and trap destinations to the SNMP agent. Required if SNMPv3 security is used. Can also be used for community-based (SNMPv1 and SNMPv2c) security.

SNMPTRAP.DEST
Defines trap destinations, if the SNMPD.CONF file is not being used.

With z/OS Communications Server, the SNMP agent allows the use of user-based security (SNMPv3) in addition to, or instead of, community-based security (SNMPv1 and SNMPv2c).

The choice of configuration data sets depends on the security methods chosen, as shown in Table 51.

Table 51. Configuration files and security types

<table>
<thead>
<tr>
<th>Data set</th>
<th>SNMPv1 and SNMPv2c</th>
<th>SNMPv1, SNMPv2c, and SNMPv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>PW.SRC</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SNMPTRAP.DEST</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>OSNMPD.DATA</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SNMPD.CONF</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SNMPD.BOOTS</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

TCP/IP subagent
The TCP/IP subagent is controlled by statements in the TCP/IP profile. The following statements are particularly important:

SACONFIG
Defines configuration parameters for the TCP/IP subagent.

ITRACE
Specifies the level of tracing used by the TCP/IP subagent.

OMPROUTE subagent
The SNMP OMPROUTE subagent is controlled by statements in the OMPROUTE configuration file. The following statements are particularly important:

ROUTESA_CONFIG
Defines configuration parameters for the OMPROUTE subagent.
You can also use the command MODIFY ROUTESA.
**OMPROUTE start option -s n**

Specifies the level of tracing used by the OMPROUTE subagent. You can also use the MODIFY SADEBUG command.

**OSPF_INTERFACE**

Defines an OSPF interface. The OMPROUTE subagent supports only OSPF MIB (RFC 1850).

**Note:** At least one OSPF_INTERFACE must be defined.

**Network SLAPM2 subagent**

The Network SLAPM2 subagent is controlled by start options specified when the subagent is started. The following options are particularly important:

**NSLAPM2 start option -c community**

Defines the community name to be used in connecting to the SNMP agent.

**NSLAPM2 start option -P port**

Defines the port to be used in connecting to the SNMP agent.

**NSLAPM2 start option -d n**

Specifies the level of tracing used by the Network SLAPM2 subagent. You can also use the MODIFY DEBUG,LEVEL command.

**TN3270E Telnet subagent**

The TN3270E Telnet subagent is controlled by the TNSACONFIG Profile statement. Refer to z/OS Communications Server: IP Configuration Reference for a detailed description of this statement.

**SNMP socket call settings**

Finally, SNMP makes socket calls that require correct settings in the TCPIP.DATA file. Statements used by SNMP include:

**DATASETPREFIX**

Can be used in determining the high-level qualifier for agent configuration data sets.

**TCPIPJOBNAME**

Determines the TCP/IP instance in which SNMP attempts to establish its relationship through the SETIBMOPT socket call. For more information about TCPIPJOBNAME, refer to the z/OS Communications Server: IP Configuration Reference.

**Trap forwarder daemon**

The Trap Forwarder daemon is controlled by the TRAPFWD.CONF file. TRAPFWD.CONF defines the configuration data to forward trap datagrams received on a port to other management applications listening on different ports.

**Diagnosing SNMP problems**

Problems with SNMP are generally reported under one of the following categories:

- "Abends” on page 610
- Connection problems
  - "Problems connecting the SNMP agent to the TCP/IP address space” on page 610
  - "Problems connecting SNMP agents to multiple TCP/IP stacks” on page 611
Abends

An abend during SNMP processing should result in messages and error-related information being sent to the system console. A dump of the error is needed unless the symptoms match a known problem.

Documentation

Code a CEEDUMP DD statement in the PROC used to start the SNMP agent to ensure that a useful dump is obtained in the event of an abend.

Analysis

Refer to z/OS Problem Management or Chapter 3, “Diagnosing abends, loops, and hangs,” on page 25, for information about debugging dumps produced during SNMP processing.

SNMP connection problems

This section describes how to diagnose and correct SNMP connection problems.

Problems connecting the SNMP agent to the TCP/IP address space

Problems connecting the SNMP agent to the TCP/IP address space are usually indicated by an error message in the agent traces in the syslog daemon output, indicating a socket error. For more information on reading the syslogd traces, refer to the z/OS Communications Server: IP Configuration Guide.

Documentation: The following documentation should be available for initial diagnosis of problems connecting the SNMP agent to the TCP/IP address space:

- PROFILE.TCPIP information
- SNMP agent tracing (at level 255) to the syslog daemon output
- TCP/IP.DATA information
- OMVS console output for any command responses and traces
Analysis: Use the following checklist to check for problems connecting the SNMP client or agent address space to the TCP/IP address space:

- Are you connected to the correct TCP/IP address space? This is obviously a concern when running multiple stacks. See "Problems connecting SNMP agents to multiple TCP/IP stacks."

If you get a message "unable to connect to TCPIP JOBNAME," you are not connected to the correct address space. If you have defined two or more stacks, make sure your TCPIPjobname in the TCPIP.DATA data set used by the SNMP agent matches the NAME field on the SUBFILESYSTYPE statement for ENTRYPOINT(EZBPFIN) in the BPZPRMxx member you used to start z/OS UNIX MVS.

- Did any socket-related errors occur?

Check the SNMP agent syslogd for socket(), bind(), accept(), or other socket error messages. For example, a bind() failure occurs when one or more of the ports needed by the SNMP agent is already in use. Refer to the z/OS Communications Server: IP Configuration Guide for more information about syslogd.

- Is the correct TCPIP.DATA information being used? Is the SYSTCPD DD statement coded in the PROC JCL? Is the RESOLVER_CONFIG environment variable passed on the SNMP agent initialization parameters?

If the problem still occurs after checking the preceding items and making any needed changes, obtain the following documentation for problems connecting the agent.

- Dump of SNMP agent address space.
- Dump of TCP/IP address space.
- The syslogd traces from the agent (using trace level 255). Refer to the z/OS Communications Server: IP Configuration Guide for more information about reading the syslogd.

Information on obtaining a dump can be found in the z/OS MVS Diagnosis: Tools and Service Aids manual for your release of MVS. Obtaining SNMP traces is discussed in "SNMP traces” on page 627.

Problems connecting SNMP agents to multiple TCP/IP stacks

To receive TCP/IP related management data, each TCP/IP stack that is started must run its own SNMP agent. This requires that each agent can find the TCP/IP job name of the TCP/IP stack that it wants to associate with.

Analysis: Use the following checklist to check for problems connecting the SNMP agent to the correct TCP/IP stack:

- Message EZZ6205I indicates that when _iptcpn() was called, it did not return the correct TCPIPjobname for that agent.
  - Check _iptcpn()'s search path.
  - Check to see if the _BPXK_SETIBMOPT_TRANSPORT environment variable has been set in the cataloged procedure.

Refer to the z/OS Communications Server: IP Configuration Reference for additional information.

- Message EZZ6272I indicates that the setibmopt call failed. This means that _iptcpn() returned a name that z/OS UNIX did not recognize as a PFS. Check the BPXPRMxx member (in SYS1.PARMLIB) used to configure z/OS UNIX.
Problems connecting subagents to the SNMP agent

Problems connecting an SNMP subagent to the SNMP agent are generally indicated by one of the following:

- A socket error at the subagent.
- Authentication failures when the subagent attempts to open a connection.
- A “no such name” response from the SNMP agent when an SNMPv1 manager requests a variable owned by the subagent.
- A “no such object” response from the SNMP agent when an SNMPv2 or SNMPv3 manager requests a variable owned by the subagent.

Documentation:
The following documentation should be available for initial diagnosis of interface connection problems:

- **PROFILE.TCPIP** information.
- **SNMP agent** job output, including syslogd output.
- Documentation for the subagent which is not connecting, as follows:
  - TCP/IP subagent syslogd output obtained by specifying the profile statement ITRACE ON SUBAGENT 2 (if the subagent is the TCP/IP subagent).
  - Output of the Netstat HOME/-h command.
  - TCPIP.DAT informatio.
  - OMPROUTE subagent syslogd output obtained by starting OMPROUTE with the -s1 option or by issuing the MODIFY SADEBUG command to start OMPROUTE subagent tracing (if the subagent is the OMPROUTE subagent).
  - Network SLAPM2 subagent syslogd output obtained by starting the Network SLAPM2 agent with the -d 131 option or by issuing the MODIFY DEBUG,LEVEL command to start Network SLAPM2 subagent tracing (if the subagent is the Network SLAPM2 subagent). The value 131 for -d turns on the following traces.
    1. Trace Network SLAPM2 Subagent Error and System Console Messages
    2. Trace Network SLAPM2 Subagent Warning Message
    128. Trace DPIdebug()level 2
  - TN3270E Telnet subagent syslogd output obtained by specifying the TNSATRACE keyword on the TNSACONFIG profile statement.

Analysis:
Use the following checklist to check for problems connecting an SNMP subagent program to the SNMP agent:

1. Are there multiple TCP/IP stacks active on the same MVS image, are there subagents active for each stack, and are the subagents using UNIX to connect to the agent (as opposed to using TCP)? If so, have you configured a unique UNIX pathname to be used by the subagents connecting to the Agent through UNIX? In a multi-stack environment, each Agent must use a unique UNIX pathname for subagent connections. The default UNIX pathname is /var/dpi_socket. Additional UNIX pathnames can be specified in one of two ways:
   - As the value of the dpiPathNameForUnixStream MIB object in the OSNMPD.DATA configuration file read by the Agent.
   - On the -s start option in the PARM= field of the EXEC JCL statement in the Agent’s started procedure.

2. Is the subagent in question the TCP/IP subagent? If so,
   - Is the SACONFIG statement configured correctly?
3. Is the subagent in question the OMPROUTE subagent?
   - Is the OMPROUTE ROUTESA_CONFIG statement configured correctly?
   - Is the OMPROUTE subagent (ROUTESA) disabled?
   - Does the port number match the SNMP agent and OMPROUTE application for the OMPROUTE ROUTESA_CONFIG parameter AGENT=<agent port number>?
   - Does the community name (or password) match with the SNMP agent and OMPROUTE application for the OMPROUTE ROUTESA_CONFIG parameter COMMUNITY=<community string>?

4. Is the subagent in question the Network SLAPM2 subagent?
   - Does the port number specified on the -P parameter of the Network SLAPM2 subagent match the port number specified by the SNMP agent?
   - Does the community name (or password) specified on the -c parameter of the Network SLAPM2 subagent match the community name (or password) specified by the SNMP agent?

5. Is the subagent in question the TN3270E Telnet subagent? If so:
   - Is the TNSACONFIG statement configured correctly?
   - Is TNSACONFIG DISABLED specified?

6. If you are using an hlq.HOSTS,SITEINFO file (or its z/OS UNIX file system equivalent, /etc/hosts), you must ensure that the IP address in this file for the system on which the agent/subagent are executing matches an interface IP address of the TCP/IP stack to which the agent/subagent are connected. The interface IP addresses for a TCP/IP stack are defined on the HOME profile statement.

7. Is the subagent using the correct IP address to send the connection request to the SNMP agent? The subagent uses the IPv4 primary interface IP address of this stack when sending the connection request to the SNMP agent. The IPv4 primary interface IP address is either the first IP address in the HOME list or the IP address specified on a PRIMARYINTERFACE TCP/IP profile statement. Check the Netstat HOME/-h output to verify the IPv4 primary interface address of the stack. This IP address is the one that is used by the SNMP agent, along with the community name to verify the subagents authority to connect to the SNMP agent.

8. Is the port number correct?

9. Is the community name (or password) correct?
   
   **Guideline:** Community name is a mixed-case, case-sensitive field. Many times the client cannot get a response from an agent because the agent has a community string of PUBLIC. Most clients default their community string to public.

10. If the SNMP agent is configured for SNMPv3, is the community name configured in the agent SNMPD.CONF file? The subagent can use the community name only if VACM_GROUP, VACM_VIEW, and VACM_ACCESS are defined. For the subagent to connect, the VACM_VIEW must include the dpiPort objects.

11. Did any socket-related errors occur?
   
   Check the SNMP agent/subagent syslogd for socket(), bind(), accept(), or other socket error messages, particularly error messages related to the DPI connection.

12. If the subagent is using TCP to connect to the SNMP agent then the connection could have been closed by the agent due to a security
authorization failure. If the agent security resource name has been defined in the SERVAUTH class, then the subagent must be running on the same TCP/IP stack as the agent and the user ID of the subagent must be permitted to the resource name in order for the connection to succeed. See the SNMP information of the z/OS Communications Server: IP Configuration Guide for a description of the agent security resource name used with TCP connections between SNMP agent and subagent.

If the problem still occurs after checking the preceding items and making any needed changes, obtain the following documentation:

- SNMP agent 255 (trace all) output.
- If the problem is with the TCP/IP subagent, get the subagent traces (level 2). These are turned on by specifying the ITRACE statement in the PROFILE.TCPIP file. This can be done as part of the initial TCP/IP startup. It can also be done after TCP/IP has been started by using the VARY TCPIP command, which is documented in the z/OS Communications Server: IP System Administrator's Commands.
- If the problem is with the OMPROUTE subagent, get the OMPROUTE subagent traces. Turn these on by starting OMPROUTE with the -s1 option or by issuing the MODIFY SADebug command to start OMPROUTE subagent tracing.
- If the problem is with the Network SLAPM2 subagent, get the Network SLAPM2 subagent traces. Turn these on by starting the Network SLAPM2 subagent with the -d 131 option or by issuing the MODIFY DEBUG,LEVEL command to start Network SLAPM2 subagent tracing.
- If the problem is with a user-written subagent program, use the DPIdebug() DPI library routine to collect dpi traces in the user-written subagent program. DPIdebug() sends output to the syslogd.
- If the problem is with the TN3270E Telnet subagent, get the subagent traces. These are turned on by specifying the TNSATRACE keyword on the TNSAConfig statement in the PROFILE.TCPIP file. This can be done as part of the initial TCP/IP startup. It can also be done after TCP/IP has been started by using the VARY TCPIP,OBEYFILE command, which is documented in the z/OS Communications Server: IP System Administrator's Commands. In order to enable tracing using the VARY TCPIP,OBEYFILE command, the subagent must first be disabled and then re-enabled with the TNSATRACE keyword.

The following is a list of things to look for in the SNMP agent trace:

- One of the following incoming SNMP GetRequest-PDU:
  - dpiPortForTcp (1.3.6.1.4.1.2.2.1.1.1) for TCP connect. This is caused by DPIconnect_to_agent_TCP
  - dpiPathNameForUnixStream (1.3.6.1.4.1.2.2.1.1.3) for UNIX connect. This is caused by DPIconnect_to_agent_UNIXstream.

Some questions to consider:

- Was the GetRequest-PDU received? If the GetRequest was not received, was it sent to the right port?
  In the case of the TCP/IP subagent, the value of the AGENT keyword on the SACONFIG statement in the profile must match the value of -p that was specified (or defaulted) when the agent was invoked.
- Does it have a valid community name in the request?
  - SNMP subagents must use a valid (including correct case) community name as defined in the PWSRC data set (or SNMPD,CONF data set when using SNMPv3 security) when requesting the dpiPort or dpiPath variable.
- Note that community name is a mixed-case, case-sensitive field. Specify as follows:
  - For the TCP/IP subagent, specify the community name in the SACONFIG statement.
  - For the OMPROUTE subagent, specify the community name in the ROUTESA_CONFIG statement.
  - For the Network SLAPM2 subagent, specify the community name by way of the -c parameter.
- For the TN3270E Telnet subagent, specify the community name on the TNSACONFIG statement.
- If SNMPv3 is being used, the community name must be defined in the VACM_GROUP statement in the SNMPD.CONF file for the SNMP agent. A VACM_ACCESS statement also needs to be defined to give that group read access to a VACM_VIEW that includes dpiPort objects.
- dpiPathNameForUnixStream defaults to /var/dpi_socket and provides a z/OS UNIX file system pathname used in connecting a DPI subagent with the SNMP agent. The default can be overridden by using the -s parameter when starting the agent or by adding an entry for dpiPathNameForUnixStream in the OSNMPD.DATA file.
  A user-written subagent running from a nonprivileged user ID needs write access to the file. Otherwise, a subagent using DPIconnect_to_agent_UNIXstream() would have to be run from an OMVS superuser user ID or other user ID with the appropriate privilege.

- Outgoing GetResponse-PDU for the dpiPort variable:
  - Was the SNMP GetResponse-PDU sent back to the SNMP subagent?
  - Was it sent to the correct IP address?
  - Did it have the correct value for the DPI port?
  - The actual value for the DPI port for TCP can be determined by issuing a Netstat ALL/-A command at the SNMP agent. This displays the port on which the agent is accepting incoming UDP requests.
  - To display the dpiPath name, issue an osnmp get request for dpiPathNameForUnixStream.

- One of the following incoming subagent connections:
  - Message EZZ6244I Accepted new DPI inet subagent connection on fd
    
    fd=xx from inet address xxxxx port xxxx.
  - EZZ6246I Accepted new DPI inet socket connection on fd=xx

  **Note:** *fd=xx* is the number associated with this specific subagent connection. Use it to correlate with later DPI trace messages. The name and number of the port *xxxxx port xxxx*.

- DPI packets transferred for this FD number

The following documentation might also be needed in some cases, but it is suggested that the IBM Software Support Center be contacted before this documentation is obtained:
- Dump of SNMP agent address space
- Dump of TCP/IP address space (for TCP/IP and TN3270E Telnet subagent problems)
- Dump of OMPROUTE address space (for OMPROUTE subagent problems)
- Dump of Network SLAPM2 subagent address space (for Network SLAPM2 subagent problems)
- Dump of user SNMP subagent address space
- Trace from subagent in syslogd
Problems connecting to the SNMPIUCV subtask

Problems in connecting the SNMPIUCV subtask of NetView to the SNMP query stack address space are usually indicated by an error message at the NetView operator console in response to an SNMP request or an attempt to start the SNMPIUCV subtask.

Documentation: The following documentation should be available for initial diagnosis of problems connecting the SNMPIUCV subtask to the SNMP query stack:

- PROFILE.TCPIP data set
- SNMP query stack job output, including SYSPRINT output
- NetView log
- SNMPARMS member of DSIPARMS data set

Analysis: Check for problems connecting the SNMP query stack to the NetView SNMPIUCV subtask:

- Has the SNMP query stack job started successfully?
  - Check the SNMP query stack job output for errors. If the SNMP query stack is started successfully, you should see the message:
    ```
    SQE1001 -- SNMP Query Stack running and awaiting queries...
    ```
  - Otherwise, check for errors that might have occurred during socket processing (socket, bind, accept, select, and so on).

- Is the SNMPIUCV subtask started?
  - If not, start the subtask by issuing the command:
    ```
    START TASK=SNMPIUCV
    ```
  - from a NetView operator console.

- Was the following message received at the NetView operator console?
  ```
  SMN101W SNMP task (SNMPIUCV) found Query Stack ('name') not ready
  ```
  - Is the `name` that the SNMPIUCV subtask is trying to connect to the correct name for the SNMP query stack address space?
  - If not, check the SNMPARMS member of the DSIPARMS data set to make sure that the value specified for the SNMPQE keyword is the correct SNMP query stack address space name.

If the problem still occurs after checking the preceding items and making any needed changes, obtain the following documentation:

- SNMP query stack level-two trace output
- SNMP query stack IUCV communication trace output

The following documentation might also be needed in some cases, but it is suggested that the IBM Software Support Center be contacted before this documentation is obtained:

- Dump of SNMP query stack address space
- Dump of NetView address space

Information about obtaining a dump can be found in the z/OS MVS Diagnosis: Tools and Service Aids manual for your release of z/OS. Obtaining SNMP traces is discussed in “SNMP traces” on page 627.
Problems connecting the SNMP query stack to the TCP/IP address space

Problems connecting the SNMP query stack to the TCP/IP address space are usually indicated by an error message in the SNMP client output, indicating either a socket or IUCV error.

Documentation: The following documentation should be available for initial diagnosis of problems connecting the SNMP query stack to the TCP/IP address space:
- PROFILE.TCPIP data set
- SNMP client output, including SYSPRINT output
- TCPIP.DATA data set

Analysis: Check the following for problems connecting the SNMP client address space to the TCP/IP address space:

Use the following checklist to check the connection problems:

- Did any socket-related errors occur?
  Check the SNMP query stack job output for socket(), bind(), accept(), or other socket error messages.
- Does job output indicate RC=1011 received for IUCV_CONNECT to tcpip_name?
  Is the tcpip_name indicated by the IUCV_CONNECT error the correct name for the TCP/IP address space?
  - Is the correct TCPIP DATA data set being used? (The job output should indicate which data set is being used).
    - Is the SYSTCPD DD statement coded in the PROC JCL?
      Tip: SYSTCPD can be overridden by the global TCPIP.DATA file. Refer to the z/OS Communications Server: IP Configuration Reference for additional information about the search order for the TCPIP.DATA data set.
  - Does the TCPIPJOBNAME keyword in the TCPIP.DATA data set being used have the correct TCP/IP address space name?

If the problem still occurs after checking the preceding items and making any needed changes, obtain SNMP query stack IUCV communication trace output for problems connecting the client.

The following documentation might also be needed in some cases, but it is suggested that the TCP/IP IBM Software Support Center be contacted before this documentation is obtained:
- Dump of SNMP client address space
- Dump of TCP/IP address space

Information on obtaining a dump can be found in the z/OS MVS Diagnosis: Tools and Service Aids manual for your release of z/OS. Obtaining SNMP traces is discussed in “SNMP traces” on page 627.

Incorrect output

Unknown variable
Unknown variable problems are indicated by a noSuchName or noSuchObject response on an SNMP request. The noSuchName response indicates an error returned on an SNMPv1 request. For SNMPv2 and SNMPv3, more specific errors are returned, such as noSuchObject and noSuchInstance.
Unknown variable problems are usually caused by one of the following:

- A typographical error in the name or OID
- An incorrect instance number

**Guideline:** If the dot-zero (.0) version of this OID contains a non-NULL value, the getnext would return ifNumber.0 and its value. It should be noted that if the dot-zero version of the requested OID is NULL, the getnext returns the first non-NULL value encountered in the MIB tree after ifNumber.0.

- The subagent supporting the MIB object is not started or is not completely connected to the SNMP agent.
- When SNMPv3 is configured, the object is not within the MIB view the user or community can access.

When the NetView SNMP client is used, unknown variable problems are reported when the SNMP client receives either a major error code 2 (internally detected error), minor error code 7 (unknown MIB variable), or a major error code 1 (SNMP agent reported error), minor error code 2 (no such name) in response to an SNMP request.

**Documentation:** The following documentation should be available for initial diagnosis of unknown variable problems:

- SNMP syslogd output with traces for both the agent and subagent. Refer to the [z/OS Communications Server: IP Configuration Guide](https://www.ibm.com/support/docview/cn/en/6669716) for more information about syslogd.
- MIBS.DATA, when snmp is used.
- SNMP query stack job output, when NetView SNMP is used.
- NetView log, when NetView SNMP is used.
- hlq.MIBDESC.DATA data set, when NetView SNMP is used.
- If SNMPv3 security is being used, the SNMP agent configuration file (SNMPD.CONF). If the snmp command is the client being used, the snmp command configuration file (OSNMP.CONF) might also be needed.
- Include all the configuration files described earlier under “Definitions” on page 607.

**Analysis:** Use the following checklist to check for unknown variables at the SNMP agent:

1. Was the variable requested with the correct instance number?
   
   Variables that are not in a table have an instance number of 0. Variables that are part of a table might have more than one occurrence of the variable value. To get the value of the variable, you need to request a specific instance of the variable. To find the instance number, issue a GET NEXT request; the first occurrence of the variable should be returned.

2. If the variable is not defined in any compiled MIB, is the variable name included in the MIBS.DATA file (for the snmp command) or the hlq.MIBDESC.DATA file (for the Netview SNMP command)?

3. Did the DPI connection come up successfully?
   
   a. Check the SNMP agent job output for messages indicating a problem in create_DPI_port.
   
   b. If the DPI port was not successful, no SNMP subagents are able to register MIB variables. The SNMP agent has no knowledge of these unregistered variables and reports them as “noSuchName” for SNMPv1 requests or “noSuchObject” for SNMPv2 and SNMPv3 requests.

4. Has the subagent successfully connected to the SNMP agent?
a. For subagents shipped as part of z/OS Communications Server, check the MVS operator console for a message indicating that the subagent has completed its initialization.

b. Issue an `snmp walk` command on the SNMP agent subagent status table. For example, either of the following commands display the subagents that are connected to the z/OS Communications Server SNMP agent and the status of their connections:
   • `snmp -v walk saDescription`
   • `snmp -v walk saStatus`

   A value of 1 for saStatus indicates that the subagent connection to the SNMP agent is valid. Following are other possible status values:
   
   invalid (2)
   connecting (3)
   disconnecting (4)
   closedByManager (5)
   closedByAgent (6)
   closedBySubagent (7)
   closedBySubAgentTimeout (8)
   closedBySubAgentError (9)

__5. If the SNMP agent was configured with SNMPv3 security, is the object within the MIB view of that allowed for that user or community?

   a. Look at the agent SNMPD.CONF file to determine to which VACM_GROUP the user or community name on the failing request belongs. Then examine the VACM_ACCESS statements for that group for the level of security requested on the failing request to determine which MIB views have been permitted to the user or community name.

   b. Alternatively, SNMP agent configuration can be determined from SNMP agent traces if they were set to level 255 at agent initialization.

   c. SNMP agent configuration can also be determined dynamically by issuing `snmp walk` requests against the agent configuration MIB objects, such as the vacmSecurityToGroupTable and the vacmAcessTable. Reading the values in these tables requires an understanding of how the tables are indexed. Refer to Requests for Comments (RFCs) 2573, 2574, and 2575 for an explanation of the MIB objects containing the SNMP agent configuration.

If the problem still occurs after checking the preceding items and making any needed changes, obtain the following documentation:

For variable not recognized by manager messages:
   • If the manager is the `snmp` command, use the `-d 4` flag to get level four manager traces.
   • If the manager is the NetView SNMP command, obtain the SNMP query stack level two output. The SNMP query stack level two trace shows the information flowing between the SNMP query stack and the SNMPIUCV subtask of NetView. Verify in the trace that the variable name being requested is being passed correctly to the SNMP query stack.

For agent unknown variable:
   • SNMP agent level 15 trace output
   • Traces from SNMP subagent programs (if the variable is supported by a z/OS Communications Server subagent)
The SNMP agent level 15 trace shows PDUs between the manager and agent, as well as between the agent and any existing subagents. Look for the following in the trace:

- Is the ASN.1 number received from the manager in the SNMP GetRequest-PDU correct?
- Has a DPI packet registering the requested variable been received?
  1. If not, if you know which subagent program owns the variable, check the subagent program for errors.
  2. If the DPI register has been received, make a note of the FD number for further trace information.
- Were any errors reported for this FD number after the DPI register request was received?
- Was there a DPI information exchange over this FD number as a result of the incoming SNMP GetRequest-PDU?

Another approach to this problem is to look at the agent saMIB variables. This information can be useful when traces are not available. The saMIB variables include the following information:

- An entry for each subagent (including a field for subagent status)
- A table of all trees registered, including:
  - Subagent to which the tree is registered
  - Status of the tree (valid, not valid, and so on)

A description of the saMIB objects can be found in the file samib.mib in the /usr/lpp/tcpip/samples directory.

The following documentation might also be needed in some cases, but it is suggested that the IBM Software Support Center be contacted before this documentation is obtained:

- Dump of SNMP agent address space
- Dump of the subagent responsible for the MIB object whose value is being returned incorrectly.
  - Dump of TCP/IP address space (for TCP/IP and TN3270E Telnet subagent variables)
  - Dump of OMPROUTE address space (for OMPROUTE subagent problems)
  - Dump of Network SLAPM2 subagent address space (for Network SLAPM2 subagent problems)
  - Dump of user subagent address space

Information on obtaining a dump can be found in z/OS MVS Diagnosis: Tools and Service Aids. Obtaining SNMP traces is discussed later in "SNMP traces" on page 627.

**Variable format incorrect**

Problems with incorrectly formatted variables are generally reported when the variable value from the GetResponse-PDU is displayed at the manager in the incorrect format (for example, as hexadecimal digits instead of a decimal value or a display string).

**Documentation:** The following documentation should be available for initial diagnosis of incorrectly formatted variables:

- MIBS.DATA, when snmp is used
- NetView log, when the NetView SNMP command is used
- hlq.MIBDESC.DATA data set, when NetView SNMP is used
Analysis: Use the following checklist to check incorrect variable format:

__ 1. Is the variable contained in the hlq.MIBDESC.DATA data set or MIBS.DATA file?
   a. The SNMP query stack uses the hlq.MIBDESC.DATA data set to determine the display syntax of the variable value. NetView SNMP requires that all MIB object names be included in the hlq.MIBDESC.DATA data set.
   b. snmp searches the MIBS.DATA file for a MIB name definition. If it is not found, the value in the compiled MIB is used.

__ 2. Is the value listed in the syntax position of the hlq.MIBDESC.DATA data set or MIBS.DATA file record for this variable the correct syntax?
   Note that the value specified for syntax (for NetView) is case-sensitive and must be specified in lowercase.

__ 3. For NetView SNMP, is the variable value type specified in message SNM043I Variable value type: correct?
   Refer to the z/OS Communications Server: IP System Administrator’s Commands section about “Managing TCP/IP Network Resources Using SNMP” for the meanings of the variable value types.

If the problem still occurs after checking the preceding and making any needed changes, obtain the following documentation:

• For the TCP/IP subagent, subagent ITRACE level four output to show that the subagent returned to the SNMP agent.
• For the OMPROUTE subagent, syslogd output obtained by starting OMPROUTE with the -s1 option or by issuing the MODIFY SADEBUG command to start OMPROUTE subagent tracing.
• For the Network SLAPM2 subagent, syslogd output obtained by the Network SLAPM2 subagent with the -d 131 option or the MODIFY DEBUG,LEVEL command to start Network SLAPM2 subagent tracing.
• For user-written subagents, DPIdebug(2) output, which is sent to the syslogd. For more information on reading the syslogd traces, refer to the z/OS Communications Server: IP Configuration Guide.
• SNMP query stack level four trace output or snmp command trace level four.
• SNMP manager command output showing incorrectly formatted variable.
• SNMP agent level 31 trace output shows the DPI packet exchanges between the agent and subagent, as well as the value returned to the manager.
• For the TN3270E Telnet subagent, syslogd output from TNSATRACE keyword on TNSACONFIG profile statement to show what the subagent returned to the SNMP agent.

In the traces, verify that the variable value and syntax are passed correctly in the SNMP GetResponse-PDU from the agent to the SNMP manager.

The following documentation might also be needed in some cases, but it is suggested that the IBM Software Support Center be contacted before this documentation is obtained:

• Dump of the TCP/IP address space (for TCP/IP and TN3270E Telnet subagent problems)
• Dump of SNMP agent address space
• Dump of SNMP query stack address space
• Dump of OMPROUTE address space (for OMPROUTE subagent problems)
• Dump of Network SLAPM2 subagent address space (for Network SLAPM2 subagent problems)
Information on obtaining a dump can be found in z/OS MVS Diagnosis: Tools and Service Aids. Obtaining SNMP traces is discussed in “SNMP traces” on page 627.

Variable value incorrect
Problems with incorrect variable values are generally reported when the variable value from the GetResponse-PDU displayed at the manager contains incorrect information.

Documentation: The following documentation should be available for initial diagnosis of variables with incorrect values:

- SNMP agent syslogd trace output.
- If the object is supported by the TCP/IP subagent, the syslogd output. Obtain the syslogd output using the profile statement ITRACE ON SUBAGENT 4.
- MIBS.DATA, if snmp is being used.
- hlq.MIBDESC.DATA, if NetView SNMP is being used.
- NetView log, if NetView SNMP is used.

Analysis: Use the following checklist to check for incorrect variable value:

__ 1. Is the object identifier in the MIB description file correct?
__ 2. Were any errors reported at the SNMP agent when the variable was requested?
__ 3. Is the variable being cached at the SNMP query stack?
   The SNMP query stack uses the hlq.MIBDESC.DATA data set to determine
   the length of time to cache the variable value (or a default time length if the
   variable is not found in the hlq.MIBDESC.DATA data set). If the variable is
   requested before the caching time is up, the cached value is used instead of
   obtaining a new value.
__ 4. Is the variable cached at the TCP/IP subagent?
   The TCP/IP subagent caches variable values for the length of time specified
   by the ibmMvsSubagentCacheTime MIB object, set by default to 30 seconds.
__ 5. Is the variable supported by the Network SLAPM2 subagent? If so, is it being cached? The Network SLAPM2 subagent caches MIB objects for 30 seconds by default, but the cache time can be overridden at subagent initialization time with the -t parameter.
__ 6. Is the variable cached at the TN3270E Telnet subagent? The TN3270E Telnet subagent caches variable values for the length of time specified by the CACHETIME keyword on the TNSACONFIG profile statement, set by default to 30 seconds.

If the problem still occurs after checking the preceding items and making any needed changes, obtain the following documentation:

- SNMP agent level three showing what was returned to the client.
- For the TCP/IP subagent, ITRACE level four trace output showing what the subagent returned to the SNMP agent.
- For the OMPROUTE subagent, syslogd output obtained by starting OMPROUTE with the -s1 option or by issuing the MODIFY SADEBUG command to start OMPROUTE subagent tracing.
- For the Network SLAPM2 subagent, syslogd output obtained by the Network SLAPM2 subagent with the -d 131 option or the MODIFY DEBUG,LEVEL command to start Network SLAPM2 subagent tracing.
- For user-written subagents, DPIdebug(2) output which is sent to the syslogd. For more information on reading the syslogd traces, refer to the z/OS Communications Server: IP Configuration Guide.
For the TN3270E Telnet subagent, syslogd output from the TNSATRACE keyword on the TNSACONFIG profile statement showing what the subagent returned to the SNMP agent.

In the traces, verify that the variable value is passed correctly from the SNMP subagent to the SNMP agent and from the SNMP agent to the client.

The following documentation might also be needed in some cases, but it is suggested that the IBM Software Support Center be contacted before this documentation is obtained:
- Dump of TCP/IP address space (for TCP/IP and TN3270E Telnet subagent variables).
- Dump of SNMP query stack address space
- Dump of OMPROUTE address space (for OMPROUTE subagent problems)
- Dump of Network SLAPM2 subagent address space (for Network SLAPM2 subagent problems)
- Incorrect values from the TCP/IP subagent are probably due to an error in the TCP/IP stack. In this case, a dump of the TCP/IP address space and a CTRACE from the stack might be useful. You can also use the Netstat command to verify that the TCP/IP subagent is reporting what the TCP/IP stack believes the value to be.

Information on obtaining a dump can be found in *z/OS MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP traces is discussed in "SNMP traces" on page 627.

No response from the SNMP agent

Problems receiving a response from the SNMP agent are generally reported when an SNMP request is issued from a manager but no response from the agent is received. This is usually reported as a timeout message.

Documentation

The following documentation should be available for initial diagnosis when no response is received from the agent:
- If the z/OS UNIX snmp command is being used, any command responses and traces for TSO user ID output from the z/OS UNIX shell.
- If the NetView SNMP command is being used, Netview console output or command responses.
- SNMP agent syslogd trace output at trace level 35.
- The OSNMP:CONF file (if the z/OS UNIX snmp command is the manager).
- PW.SRC or SNMPD.CONF file being used by the SNMP agent.

Analysis

Use the following checklist when no response is received from an agent:

__ 1. Is the SNMP agent running?
__ 2. Is a path to the agent available? Try issuing a PING request to the IP address of the agent.
__ 3. What is the timeout value? For example, the timeout value on the snmp command defaults to three seconds. Trying the request again with a larger timeout value, such as 15 seconds, might result in an answer.
__ 4. Does the request use the correct port number and IP address?
__ 5. Were any errors reported at the SNMP agent when the variable was requested?
6. If community-based security is being used, is the correct community name (including correct case) being used in the request?

7. Is the community name defined for the IP address from which the request originates? For example, a community name defined only for IPv4 addresses is not be usable from an IPv6 address.

8. Is the community name defined for the SNMP version of the request? If the PW.SRC file is being used for community name definitions, community names are usable for both SNMPv1 and SNMPv2c requests. If the SNMPD.CONF file is being used for community name definitions, separate definitions are required to allow the use of the community name for both SNMPv1 and SNMPv2c requests. Note that the \texttt{snmp} command defaults to sending SNMPv1 requests. To send an SNMPv2c request using the \texttt{snmp} command, an entry is required in the OSNMP\_CONF file and the \texttt{snmp} command must be issued with a -h value that refers to an entry in the OSNMP\_CONF file.

9. Does the agent support the SNMP version of the request? The z/OS Communications Server supports SNMPv1, SNMPv2c and, if configured with SNMPD\_CONF, SNMPv3.

If the problem still occurs after checking the preceding items and making any needed changes, obtain SNMP agent level seven trace output documentation.

Check the following in the SNMP agent traces:
1. Was the SNMP request PDU received by the agent?
2. Did it have a valid community name? Note that community name is case-sensitive and mixed-case.
3. Was the IP address of the manager the expected IP address?
4. Was an SNMP GetResponse-PDU sent back to the manager?
5. Was an AuthenticationFailure trap generated?

Guideline: For these traps to be generated, you must first provide the trap destination information in either the SNMPTAP\_DEST or SNMPD\_CONF file. Then, provide OSNMP\_DATA information where the snmpEnableAuthenTraps MIB object is set to 1, to enable the authentication traps. For detailed information on enabling traps, refer to the \textit{z/OS Communications Server: IP Configuration Reference}.

The following documentation might also be needed in some cases, but contact the IBM Software Support Center before this documentation is obtained:
- Dump of SNMP agent address space
- Dump of the TCP/IP address space (for TCP/IP and TN3270E Telnet subagent problems)
- Dump of OMPROUTE address space (for OMPROUTE subagent problems)
- Dump of Network SLAPM2 subagent address space (for Network SLAPM2 subagent problems)

Information on obtaining a dump can be found in \textit{z/OS MVS Diagnosis: Tools and Service Aids}. Obtaining SNMP traces is discussed in \textit{SNMP traces} on page 627.

**Report received from SNMP agent**

With SNMPv3, certain error conditions detected on a request are sent back from the SNMP agent to the SNMP manager as a report. Some reports are expected as part of normal processing, but most often they indicate an error condition.
For the `snmp` command, some reports occur during normal processing, such as the `usmStatsUnknownEngineIDs` condition, which occurs as the `snmp` command performs discovery processing to learn the SNMP stackID of the agent with which it is communicating. Normal processing reports are not displayed by `snmp` unless debug tracing is active. Reports that indicate error conditions are typically displayed using the EZZ33431 message. For example, when an attempt is made to use a USM user with an authentication key that does not match the key that is configured at the SNMP agent, the `usmStatsWrongDigests` report is received.

Figure 85 on page 632 shows the output received by an SNMP manager when the authentication key sent by an `snmp` command did not match the key defined at the agent. The command issued in the z/OS UNIX shell was:

```
$ snmp -h v374 -v walk usmUserStatus
```

EZZ33431 Report received : usmStatsWrongDigests
EZZ33011 Error return from SnmpRecvMsg()

Following are other common reports:

- **usmStatsUnknownUserNames**: Indicates a request was received for a user that is not defined at the SNMP agent.

- **usmStatsUnsupportedSecLevels**: Indicates a request was received for a defined user, but the user was not configured at the SNMP agent to use the security level specified in the request.

- **usmStatsDecryptionErrors**: Indicates an encrypted request was received at the SNMP agent, but the request could not be decrypted. This can be the result of an invalid privacy key.

### 0.0.0.0 address in traps from the SNMP agent

SNMPv1 traps contain the IP address of the originating agent encoded as part of the protocol data unit. The address field is four bytes long. If SNMPv1 traps are received from the z/OS Communications Server SNMP agent with an agent address of 0.0.0.0, it is most likely due to the fact that the agent obtained an IPv6 address for itself when it initialized. To avoid this situation, the SNMP agent can be started with the `-A` parameter to request that the SNMP agent obtain an IPv4 address for itself when initializing. Refer to the [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com/support/knowledgecenter/en/SS710_1.11.0/com.ibm.zos.zos.doc/ct_zosipv4_config tmp.html) for more information.

### I/O error for SNMP PING

NetView users can issue a PING request using SNMP PING. SNMP I/O error problems are reported when a major return code of 2 (internally-detected error) and a minor return code of 4 (some I/O error occurred) are received when issuing an SNMP PING. This type of problem is generally caused by an error in the PROFILE.TCPIP data set.

**Documentation**

The PROFILE.TCPIP data set should be available for initial diagnosis of SNMP I/O problems.

Additional documentation that might be needed later is discussed in [“Analysis” on page 626](#).
Analysis

Obtain the following documentation:
- SNMP query stack job SYSPRINT output
- SNMP query stack level two trace output
- SNMP query stack IUCV communication trace output

The following documentation might also be needed in some cases, but it is suggested that TCP/IP customer support be contacted before this documentation is obtained:
- Dump of SNMP address space
- TCP/IP packet trace

Information on obtaining a dump can be found in the z/OS MVS Diagnosis: Tools and Service Aids manual for your release of MVS. Obtaining SNMP traces is discussed “SNMP traces” on page 627.

Traps not forwarded by trap forwarder daemon

Problems with traps not getting forwarded by the trap forwarder daemon are most likely the result of configuration errors or problems in the network.

Documentation

The following documentation should be available for initial diagnosis:
- TRAPFWD.CONF file
- Trapfwd traces, level three
- Traces from the sending agent (the originator of the trap)
- Trace from the receiving client (the target of the forwarded trap)

Analysis

Use the following checklist to check for traps not forwarded by trap forwarder daemon:

__ 1. Is the target address correctly configured in the TRAPFWD.CONF file?
   If the target is designated by a host name, check the trapfwd trace to determine whether or not the hostname was correctly resolved to an IP address. If the target is designated by an IPv6 colon-hexadecimal address, then your TCP/IP stack must be running with IPv6 support. If the stack is not running with IPv6 support, then the trap forwarder daemon cannot forward traps to IPv6 listener addresses.

__ 2. Is the trap being received at the trap forwarder daemon?
   If trapfwd traces indicate the trap is not being received at the trapfwd daemon, examine traces from the SNMP agent sending the trap. Determine whether or not the SNMP agent did in fact send the trap.

__ 3. Are there network problems between the trap forward daemon and the target client?
   By issuing an SNMP GET request at the target client to the SNMP agent on the same host as the trap forward daemon, you can determine whether or not UDP packets are correctly reaching the client.

__ 4. Are the UDP packets being discarded due to congestion at the TCP/IP stack?
   If the trapfwd trace indicates that the trap is correctly being sent from the trap forwarder daemon to the target client, but the trap is not being received, consider setting NOUDPQueuelimit on the UDPCONFIG.
statement. This is used to specify that UDP should not have a queue limit and would prevent traps from being lost due to congestion.

If the above analysis does not correct the problem, the following documentation should be gathered and the IBM Software Support Center should be contacted:

- UDP packet trace on the TCP/IP stacks where the originating SNMP agent, the trap forwarder daemon, and the target client are running.

Incorrect address in forwarded trap

Documentation
The following documentation should be available for initial diagnosis:

- TRAPFWD.CONF file
- Trapfwd traces, level 3
- Traces from the sending agent (the originator of the trap)
- Trace from the receiving client (the target of the forwarded trap)

Analysis
Use the following checklist to check for an incorrect address in forwarded trap:

__ 1. What is the version of the SNMP trap?

In the case of SNMPv1 traps, the address from which the trap originated is encoded within the trap packet. A manager that needs the originating address should look into the SNMP packet to get the address.

If the address is 0.0.0.0, the most likely cause is that the trap originated at an IPv6 address. If the trap originated at the z/OS SNMP Agent, see “0.0.0.0 address in traps from the SNMP agent” on page 625.

In the case of SNMPv2 traps, the originating address is not encoded within the trap PDU. If a manager uses the address from which the trap packet is received, it would not be the originating address but the address at which the trap forwarder daemon is running. If the manager needs the originating address in the case of SNMPv2 traps, the trap forwarder should be configured to append the originating address to the trap and the manager should be capable of reading the address from the end of the received trap packet. For more information on the format in which the address is appended, refer to the z/OS Communications Server: IP User’s Guide and Commands.

__ 2. Is it a SNMPv2 trap?

Check to see if the ADD_RECVFROM_INFO is specified correctly in the TRAPFWD.CONF file. If it is not specified, then add the option to the configuration file. Note, the receiving manager must be capable of processing the RECFROM information at the end of the trap packet.

If the above analysis does not correct the problem, collect the above documentation and contact the IBM Software Support Center.

SNMP traces

There are several types of traces that can be useful in identifying the cause of SNMP problems:

- Manager traces
- Agent traces
- Subagent traces
• TRAPFWD traces

These traces are discussed in the following sections.

Starting manager traces

To obtain traces when the SNMP manager being used is the snmp command, issue snmp with the -d option. You can specify a trace level of zero to four. A trace level of zero provides no tracing, while a level four provides the most. Tracing for snmp is done on a per-request basis. Traces return to the console, but they can be redirected to a file issuing the OMVS redirect operand (>).

When NetView SNMP is being used, traces for the SNMP Query Engine can be obtained by starting the SNMP Query Engine and specifying -d trace_level where trace_level is one of the following:

1  Display errors.
2  In addition to 1, also display SNMP query stack protocol packets exchanged between the SNMP query stack and the SNMPIUCV subtask, with the exception of TRAP packets sent to NetView from the query stack.
3  In addition to 2, also display decoded SNMP protocol packets sent and received along with some additional informational messages.
4  In addition to three, display the BER-encoded packets received from NetView or from an SNMP agent. Also, add display of SNMP query stack protocol packets for TRAPs sent from the query stack to NetView.

For example:

//SNMPQE EXEC PGM=SQESERV,REGION=4096K,TIME=1440,PARM=' -d 3'

Also, the -it option can be used to obtain a trace of IUCV communication.

SNMP Query Engine trace output is sent to the SYSPRINT DD specified in the Query Engine JCL.

Starting SNMP agent traces

If agent is not running

If the SNMP agent is not already running, specify the -d parameter when you invoke the agent. You specify this parameter based on the method by which you start the SNMP agent:

• Using the start options in the JCL used to start the SNMP agent (more common). For example,

//OSNMPPD EXEC PGM=EZASNMPD,REGION=4096K,TIME=1440,PARM=' -d 8'

• Using the z/OS UNIX shell, using the snmpd command. For example:

snmpd -d 255 &

Use one of the following trace levels or a combination of them:

1  Trace SNMP requests
2  Trace SNMP responses
4  Trace SNMP traps
8  Trace DPI packets level 1
16  Trace DPI internals level 2
Internal trace (debug level 1)

Extended trace (debug level 2)

Trace DPI internals level 2

Combining trace levels: To combine trace levels, add trace level numbers. For example, to request SNMP requests (level 1) and SNMP responses (level 2), you would request trace level 3.

Trace records are sent to the file specified by the daemon.debug entry in the SYSLOG configuration file. For more information refer to the z/OS Communications Server: IP Configuration Guide.

If agent is already running

You can use the MVS MODIFY command to start and stop trace dynamically. Use of this support is restricted to the users with MODIFY command privilege.

If you start the agent from JCL, you have no difficulty knowing the procnname. However, if you start the agent from the z/OS UNIX shell, the agent generates a message to syslogd. This message indicates the job name the agent is running: this is the job name to specify on the MODIFY command.

For example, assume the procnname is OSNMPD and you want to change the trace level to 3 (tracing SNMP requests and responses). Enter:

```
MODIFY OSNMPD,trace,level=3
```

For more information on using the MVS MODIFY command, refer to the z/OS Communications Server: IP System Administrator’s Commands.

Starting TCP/IP subagent traces

To start the TCP/IP subagent traces, code the ITRACE statement in the PROFILE.TCPIP data set or in the data set specified on the VARY,TCPIP,OBEYFILE command. For more information, refer to z/OS Communications Server: IP Configuration Reference.

```
ITRACE ON SUBAGENT level
```

where level is one of the following values:

1. General subagent trace information.
2. General subagent trace information plus DPI traces.
3. General subagent trace information plus extended dump trace. This level provides storage dumps of useful information, such as storage returned by the IOCTL calls.
4. General subagent trace information, plus extended dump trace and DPI traces.

The trace output is sent to the syslogd. Trace records are found in the file specified by the daemon.debug entry in the SYSLOG configuration file. For more information refer to the z/OS Communications Server: IP Configuration Guide.

To stop TCP/IP subagent traces, code the ITRACE statement in the PROFILE.TCPIP data set or in the data set specified on the VARY TCPIP,OBEYFILE command:

```
ITRACE OFF SUBAGENT
```
For more information on the VARY command, refer to the [z/OS Communications Server: IP System Administrator’s Commands](https://www.ibm.com/support/knowledgecenter/STQHRU_1.11.0/com.ibm.zos.v1r11.langref_11.1.0/comm/cmdref/cmdvary.htm).

### Starting OMPROUTE subagent traces

To start OMPROUTE subagent tracing, start OMPROUTE with the -s1 option or issue the MODIFY SADEBUG command. Output is sent to syslogd. For details, see "Starting OMPROUTE tracing and debugging from the z/OS UNIX System Services shell" on page 759 and "Starting OMPROUTE tracing and debugging using the MODIFY command" on page 760.

### Starting Network SLAPM2 subagent traces

To start Network SLAPM2 subagent tracing, start the Network SLAPM2 subagent with the -d option or by issuing the MODIFY DEBUG,LEVEL command. Output is sent to syslogd.

The Network SLAPM2 subagent trace levels are 0-511. There are nine levels of tracing provided. Each level selected has a corresponding number. The sum of the numbers associated with each level of tracing selected is the value which should be specified as level. After the Network SLAPM2 Subagent is started, tracing options can be dynamically changed using the MVS MODIFY command.

The numbers for the trace levels are:

0  No tracing
1  Trace Network SLAPM2 Subagent Error and System Console Messages
2  Trace Network SLAPM2 Subagent Warning Messages
4  Trace Network SLAPM2 Subagent Informational Messages
8  Trace Network SLAPM2 Subagent Internal statistics table
16 Trace Network SLAPM2 Subagent Internal monitor table
32 Trace Network SLAPM2 Subagent Internal traps
64 Trace Network SLAPM2 Subagent Internal monitoring
128 Trace Network SLAPM2 Subagent Internal Policy Agent API
256 Trace DPIdebug()level 2

### Starting TN3270E Telnet subagent traces

To start the TN3270E Telnet subagent traces, code the TNSATRACE keyword on the TNSACONFIG statement in the PROFILE.TCPIP data set or in the data set specified on the VARY,TCPIP,OBEYFILE command. For more information, refer to [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com/support/knowledgecenter/STQHRU_1.11.0/com.ibm.zos.v1r11.configref_11.1.0/comm/configref/configref第一章.16.htm).

If the subagent is not currently tracing, the subagent must first be disabled. Disable the subagent by using the VARY TCPIP,OBEYFILE command where the data set for the command contains:

```verbatim
TELNETGLOBALS
   TNSACONFIG DISABLED
ENDTELNETGLOBALS
```

Then re-enable the subagent by using the VARY TCPIP,OBEYFILE command where the data set for the command contains:
TELNETGLOBALS
    TNSACONFIG ENABLED TNSATRACE
ENDTELNETGLOBALS

The trace output is sent to the syslogd. Trace records are found in the file specified by the daemon.debug entry in the SYSLOG configuration file. For more information, refer to the [z/OS Communications Server: IP Configuration Guide](z/OS_Communications_Server:_IP_Configuration_Guide).

**Starting TRAPFWD traces**

The following sections provide information about starting TRAPFWD traces.

**If TRAPFWD is not running**

If TRAPFWD is not already running, specify the -d parameter during startup. You can start the TRAPFWD trace in one of the following ways:

- Through the start options in the JCL used to start the TRAPFWD. For example,
  ```
  //TRAPFWD EXEC PGM=EZASNTRA,REGION=4096K,TIME=NOLIMIT,
  //PARM='POSIX(ON) ALL31(ON)/-d 3'
  ```
- Through OMVS, using the `trapfwd` command. For example,
  ```
  trapfwd -d 3 &
  ```

Use one of the following trace levels:

1. Minimal tracing, trace address from which the trap is received.
2. In addition to 1, trace addresses to which the trap packet is forwarded.
3. In addition to 2, trace trap packets.

Trace records are sent to the file specified by the daemon.debug entry in the SYSLOG configuration file. For more information refer to the [z/OS Communications Server: IP Configuration Guide](z/OS_Communications_Server:_IP_Configuration_Guide).

**If TRAPFWD is already running**

You can use the MVS MODIFY command to start and stop the trace dynamically. Use of this support is restricted to users with MODIFY command privilege.

If you start the trapfwd from JCL, you have no difficulty knowing the procname. However, if you start the trapfwd from OMVS, the trapfwd generates a message to syslogd. This message indicates the job name the trapfwd is running; this is the job name to specify on the MODIFY command.

For example, assume that the procname is TRAPFWD and you want to change the trace level to 3. You would enter the following:

```
MODIFY TRAPFWD,trace,level=3
```

For more information on using the MVS MODIFY command, refer to the [z/OS Communications Server: IP System Administrator's Commands](z/OS_Communications_Server:_IP_System_Administrator's_Commands).

**Trace examples and explanations**

The following examples are shown in this section:

- Agent trace
- TCP/IP subagent traces
- TRAPFWD trace
- NetView SNMP Query Engine trace
- NetView SNMP Query Engine IUCV Communication trace
SNMP agent traces

Figure 83 was produced by using `snmp get sysUpTime.0`. When the SNMP agent is tracing responses, it makes the following entry in the syslogd output file:

```
Dec 19 15:55:38 snmpagent.9.: SNMP logging data follows ==============  
Dec 19 15:55:39 snmpagent.9.: Log_type: snmpLOGresponse_out  
Dec 19 15:55:39 snmpagent.9.: send rc: 0  
Dec 19 15:55:39 snmpagent.9.: destination: UDP 127.0.0.1 port 5000  
Dec 19 15:55:39 snmpagent.9.: version: SNMPv1  
Dec 19 15:55:39 snmpagent.9.: community: public  
Dec 19 15:55:39 snmpagent.9.: ('70 75 62 6c 69 63'h)  
Dec 19 15:55:39 snmpagent.9.: addressInfo: UDP 127.0.0.1 port 5000  
Dec 19 15:55:39 snmpagent.9.: PDUtype: GetResponse ('a2'h)  
Dec 19 15:55:39 snmpagent.9.: request: 1  
Dec 19 15:55:39 snmpagent.9.: error-status: noError (0)  
Dec 19 15:55:39 snmpagent.9.: error-index: 0  
Dec 19 15:55:39 snmpagent.9.: varBind oid: 1.3.6.1.2.1.1.3.0  
Dec 19 15:55:39 snmpagent.9.: name: sysUpTime.0  
Dec 19 15:55:39 snmpagent.9.: value: TimeTicks  
Dec 19 15:55:39 snmpagent.9.: 5900 - 59.00 seconds  
Dec 19 15:55:39 snmpagent.9.: End of SNMP logging data:  
```

Figure 83. SNMP agent response trace

In the following scenario, the SNMP agent attempted to initialize, but it was not successful. The port it was using was already in use. The trace shown in Figure 84 was obtained with SNMP agent tracing set to 7.

```
Dec 19 11:57:52 snmpagent.16777227.: EZZ6235I socket function failed for  
SNMP inet udp socket; EDC5112I Resource temporarily unavailable.  
Dec 19 11:57:52 snmpagent.16777227.: ... errno = 112, errno2 =12fc0296  
```

Figure 84. SNMP agent trace of unsuccessful initialization

Note: Errno 112 translates to “Resource temporarily unavailable.” The errno is used primarily by IBM service in diagnosing the error. In this case, issue the Netstat CONN/-c command to determine if TCP/IP is running and, if so, which ports are in use.

Figure 85 shows the trace produced for the agent when the authentication key sent by a manager does not match the key defined at the agent. The command receives a report indicating usmStatsWrongDigests.

```
IDSTMVS.S0AU1104.SOURCE.S0AGY123(1624): rc=-65 (SNMP_RC_USM_WRONG_DIGEST)  
from snmp_process_message()  
```

Figure 85. SNMP messages and agent trace for nonmatching key

Figure 86 on page 633 shows the output received by an SNMP manager and the trace produced for the agent when the operator attempted to retrieve data not within the defined view. The command issued in the z/OS UNIX shell was:
```
snmp -h v374a -v get usmUserStatus.12.0.0.0.2.0.0.0.0.9.67.35.37.2.117.49  
```
The following return codes in SNMP agent traces typically indicate configuration errors:

- **SNMP_RC_NOT_AUTHENTICATED** - indicates the SNMP agent received an SNMPv1 or SNMPv2c request with a community name that was not valid for use by the IP address making the request.
- **SNMP_RC_NOT_IN_VIEW** - indicates the SNMP agent received an SNMPv3 request for a MIB object that is not defined to be accessible by the community name or user name making the request.
- **SNMP_RC_USM_UNKNOWN_USERNAME** - indicates the SNMP agent received an SNMPv3 request for a username not configured at the SNMP agent.
- **SNMP_RC_USM_WRONG_DIGEST** - indicates the SNMP agent received an SNMPv3 request for which the authentication key for the user making the request was not valid.
- **SNMP_RC_USM_DECRYPTION_ERROR** - indicates the SNMP agent received an encrypted request, but the request could not be decrypted because the encryption key for the user making the request was not valid.
- **SNMP_RC_USM_UNSUPPORTED_SECLEVEL** - indicates the SNMP agent received a request for a defined user, but the user was not configured to use the security level specified in the request.

**TCP/IP subagent trace**

When requests for MIB variables supported by the TCP/IP subagent fail with an indication that the variable is not supported (noSuchName or noSuchObject), one possibility is that the TCP/IP subagent was unable to connect to the SNMP agent.

Figure 87 illustrates a scenario where the subagent is unable to connect because the password it is using is not accepted by the SNMP agent. (The password used by the subagent is defined or defaulted on the SACONFIG statement in the TCP/IP profile.) The following traces were obtained with SNMP agent traces set to 15 and the subagent traces (as set on the ITRACE profile statement) set to 3.
NetView SNMP query engine trace

This section discusses the output produced by the SNMP query stack trace.

Figure 88 on page 635 shows an example of the output produced by the SNMP query stack trace. This trace was produced by starting the SNMP query stack address space with start option -d 4, which is the maximum amount of trace records produced. In the figure, the column labeled “trc lvl” shows the lowest trace level required to see that particular trace entry. For example, lines five through nine have a “trc lvl” of four. This means that only the -d 4 trace option shows this type of trace entry. On the other hand, lines 10 through 17 have a “trc lvl” of two. This means that trace level two or higher produces this trace information.

Guideline: The column headed “line no.” numbers the trace records for reference in the discussion that follows the figure. Neither the “trc lvl” nor the “line no.” column appear in the actual trace output.

The following sequence of events occurred to create the trace output:

1. Started the SNMP query stack address space
   Trace output lines in the range 1–3
2. Started the SNMPIUCV subtask at the NetView host (attempted connection to the query stack when started)
   Trace output line 4
3. Issued an SNMP TRAPSON request (request 1001)
   Trace output lines in the range 5–27
4. Incoming SNMP Trap-PDU received from SNMP agent
   Trace output lines in the range 28–61
5. Issued an SNMP TRAPSOFF request (request 1002)
   Trace output lines in the range 62–82
6. Incoming SNMP Trap-PDU received from SNMP agent
   Trace output lines in the range 83–104
7. Issued an SNMP GET request (request 1003)
   Trace output lines in the range 105–148
8. Received the response to request 1003
   Trace output lines in the range 149–191
9. Issued an SNMP GETNEXT request (request 1004)
   Trace output lines in the range 192–235
10. Received the response to request 1004
    Trace output lines in the range 236–278
11. Issued an SNMP SET request (request 1005)
    Trace output lines in the range 279–326
12. Received the response to request 1005
    Trace output lines in the range 327–369
13. Issued an SNMP MIBVNAME request (request 1006)
    Trace output lines in the range 370–397
14. Issued an SNMP PING request (request 1007)
    Trace output lines in the range 398–429
15. Issued an SNMP GET request for a variable name not defined in the
    hlq.MIBDESC.DATA data set (request 1008)
    Trace output lines in the range 430–462
16. Stopped the SNMPIUCV subtask of the NetView program
    Trace output line 463
Figure 88. SNMP query engine traces (Part 1 of 10)
Received following NVquery packet:

Dumping packet of 15 bytes:

00 0d 01 01 01 02 07 00 00 03 ea 00 00 03 e9

Received following SNMP_trap packet:

Dumping packet of 43 bytes:

30 29 02 01 00 04 04 4d 56 53 4c a4 1e 06 0a 2b 06 01 04 01 02 02 01 02 04 04 04 09 43 72 25 02 01 04 02 01 00 43 02 38 40 30 00

Decoded SNMP PDU:

{ version version-1, community '4d56534c'H, data {
trap {
enterprise 1.3.6.1.4.1.2.2.1.2.4,
agent-addr {
internet '09437225'H
},
generic-trap authenticationFailure,
specific-trap 0,
time-stamp 14400,

Figure 88. SNMP query engine traces (Part 2 of 10)
Received following NVquery packet:

```
00 28 01 01 01 02 01 00 00 03 eb 00 05 d4 e5 e2
d3 00 00 05 e2 d5 d4 d7 00 00 03 01 ff 01 00 0a
e2 e8 e2 e4 d7 e3 c9 d4 c5 00
```

Received following SNMP_response packet:

```
30 25 02 01 00 04 04 53 4e 4d 50 a2 1a 02 01 01
02 01 00 02 01 00 30 0f 30 0d 06 07 2b 06 01 02
01 01 03 43 02 48 a8
```

Decoded SNMP PDU:

```
{    
    version version-1, 
    community '534e4d50'H, 
    data {       
        get-request {           
            request-id 1,           
            error-status noError,           
            error-index 0,           
            variable-bindings {           
                
            name 1.3.6.1.2.1.1.3,           
            value {               
                simple {               
                    empty {}               
                }               
            }               
        }               
    }       
}      
```

Sending SNMP request to 9.67.114.37

Figure 88. SNMP query engine traces (Part 3 of 10)
```plaintext
{ 
  version version-1,
  community '534e4d50'H,
  data {
    get-response {
      request-id 1,
      error-status noError,
      error-index 0,
      variable-bindings {
        { 
          name 1.3.6.1.2.1.1.3,
          value { 
            application-wide { 
              ticks 18600
            }
          }
        }
      }
    }
  }
}

2 EZA6359I major version: 1
EZA6360I minor version: 1
EZA6361I release: 1
EZA6363I native set: EBCDIC
EZA6364I packet type: RESPONSE
EZA6367I sequence id: 1003
EZA6388I major error: 0
EZA6389I minor error: 0
EZA6390I error index: 0
EZA6391I error text len: 9
EZA6392I error text: no error
EZA6380I name length: 16
EZA6381I name: 1.3.6.1.2.1.1.3
EZA6382I value type: time ticks
EZA6384I value length: 4
EZA6387I value: 18600
}

4 EZA6292I Received following NVquery packet:
EZA6305I dumping packet of 42 bytes:
00 28 01 01 01 02 02 00 00 03 ec 00 05 d4 e5 e2
d3 00 00 05 e2 d5 d4 d7 00 00 03 01 ff 01 00 0a
c9 c6 c4 c5 e2 c3 d9 4b f0 00

2 EZA6359I major version: 1
EZA6360I minor version: 1
EZA6361I release: 1
EZA6363I native set: EBCDIC
EZA6364I packet type: GET-NEXT
EZA6367I sequence id: 1004
EZA6368I hostname len: 5
EZA6370I hostname: MVSL
EZA6371I community len: 5
EZA6373I community: SNMP
EZA6374I optional length: 3
EZA6375I max. retries: 1
EZA6376I initial timeout: 255
```

Figure 88. SNMP query engine traces (Part 4 of 10)
Decoded SNMP PDU:

```plaintext
{  
  version version-1,  
  community '534e4d50'H,  
  data {  
    get-next-request {  
      request-id 2,  
      error-status noError,  
      error-index 0,  
      variable-bindings {  
        name 1.3.6.1.2.1.2.2.1.2.0,  
        value {  
          simple {  
            empty {}  
          }  
        }  
      }  
    }  
  }  
}
```

EZA6308I sending SNMP request to 9.67.114.37

Received following SNMP_response packet:

```plaintext
{  
  version version-1,  
  community '534e4d50'H,  
  data {  
    get-response {  
      request-id 2,  
      error-status noError,  
      error-index 0,  
      variable-bindings {  
        name 1.3.6.1.2.1.2.2.1.2.1,  
        value {  
          simple {  
            string '49424d204c4353'H  
          }  
        }  
      }  
    }  
  }  
}
```

EZA6359I major version: 1
EZA6361I minor version: 1
EZA6361I release: 1

Figure 88. SNMP query engine traces (Part 5 of 10)
Figure 88. SNMP query engine traces (Part 6 of 10)
Decoded SNMP PDU:

```plaintext
{
  version version-1,
  community '534e4d50'H,
  data {
    set-request {
      request-id 3,
      error-status noError,
      error-index 0,
      variable-bindings {
        name 1.3.6.1.4.1.2.2.1.4.1,
        value {
          simple {
            number 12345
          }
        }
      }
    }
  }
}
```

Figure 88. SNMP query engine traces (Part 7 of 10)
Received following NQuery packet:
00 1b 01 01 01 02 08 00 00 03 ee 00 10 f1 4b f3
4b f6 4b f1 4b f2 4b f1 4b f1 4b f1 00

Received following NQuery packet:
00 15 01 01 01 02 0a 00 00 03 ef 00 05 d4 e5 e2

Figure 88. SNMP query engine traces (Part 8 of 10)
Figure 88. SNMP query engine traces (Part 9 of 10)
The following is an explanation of the traces in Figure 88 on page 635.

- Line 1 is an information message listing the actual name of the data set being used as the hlq.MIBDESC.DATA data set.
- Line 2 is an informational message indicating that the SNMP query stack has been successfully started.
- Line 3 is an informational message indicating the number of client connections the query stack allows. (A client connection is a connection from a program using the query stack protocol to communicate with the SNMP query stack to initiate SNMP requests. For example, the SNMPIUCV subtask of the NetView program is a client connection).
- Line 4 is an information message indicating that the SNMPIUCV subtask of the NetView program has successfully contacted the query stack.
- Lines 5–8 are the encoded packet received from the client (the SNMPIUCV subtask) by the query stack. This particular packet is the TRAPSON request.
- Lines 9–16 are the decoded SNMPIUCV request. The decoded packet indicates that this request is number 1001 (line 14), and was a TRAPSON request (line 13) for network mask 0.0.0.0 (line 15) with the desired network 0.0.0.0 (line 16).
- Lines 17–27 are the response sent back to SNMPIUCV from the query stack. The response (line 21) is to request number 1001 (line 22) and indicates that the TRAPSON request was successful (lines 23–27).
- Line 28 indicates that an SNMP Trap-PDU was received. Lines 29–32 are the actual BER encoded SNMP packet as it was received by the query stack.
- Lines 33–49 are the decoded version of the trap packet reported by lines 28–32.
- Lines 50–61 are the trap information being passed from the query stack up to the SNMPIUCV subtask to be displayed to the NetView operator. This trap is being forwarded to the NetView program because the IP address of the agent sending the trap (line 56), when ANDed with the network mask (line 15) matches the desired network (line 16) of filter number 1001 (line 55) that was set by the TRAPSON request 1001 (line 14) received previously (lines 9–16).
- Lines 62–64 show an incoming query stack packet sent from SNMPIUCV to the query stack.
- Lines 65–71 are the decoded packet received in lines 62–64. This packet is the TRAPSOFF request (line 69). It requests that trap filter 1001 (line 71) be turned off.
- Lines 72–82 are the response from the query stack to the SNMPIUCV subtask. The response indicates that the TRAPSOFF request was completed successfully (lines 78–82).
- Lines 83–87 indicate that another SNMP Trap-PDU was received from an agent.
- Lines 88–104 are the decoded Trap-PDU. Note that following this decoded PDU, there is no indication of the trap being forwarded to SNMPIUCV. This is because
the trap filter has been turned off, so the query stack receives the trap but does not forward the information to SNMPIUCV.

- Lines 105–109 indicate another request from SNMPIUCV being received by the query stack.
- Lines 110–125 are the decoded query stack request. The request from SNMPIUCV was to issue a GetRequest-PDU (line 114) to host MVSL (line 117), using community name SNMP (line 119) and requesting variable 1.3.6.1.2.1.1.3 (line 125). Lines 121–123 are the retry information that SNMPIUCV has gotten from the SNMPARMS member of the DSIPARMS data set.
- Lines 126–147 are the decoded SNMP GetRequest-PDU that the query stack has built as a result of the SNMPIUCV request received in lines 110–125. This PDU has been assigned request number 1 (line 132). This number is used to correlate the response when it is received.
- Line 148 indicates that the encoded SNMP GetRequest-PDU has been sent to the SNMP agent at the specified IP address. This should be the IP address of the host specified in line 117.
- Line 149 indicates that an SNMP GetResponse-PDU was received. Lines 150–153 are the encoded GetResponse-PDU.
- Lines 154–175 are the decoded GetResponse-PDU. This was a GetResponse (line 159) in response to request number 1 (line 160, matches up to the request number in the request, line 132). The request was completed with no errors (lines 161–162). The requested variable 1.3.6.1.2.1.1.3 (line 165) has a value of 18600 timeticks (line 168).
- Lines 176–191 are the query stack response to SNMPIUCV request number 1003 (lines 115 and 181). The response contains the information received from the agent in the GetResponse-PDU in lines 154–175.
- Lines 192–196 are the next query stack protocol requests received from SNMPIUCV by the query stack.
- Lines 197–212 are the decoded version of the query stack request. This is a GetNext request (line 201) to host MVSL (line 204) for variable 1.3.6.1.2.1.2.2.1.2.1 (line 212). The request number associated with this request is 1004 (line 202).
- Lines 213–234 are the decoded SNMP GetRequest-PDU built as a result of the query stack request received in lines 197–212. This GetRequest-PDU is request number 2 (line 219).
- Line 235 indicates that the encoded GetRequest-PDU has been sent to the requested host.
- Lines 236–240 indicate that a GetResponse-PDU has been received.
- Lines 241–262 are the decoded GetResponse-PDU. This is the response to request number 2 (line 247) for variable 1.3.6.1.2.1.2.2.1.2.1 (line 252). The value of the variable is a display string with the ASCII value of X'49424D204C4353' (line 255). The GetNext request completed successfully (lines 248–249).
- Lines 263–278 are the query stack response to SNMPIUCV request 1004 (line 268). The response contains the information from the GetResponse-PDU (lines 241–262). Note that the variable value in line 255 has been converted to the proper display format in line 278.
- Lines 279–284 are the next query stack protocol request from SNMPIUCV to the query stack.
- Lines 285–303 are the decoded query stack request. It is a SET request (line 289) to host MVSL to set variable 1.3.6.1.4.1.2.1.4.1 (line 300) to 12345 (line 303). This is request number 1005 (line 290).
- Lines 304–325 are the decoded SNMP SetRequest-PDU built as a result of the request received in lines 285–303. This is request number 3 (line 310).
• Line 326 indicates that the SetRequest-PDU has been sent to the specified host.
• Lines 327–331 indicate that a GetResponse-PDU has been received.
• Lines 332–353 are the decoded GetResponse-PDU. This PDU is the response to the SetRequest-PDU number 3 (line 338). It was completed successfully (lines 339–340) and variable 1.3.6.1.4.1.2.2.1.4.1 (line 343) was set to 12345 (line 346).
• Lines 354–369 are the query stack response to request 1005 (line 359) containing the information received in the GetResponse-PDU received in lines 332–353.
• Lines 370–373 are the next query stack request packet from SNMPIUCV.
• Lines 374–381 are the decoded query stack request. This is a MIBVNAME request (line 378) requesting the name of variable 1.3.6.1.2.1.1.1 (line 381). The request number is 1006 (line 379).
• Lines 382–397 are the query stack response (line 386) to request 1006 (line 387). The request completed successfully (lines 388–392) and the name of variable 1.3.6.1.2.1.1.1 (line 394) is sysDescr (line 397).
• Lines 398–401 are the next query stack request packet from SNMPIUCV.
• Lines 402–413 are the decoded query stack request packet. This is a PING request (line 406) to ping host MVSL (line 409). The request number is 1007 (line 407).
• Lines 414–429 are the query stack response (line 418) to request 1007 (line 419). The PING completed successfully (lines 420–424) and the round-trip response time was 76 milliseconds (line 429). Note that no SNMP PDUs were generated as a result of the PING request. The SNMP query stack uses a raw socket to send a PING to the requested host and SNMP protocols are not involved.
• Lines 430–434 are the next query stack request packet received from SNMPIUCV.
• Line 435 indicates that an error occurred while the query stack was decoding the request packet. The MIB variable name in the request was unknown to the query stack.
• Lines 435–451 are the decoded query stack request. This is a GET request (line 440). The variable name is unknown (line 451). This is request 1008 (line 441).
• Lines 452–462 are the query stack response (line 456) to SNMPIUCV request 1008 (line 457). The request was unsuccessful. The query stack returns major error code 2 (line 458), minor error code 7 (line 459), unknown variable (line 462). Note that no SNMP PDUs were generated since the query stack could not resolve the variable name.
• Line 463 indicates that the client connection (SNMPIUCV) has been terminated. This is the result of the STOP TASK=SNMPIUCV command.

NetView SNMP query engine IUCV communication trace

[Figure 89 on page 647] shows an example of the output produced by the IUCV communication trace. This trace was produced by starting the SNMP query stack address space with start option -it.
descarray is at 3985ab8, size is 4 bytes
descarray has 50 entries, entry size is 928
iucvdesc is at 32508
Rc=0 on IUCV_CLR to TCPCS, fd=-254, path=0, iprcode=0, ipmsgid=0, iucvname=00032508
ciucv_data area (ipbfadr2) is at 00000000
Rc=0 on IUCV_SET to TCPCS, fd=-254, path=0, iprcode=0, ipmsgid=0, iucvname=00032508
ciucv_data area (ipbfadr2) is at 00005480
Rc=0 on IUCV_CONNECT to TCPCS, fd=-254, path=1, iprcode=0, ipmsgid=0, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SET to TCPCS, fd=-254 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00005494
IUCV interrupt from TCPIP, fd=-254, path=1 type=2 (Connection Complete)
sock_request_inet entry parms:
  f=0 d=-254 r1=00000000 rd=0005ddfc rdL=20 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS, fd=-254, path=1, iprcode=0, ipmsgid=0, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00005494
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=0 d=-254 r1=00000000 rd=0005ddfc rdL=20 pdh=0 pdl=0
  rc=0 err=49 rpl=00000000 rpb=00000000 rpbl=0
sock_request_inet entry parms:
  f=25 d=3 r1=00000000 rd=0005db2c rdL=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS, fd=3 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS, fd=3, path=1, iprcode=0, ipmsgid=0, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=3, path=1, iprcode=0, ipmsgid=0, iucvname=00032508
Rc=0 on IUCV_SEND to TCPCS, fd=3 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS, fd=3, path=1, iprcode=0, ipmsgid=0, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=25 d=3 r1=00000000 rd=0005db2c rdL=16 pdh=0 pdl=0
  rc=3 err=0 rpl=00000000 rpb=00000000 rpbl=0
sock_request_inet entry parms:
  f=2 d=3 r1=00000000 rd=0001d0d8 rdL=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS, fd=3 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS, fd=3, path=1, iprcode=0, ipmsgid=0, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=3, path=1, iprcode=0, ipmsgid=0, iucvname=00032508
Rc=1 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=2 d=3 r1=00000000 rd=0001d0d8 rdL=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS, fd=4 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS, fd=4, path=1, iprcode=0, ipmsgid=0, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=4, path=1, iprcode=0, ipmsgid=0, iucvname=00032508
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)

Figure 89. SNMP IUCV communication traces (Part 1 of 5)
sock_request_inet return parms:
  f=25 d=4 r1=00000000 rd=0005db44 rd1=16 pdh=0 pdl=0
  rc=4 err=0 rpl=00000000 rpb=00000000 rpbl=0

sock_request_inet entry parms:
  f=2 d=4 r1=00000000 rd=0005da9c rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=4, path=1, iprcode=0, ipmsgid=10, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 0000555c
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)

sock_request_inet return parms:
  f=2 d=4 r1=00000000 rd=0005da9c rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000

sock_request_inet entry parms:
  f=25 d=5 rl=00000000 rd=0005db64 rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=5, path=1, iprcode=0, ipmsgid=11, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00005584
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)

sock_request_inet return parms:
  f=25 d=5 r1=00000000 rd=0005daa8 rd1=16 pdh=0 pdl=0
  rc=5 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_SET to , fd=6, path=0, iprcode=0, ipmsgid=0, iucvname=SNMPQE
        ciucv_data area (ipbfadr2) is at 00005480
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000

sock_request_inet entry parms:
  f=13 d=5 r1=00000000 rd=00000000 rd1=5 pdh=0 pdl=5
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=5, path=1, iprcode=0, ipmsgid=12, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 000055ac
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)

sock_request_inet return parms:
  f=13 d=5 r1=00000000 rd=00000000 rd1=5 pdh=0 pdl=5
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
Rc=0 on IUCV_SET to , fd=6, path=0, iprcode=0, ipmsgid=0, iucvname=SNMPQE
        ciucv_data area (ipbfadr2) is at 00005480
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000

sock_request_inet entry parms:
  f=25 d=7 r1=00000000 rd=0005db2c rd1=16 pdh=0 pdl=0
  rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0

Figure 89. SNMP IUCV communication traces (Part 2 of 5)
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=7, path=1, iprcode=0, ipmsgid=14, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=7 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=7 buf (ipbfadr2) is at 000055fc
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=25 d=7 r1=00000000 rd=0005db2c rd1=16 pdh=0 pdl=0
  rc=7 err=0 rpl=00000000 rpb=00000000 rpbl=0
SQE1001 -- SNMP Query Engine running and awaiting queries...
f=3 in callers rmask
fd=4 in callers rmask
fd=5 in callers rmask
fd=6 in callers rmask
fd=7 in callers rmask
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
in inetselect
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=15, iucvname=00032508
wait ecbl=5dc5c, ecccount=2
iucvposted=1073741824, waitposted=0, callposted=0
in iucvposted
Rc=1 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00005624
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
in gotmsgcomp
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
nfds=0, return=1
sock_request_inet entry parms:
  f=16 d=4 r1=00000000 rd=00000000 rd1=0 pdh=0 pdl=0
  rc=0 err=0 rpl=0005ed88 rpb=00000000 rpbl=4120
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=16, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=4 buf (ipbfadr2) is at 0000564c
IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
  f=16 d=4 r1=00000000 rd=00000000 rd1=0 pdh=0 pdl=0
  rc=0 err=0 rpl=0005ed88 rpb=00000000 rpbl=68
fd=3 in callers rmask
fd=4 in callers rmask
fd=5 in callers rmask
fd=6 in callers rmask
fd=7 in callers rmask
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
in inetselect
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_SEND to TCPCS , fd=-254, path=1, iprcode=0, ipmsgid=17, iucvname=00032508
wait ecbl=5dc5c, ecccount=2
iucvposted=1073741824, waitposted=0, callposted=0
in iucvposted

Figure 89. SNMP IUCV communication traces (Part 3 of 5)
The following sequence of events occurred to create the trace output:

The following sequence of events occurred to create the trace output:
1. Started the SNMP query stack
2. Connected to the query stack from the SNMPUCV subtask
3. Disconnected the SNMPUCV subtask from the query stack

**TRAPFWD Trace**

The trap forwarder daemon uses syslog functions to write out debug information and traces. Diagnostic data is written using "trapfwd" as identifier.

![Figure 90](image-url) illustrates a TRAPFWD trace.

Oct 15 14:06:06 trapfwd[16777250]: EZZ8420I The Trap Forwarder daemon is running as USER17
Oct 15 14:06:06 trapfwd[16777250]: Establishing affinity with the TCPIP stack
Oct 15 14:06:06 trapfwd[16777250]: Issuing setibmopt for TCPCS
Oct 15 14:06:06 trapfwd[16777250]: Checking if TCP/IP stack is enabled
Oct 15 14:06:06 trapfwd[16777250]: Reading the configuration file : /etc/trapfwd.conf
Oct 15 14:06:06 trapfwd[16777250]: Line 1 : 9.67.113.79 2162
Oct 15 14:06:06 trapfwd[16777250]: Added entry with host: 9.67.113.79 port: 2162
Oct 15 14:06:06 trapfwd[16777250]: Line 2 : 9.67.113.79 1062
Oct 15 14:06:06 trapfwd[16777250]: Added entry with host: 9.67.113.79 port: 1062
Oct 15 14:06:06 trapfwd[16777250]: Line 3 : 9.67.113.79 169
Oct 15 14:06:06 trapfwd[16777250]: Added entry with host: 9.67.113.79 port: 169
Oct 15 14:06:06 trapfwd[16777250]: Line 4 : 9.67.113.79 179
Oct 15 14:06:06 trapfwd[16777250]: Added entry with host: 9.67.113.79 port: 179
Oct 15 14:06:06 trapfwd[16777250]: Creating sockets...
Oct 15 14:06:07 trapfwd[16777250]: EZZB4091 TRAPFWD: INITIALIZATION COMPLETE
Oct 15 14:06:07 trapfwd[16777250]: Ready to receive and forward traps....

*Figure 90. TRAPFWD trace*
Chapter 26. Diagnosing Policy Agent problems

The z/OS UNIX Policy Agent (PAGENT) provides administrative control for policies. This topic provides information and guidance to diagnose Policy Agent problems, and it contains the following sections:

- "Overview"
- "QoS policy" on page 654
- "QoS policy scope" on page 654
- "Gathering diagnostic information" on page 655
- "Diagnosing Policy Agent problems" on page 657

Overview

The Policy Agent can act in any of several roles, depending on configuration options:

- The Policy Agent can act as the Policy Decision Point (PDP) on a single system, installing policies in one or more z/OS Communications Server stacks.
- The Policy Agent can act as a centralized policy server, providing PDP services for one or more remote policy clients.
- The Policy Agent can act as a policy client, retrieving remote policies from the policy server. Each stack in a Common INET (CINET) environment that is configured to the Policy Agent acts as a separate policy client.
- A single Policy Agent can act as a policy client or a policy server, but not both.

Policy Agent reads policies defined in local or remote configuration files, or reads by way of the Lightweight Directory Access Protocol (LDAP) from an LDAP server. These policies are then installed in one or more TCP/IP stacks. Policy Agent can be configured to install identical policies to multiple (or all) stacks, or can install different sets of policies to each stack individually. Policy Agent can also monitor its configuration files and the LDAP server periodically for changed policies, and install new or changed policies as changes occur. The basic types of policies are:

- Quality of Service (QoS)
- Intrusion Detection Services (IDS)
  
  See Chapter 28, “Diagnosing intrusion detection problems,” on page 695 for more information about diagnosing IDS policies.
- IPSec
  
  See Chapter 31, “Diagnosing IP security and defensive filter problems,” on page 725 for more information about diagnosing IPSec policies.
- Application Transparent Transport Layer Security (AT-TLS)
  
- Policy-based routing (Routing)
  
  See “Steps for diagnosing problems with IP routing to a destination when using policy-based routing (PBR)” on page 34 for more information about diagnosing routing policies.
Refer to the z/OS Communications Server: IP Configuration Guide for more information about configuring and starting Policy Agent, as well as defining policies.

**QoS policy**

You should become familiar with the following terms to understand QoS policies:

**Quality of Service (QoS)**
The overall service that a user or application receives from a network, in terms of throughput, delay, and such

**Service Differentiation**
The ability of a network to provide different QoS levels to different users or applications based on their needs.

**Service Level Agreement (SLA)**
A contract, in business terms, provided by a network service provider that details the QoS that users or applications are expected to receive.

**Service Policy**
Administrative controls for a network, which are needed to achieve the QoS promised by a given SLA.

**Integrated Services**
A type of service that provides end-to-end QoS to an application, using the methodology of resource reservation along the data path from a receiver to a sender.

**Differentiated Services**
A type of service that provides QoS to broad classes of traffic or users, for example, all FTP traffic to a given subnet.

**Resource ReSerVation Protocol (RSVP)**
A protocol that provides for resource reservation in support of Integrated Services.

**QoS policy scope**

QoS policies can be defined with different scopes. The following scopes are supported:

**DataTraffic**
The policy applies to generic data traffic. This type of policy is in support of Differentiated Services.

**RSVP**
The policy applies to RSVP data traffic. This type of policy is in support of Integrated Services.

The TCP/IP stack maps TCP, UDP, and RAW traffic to QoS policies based on the selection criteria defined in the policy. Search criteria can include, but are not limited to, items such as source and destination IP addresses and ports, protocol, and interfaces. The mapping of DataTraffic scoped policies occurs at connect time for TCP traffic, and for each packet for UDP and RAW traffic. However, for UDP and RAW, the mappings are cached such that subsequent packets sent to the same destination use the cached mapping. RSVP scoped policies are only mapped when the RSVP Agent adds a reservation requested by an RSVP application. The mapping is removed when the reservation is removed. See Chapter 27, “Diagnosing RSVP agent problems,” on page 683 for more information on the operation of RSVP.
You can see the effect of defined QoS policies in the following ways:

- Use the Network SLAPM2 Subagent to display service policy and mapped application information, as well as to manage and display Network SLAPM2 performance monitoring.
- Use the z/OS UNIX `pasearch`, `z/OS UNIX netstat`, and TSO NETSTAT commands as follows:
  - The `pasearch` command shows defined policies.
  - The NETSTAT SLAP or `netstat -j` command shows performance metrics for active QoS policy rules.
  - The NETSTAT ALL or `netstat -A` command has additional information for each active connection that shows the QoS policy rule name if the connection maps to a QoS policy.

Refer to the z/OS Communications Server: IP System Administrator’s Commands for more information on the Netstat command, the `pasearch` command, and the Network SLAPM2 Subagent.

---

**Configuration file import services**

The IBM Configuration Assistant for z/OS Communications Server can request that existing configuration files be imported for further changes and additions. When the Policy Agent is providing this configuration file import service, the IBM Configuration Assistant is acting as an import requestor. These files are called import configuration files and the resulting policies are called import policies.

See the Policy-based networking information in z/OS Communications Server: IP Configuration Guide for more details.

---

**Gathering diagnostic information**

Policy Agent writes logging information to a log file. The level of logged information is controlled by the LogLevel configuration statement and the `-d` startup option. This information (loglevel and debug level) can also be changed after startup using the MODIFY command as shown in the following example:

```
MODIFY procname,LOGLEVEL,LEVEL=127
MODIFY procname,DEBUG,LEVEL=2
```

Error, console, warning, and event LogLevel messages are written by default. To gather more diagnostic information, you can specify a LogLevel value greater than the default or specify debug level 1. This debug level has the side effect of setting the maximum LogLevel value as well.

If you are using both a policy server and one or more policy clients, be sure to gather the log files from all affected Policy Agent applications.

Use the debug levels as follows:

**Debug level 1**

Use debug level 1 for most debugging, except Sysplex Distributor performance monitor. This debug value gives extra debugging messages and uses the maximum LogLevel for logging.

**Debug level 2**

Use debug level 2 to verify Policy Agent processing of LDAP objects, or if a problem is suspected in how LDAP objects are defined.
Debug level 4
Use debug level 4 for summary information concerning Sysplex Distributor performance monitor QoS fraction calculations.

Debug level 8
Use debug level 8 for detailed information concerning Sysplex Distributor performance monitor QoS fraction calculations, and additional Sysplex Distributor debugging.

Debug level 16
Use debug level 16 to assist with memory allocation and leak problems. This debug value causes memory allocation and free requests to be logged inline. This can be used in conjunction with the -m startup option and the MODIFY MEMTRC command to debug memory problems.

Debug level 32
Use debug level 32 for detailed information about all policies as they are installed in the TCP/IP stack.

Debug level 64
Use debug level 64 for detailed locking information within Policy Agent.

Debug level 128
Use debug level 128 for details about remote PAPI connections on the policy server, and about connections to the policy server on the policy client.

Use the trace option -t to turn on LDAP client library debugging. Use the trace levels as follows:

Trace level 0
Use trace level 0 for no LDAP client library debugging. This is the default.

Trace level 1
Use trace level 1 to turn on LDAP client library debugging. Note that the destination of LDAP client debug messages is stderr. This is controlled by the LDAP client library, not Policy Agent. Using trace level 1 turns on the following LDAP DEBUG options:

- LDAP_DEBUG_TRACE
- LDAP_DEBUG_PACKETS
- LDAP_DEBUG_ARGS
- LDAP_DEBUG_CONNS
- LDAP_DEBUG_BER
- LDAP_DEBUG_FILTER
- LDAP_DEBUG_MESSAGE
- LDAP_DEBUG_STATS
- LDAP_DEBUG_THREAD
- LDAP_DEBUG_PARSE
- LDAP_DEBUG_PERFORMANCE
- LDAP_DEBUG_REFERRAL
- LDAP_DEBUG_ERROR

Trace option disabled
If you start Policy Agent with the trace option disabled, the stderr output destination is closed.
Restriction: You cannot turn on the trace option later with the MODIFY command.

Refer to the [z/OS Communications Server: IP Configuration Reference](#) for details on how to use the LogLevel, debug level, and trace level.

Log output can be directed either to a set of log files or to the syslog daemon (syslogd). This can be accomplished with the -l startup option or the PAGENT_LOG_FILE environment variable. If output is directed to log files, the number and size of the files can be controlled using the PAGENT_LOG_FILE_CONTROL environment variable. This environment variable can be used to extend the size of the log information collected if necessary. For example, if a large LDAP configuration is used with debug level 2, the default log file size and number might not be sufficient to capture all of the information needed. In this case, use the environment variable to increase the number or size, or the number and size, of the log files. Refer to the [z/OS Communications Server: IP Configuration Guide](#) for more details on using LogLevel, the -d startup option, and the environment variables, as well as the location of the log file.

The following additional information might be useful in diagnosing Policy Agent problems:

- Output from the `pasearch` command
- Output from the `NETSTAT IDS` or `netstat -k` commands
- Output from the `NETSTAT SLAP` or `netstat -j` commands
- Output from the `NETSTAT ALL` or `netstat -A` commands for active connections mapped to policies
- Output from the `ipsec` command for IPSec policies
- Output from the `NETSTAT TTLS` or `netstat -x` command for AT-TLS policies
- SNMP output from walks of the Network SLAPM2 subagent MIB tables
- TCP/IP CTRACE output, using the POLICY, INTERNET and IOCTL CTRACE options
- RSVP Agent log output if RSVP scoped policies are defined

### Diagnosing Policy Agent problems

Policy Agent problems generally fall into one of the following categories:

- "Initialization problems"
- "Policy definition problems" on page 659
- "Policy client connection problems" on page 664
- "Policy client retrieval problems" on page 667
- "Import requestor connection problems" on page 669
- "Import requestor retrieval problems" on page 673
- "LDAP object retrieval problems" on page 673
- "LDAP object storage problems" on page 675
- "Policy Agent and Sysplex distribution problems" on page 677
- "Memory allocation/leakage problems" on page 678

#### Initialization problems

If Policy Agent does not complete initialization, or fails to install any policies in one or more stacks, run it with the -d 1 startup option, and check the log file for
error conditions. If Policy Agent fails to initialize, message EZZ8434I is issued to the console. Check the log file for the specific error encountered.

Table 52 lists some common Policy Agent initialization problems.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Agent started from a user ID without superuser authority</td>
<td>Policy Agent must be started from a superuser</td>
<td>EZZ8434I message, along with messages in the log file indicating that superuser authority is required and showing an exit code value of 27.</td>
</tr>
<tr>
<td>Policy Agent not authorized to security product</td>
<td>Policy Agent must be authorized to a security product profile.</td>
<td>EZZ8434I message, along with messages in the log file indicating that the user is not authorized to start Policy Agent and showing an exit code value of 18.</td>
</tr>
<tr>
<td>When the SERVAUTH class is active, either:</td>
<td>See “Common AT-TLS startup errors” on page 701 for how to handle this startup error.</td>
<td>EZZ4248E message, and in the Policy Agent log file you see the SYSErr message that indicates that the socket could not open with errno= Resource temporarily unavailable, errno2=74610296.</td>
</tr>
<tr>
<td>• INITSTACK security product profile is not defined</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Policy Agent not permitted to the INITSTACK security product profile.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unable to read configuration file</td>
<td>• The correct configuration file must be specified. Refer to the z/OS Communications Server: IP Configuration Guide for the search order used to locate the main configuration file.</td>
<td>EZZ8434I message, along with messages in the log file indicating that the configuration file could not be opened and showing an exit code value of 1.</td>
</tr>
<tr>
<td>• The file must exist.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• The permission bits must be correctly set for a z/OS UNIX file system file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Because multiple configuration files might be configured, you might need to check these files also.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unable to load one or more dynamic link libraries (DLLs) needed when Policy Agent is configured as a policy client</td>
<td>Policy Agent must have access to several DLLs at run time when configured as a policy client. These are needed to access PAPI functions and to establish an SSL connection to the policy server. Policy Agent accesses the DLLs using the LIBPATH environment variable. Check that the LIBPATH environment variable is specified, and that it contains the directory in which the DLLs reside. This is normally /usr/lib.</td>
<td>EZZ8780I message, along with messages in the log file indicating problems loading one or more of the following DLLs: • papi.dll • GSKSSL • GSKCMS31</td>
</tr>
</tbody>
</table>
Policy definition problems

If you do not see the expected results when defining policies, use the `pasearch` command to display policies (active or inactive) known by Policy Agent. Use this command to check whether policies are active or inactive and whether or not they contain the specifications that were expected.

Guidelines:

- Policy rules with complex conditions (using CNF/DNF logic) are processed by Policy Agent to arrive at a "working" set of conditions. These are the only conditions displayed by default using pasearch (use the -o option to display the original set of conditions as specified).
- The pasearch output displays overall time ranges and time of day ranges in UTC format, as well as the specified time zone, if other than UTC.

You can dynamically refresh Policy Agent so that it can pick up any changes made, including changes to policies on the LDAP server (or configuration file). Use the MODIFY procline, REFRESH command to restart Policy Agent from the beginning of its configuration files, or MODIFY procline, UPDATE to re-read the configuration files.

To check whether QoS policies are being installed and used correctly, use the NETSTAT commands. Use the NETSTAT SLAP or `netstat -j` command to display active QoS policy statistics for QoS policies installed in the stack, as opposed to the policies in Policy Agent. The NETSTAT ALL or `netstat -A` command shows which QoS policy rule (if any) is mapped to active connections.

For further diagnosis of the following policy types, refer to the topics listed below:

- Intrusion Detection Services (IDS) policy definition problems
  See Chapter 28, "Diagnosing intrusion detection problems," on page 695 for more information about diagnosing IDS policy definition problems.
- IPSec policy definition problems
  See Chapter 31, "Diagnosing IP security and defensive filter problems," on page 725 for more information about diagnosing IPSec policy definition problems.
- Application Transparent Transport Layer Security (AT-TLS) policy definition problems
  See Chapter 29, "Diagnosing Application Transparent Transport Layer Security (AT-TLS)," on page 701 for more information about diagnosing AT-TLS policy definition problems.

You might encounter some of the policy definition problems listed in Table 53.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSKCMS31 DLL not found</td>
<td>Policy Agent must have access to the GSKCMS31 DLL at run time. This is needed for IPSec KeyExchange policies. The IPSec policy being validated failed.</td>
<td>Policy Agent logs a system error message and object error message.</td>
</tr>
<tr>
<td></td>
<td>Policy Agent accesses the GSKCMS31 DLL using the LIBPATH environment variable.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Check that the LIBPATH environment variable is specified, and that it contains the directory in which the GSKCMS31 DLL resides. This is normally /usr/lib.</td>
<td></td>
</tr>
<tr>
<td>Problem</td>
<td>Cause/action</td>
<td>Symptom</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>GSKSSL DLL not found</td>
<td>Policy Agent must have access to the GSKSSL DLL at run time. This is needed for AT-TLS policies.</td>
<td>Policy Agent logs a system error message and warning message.</td>
</tr>
<tr>
<td></td>
<td>Policy Agent loads the AT-TLS policies into the TCP/IP stack, but because Policy Agent was unable to verify with System SSL that the configured cipher suites were valid, they are validated when the TLS/SSL environment is initialized for TCP/IP connections. If any values are not valid within the cipher suites, this could result in TCP/IP connections failing. Policy Agent accesses the GSKSSL DLL using the LIBPATH environment variable.</td>
<td>Check that the LIBPATH environment variable is specified, and that it contains the directory in which the GSKSSL DLL resides. This is normally /usr/lib.</td>
</tr>
<tr>
<td>Version 1 QoS policies to version 2 QoS policies conversion</td>
<td>The following circumstances might lead to problems:</td>
<td>Discrepancies between version 1 and version 2 policy definitions.</td>
</tr>
<tr>
<td>Semantic differences exist between version 1 and version 2 policy definitions.</td>
<td>- When converting such policies to version 2, be sure to also swap the source and destination attributes when the version 1 Direction is Inbound. The specified interface is also related to Direction. In version 1 only a single interface is specified, while both inbound and outbound interfaces are specified in version 2. When migrating a version 1 policy, be sure to specify an InboundInterface for Direction Inbound, and an OutboundInterface for Direction Outbound.</td>
<td></td>
</tr>
<tr>
<td>Restriction: Currently only version 2 semantics are supported. When the policies are processed by Policy Agent, version 1 policy semantics are converted to version 2 semantics.</td>
<td>- When converting version 1 rules with Direction Both specified, create two version 2 rules, one for each direction. Also, specify InboundInterface for the inbound rule and OutboundInterface for the outbound rule, if the version 1 rule specified both Interface and Direction Both.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- When converting policies with different PolicyScope values, be sure to logically merge the scopes in the version 2 policy action. Any such merge should always result in a PolicyScope value of Both.</td>
<td></td>
</tr>
<tr>
<td>Policy groups or rules are discarded when defined on an LDAP server.</td>
<td>Policy groups and policy rules defined on an LDAP server can refer to other LDAP objects (such as policy actions or time periods). When any referenced object cannot be found on the LDAP server, the referencing object is discarded.</td>
<td>Discarded policy groups or rules</td>
</tr>
<tr>
<td></td>
<td>Specify the correct reference Distinguished Names on LDAP objects that reference other objects.</td>
<td></td>
</tr>
<tr>
<td>Problem</td>
<td>Cause/action</td>
<td>Symptom</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>Policies with complex conditions (using CNF or DNF) are not mapping correctly.</td>
<td>Because some conditions are logically ANDed, a result that is not valid can occur. For example, two or more distinct interfaces cannot be ANDed and still be true. Or two non-overlapping port ranges also cannot be ANDed. Policy Agent tries to detect these types of errors and discard the policy rules with an error message, but there are cases that cannot be detected (for example, logical ANDs between CNF/DNF levels, or when negated conditions are used). In these cases, a policy rule can be installed that can never be true. Similar problems could occur when ORing conditions. For example, a very broad condition might map much more traffic than was intended, simply because it is one of a set of conditions that is ORed together. Use the <code>pasearch</code> command to display policy rules with complex conditions. By default, the “working” set of conditions is displayed (after Policy Agent has attempted to collapse and summarize the complex conditions). This working set includes the summary of each condition level, as well as the overall “global” summary condition. Use the <code>pasearch -o</code> option to also display the original set of specified conditions. This helps to show how the working set was derived.</td>
<td>Difficulty configuring complex policy conditions using CNF or DNF.</td>
</tr>
<tr>
<td>Wrong policy being mapped to traffic</td>
<td>At times, two or more policy rules are logically mapped to the same set of traffic packets. When this happens, the rule with the highest weight is selected. The weight depends on two factors. When the policy rule priority is not specified, the weight depends on the number of attributes specified in the policy conditions. When policy rule priority is specified, the weight is the specified priority plus 100, which is always higher than the weight derived from counting the number of attributes. If more than one rule is found with the same weight, the first such rule is selected to be mapped. Be sure to specify priority in policy rules to better control situations where multiple rules map to the same set of traffic.</td>
<td>Policy rule priority settings are inadequate to control situations where multiple rules map to the same set of traffic.</td>
</tr>
</tbody>
</table>
Table 53. Policy definition problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
</table>
| Policies are not installed in the TCP/IP stack. | Perform the following actions, based on what caused the problem:  
  - The stack in which policies should be installed must be configured using a TcpImage statement in the Policy Agent configuration file.  
  - The time periods configured in the policies must be correct. Verify the specifications of the day of week and time of day are correct. Verify that the specified time zone is correct. For time zones other than local time, the specified time periods might not be currently active.  
  - If the stack was started or restarted after Policy Agent was started, check that the temporary file (/tmp/tcpname.Pagent.tmp) used by the stack to inform Policy Agent of restarts has not been deleted. | Unexpected or missing set of policies. |

Perform the following steps to diagnose QoS problems:  
- Issue `pasearch -q` to see all QoS policies that are active in Policy Agent. Refer to [z/OS Communications Server](https://www.ibm.com) *IP System Administrator's Commands* for more information about the `pasearch -q` command. If you are running multiple stacks, ensure that pasearch is reporting on the stack you are interested in.  
- Issue `NETSTAT SLAP` or `netstat -j` command to see how the stack mapped your QoS statement. Refer to [z/OS Communications Server: IP System Administrator's Commands](https://www.ibm.com) for more information about the `netstat` command. If you are running multiple stacks, ensure that your resolver configuration correctly identifies the stack you are interested in. Ensure that your QoS policies are correctly defined. For more information about policy-based networking, refer to [z/OS Communications Server: IP Configuration Guide](https://www.ibm.com).  
- See Chapter 28, “Diagnosing intrusion detection problems,” on page 695 for more information about diagnosing IDS policy definition problems.  
<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoS policies not mapping to the expected traffic</td>
<td>Incorrectly specified selection criteria on the PolicyRule statement for the policy. If you think data traffic should be mapped to certain policies, but is not, check to make sure you have specified the selection criteria correctly on the PolicyRule statement for the policy. For example, TCP policies are mapped on a per connection basis, whereas for UDP and RAW, the policy is mapped on a per packet basis. As an example of TCP traffic, consider an ftp GET request from a remote client. The connection request from the client is mapped as inbound data, while the data flow is mapped as outbound data. You can use either source or destination fields in the policy rule to map both traffic flows, but the definitions must be consistent with this way of mapping. Check that the policy is not unnecessarily restrictive in its specification of IP addresses and ports. For RSVP scoped policies, remember that the policy is only mapped to data traffic while an RSVP reservation is in effect.</td>
<td>A blank policy rule name is displayed for an active connection using the NETSTAT ALL or netstat -A command.</td>
</tr>
<tr>
<td>Timing windows when switching policies based on time</td>
<td>If policy rules are defined such that different sets of policies are activated at different times (for example at each shift), be aware of nonoverlapping vs. overlapping time specifications. For example, if Rule1 is active from 00:00 to 07:29, and Rule2 is active from 07:30 to 04:00, there is a one minute interval gap between these 2 rules. Because the minimum time resolution used by Policy Agent is one minute, there is a period of one minute when neither policy is active.</td>
<td>Different sets of policies are activated at different times (for example at each shift).</td>
</tr>
<tr>
<td>Policies defined in an MVS data set are not being installed.</td>
<td>When an MVS data set is used to define policies, ensure that sequence numbers are not part of the file, because these cause parsing errors. In ISPF, use the NUMBER OFF and UNNUM or NUMBER OFF or UNNUM commands to remove the sequence numbers.</td>
<td>Parsing errors occur.</td>
</tr>
</tbody>
</table>

Note 1. Be aware of the following processing behavior:

- In version 1, source always meant local, while destination always meant remote. In version 2, source and destination mean exactly what they imply. When version 1 policies specify Direction Inbound, the semantics for source and destination are opposite between the two versions. As a result, although the specified source and destination attributes are displayed as they are specified by the `pasearch` command, the attributes are swapped when the policies are installed in the stack.

- Similarly, when Direction Both is specified in a version 1 policy, the following policies are installed in the stack:
  - Outbound direction with source and destination attributes intact
  - Inbound direction with the attributes swapped

- PolicyScope values exist in both the policy rule and action in version 1, but only in the policy action in version 2. For any policies that specified different PolicyScope values for the rule and the associated action in version 1, the scope values are merged in the policy action. For example, if the rule specified PolicyScope Both, and the associated action specified PolicyScope DataTraffic, the resulting scope value in the policy action is Both.
Policy client connection problems

When acting as a policy client, Policy Agent needs to connect to a policy server. The policy client can be configured with just a primary, or both a primary and a backup, policy server. See z/OS Communications Server: IP Configuration Guide, Policy Agent and policy applications for more information about how the policy client connects to a policy server.

If the policy client does not connect successfully, run Policy Agent on the policy client and policy server with the -d 128 startup option, and check the log files for error conditions. Connection problems are indicated by message EZZ8780I or message EZZ8782I. Check the log files for the specific error encountered.

Table 54 describes common policy client connection problems.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
</table>
| Incorrect configuration on the policy client or policy server | - The policy server must be configured with the ClientConnection statement specifying the port to which policy clients connect.  
- If you use secure connections from any policy clients, the policy server must be configured with AT-TLS policies that allow those policy clients to establish SSL connections to the policy server.  
- The policy client must be configured with the ServerConnection statement specifying the host name or IP address, and port of the primary and optional backup policy server, as well as connection retry information.  
- If you want to use a secure connection to the policy server, you must configure the policy client with SSL information on the ServerConnection statement. | Message EZZ8780I or EZZ8782I, along with messages in the log files indicating the particular connection problem details. |

Refer to the policy-based networking section in z/OS Communications Server: IP Configuration Guide for details about setting up the correct configuration.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
</table>
| Incorrect SSL configuration on the policy client or policy server | If you use secure connections from the policy client:  
  - The policy server must be configured with AT-TLS policies that allow the policy clients to establish SSL connections to the policy server.  
  - The policy server must be configured with a certificate that allows the policy clients to authenticate the server.  
  - If a self-signed server certificate is used, the policy client must import the server's certificate into the client's key ring.  
  - The ServerConnection statement on the policy client must be configured with the correct SSL parameters.  
  Refer to the policy-based networking section in [z/OS Communications Server: IP Configuration Guide](https://www.ibm.com/support/knowledgecenter/en/SSLVWMG_7.2.1/com.ibm.zos.v2r11.mfipcon/task/c242126.html) for details about setting up the correct configuration. | Message EZZ8780I or EZZ8782I, along with messages in the log files indicating the particular connection problem details. |
| Mismatched security configuration between the policy client and policy server | The configuration on the policy client must match the configuration on the policy server with respect to SSL and AT-TLS:  
  - If the policy client is configured with SSL parameters on the ServerConnection statement, the policy server must have an AT-TLS policy that protects connections from that policy client.  
  - If the policy client is not configured with SSL parameters on the ServerConnection statement, the policy server must not have an AT-TLS policy that protects connections from that policy client.  
  Use the pasearch command to display the AT-TLS policies on the policy server, and verify that the selection criteria in the policy rules select only those policy clients that use SSL. Look for policy rules that specify the port specified on the ClientConnection statement as the local port, and in particular, verify that the remote IP address and remote port parameters on those policy rules are correct for your configuration.  
  See the policy-based networking section in [z/OS Communications Server: IP Configuration Guide](https://www.ibm.com/support/knowledgecenter/en/SSLVWMG_7.2.1/com.ibm.zos.v2r11.mfipcon/task/c242126.html) for details about setting up the correct configuration. | Message EZZ8780I or EZZ8782I, along with messages in the log files indicating the particular connection problem details. |
### Table 54. Common policy client connection problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
</table>
| Incorrect certificate name specified on the ServerSSLName parameter on  | • If the AT-TLS policy on the policy server specifies HandshakeRole Server, the ServerSSLName parameter on the ServerConnection statement on the policy client must specify the name of the server’s certificate.  
  • If the AT-TLS policy on the policy server specifies HandshakeRole ServerWithClientAuth, the ServerSSLName parameter on the ServerConnection statement on the policy client must specify the name of the client’s certificate. | Message EZZ8780I or EZZ8782I, along with messages in the log files indicating the particular connection problem.                                                      |
| the ServerConnection statement                                           |                                                                                                                                                                                                           |                                                                                                                                                                                                                                |
| Policy client not authorized to access policy server                     | • The policy server must be configured with one or more user IDs and credentials for the set of policy clients that are authorized to connect.  
  **Rule:** If you use a password for credentials, the password must match the password configured using the AuthBy password parameter on the PolicyServer statement on the policy client.  
  • The policy client must be configured with a PolicyServer statement for each stack that will retrieve policies from the policy server, indicating the user ID and credentials that will be used to access the policy server. | Message EZZ8780I or EZZ8782I, along with messages in the log files indicating the particular authorization problem details.                                                                                                  |
| Incorrect passticket configuration on the policy client or policy server | If the policy client is configured to use a passticket on the PolicyServer statement, the proper PTKTDATA class profiles must be defined on both the policy server and policy client. | Message EZZ8780I or EZZ8782I, along with messages in the log files indicating the particular connection problem details.                                                                                                            |

### Table 54. Common policy client connection problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>The policy server is not listening on the port defined on the ClientConnection statement.</td>
<td>If the ClientConnection statement is configured on the policy server, the port specified on this statement may need to be reserved using the PORT statement in the TCP/IP profile.</td>
<td>Message EZZ8788I, along with messages in the log files, indicating the particular connection problem details.</td>
</tr>
<tr>
<td>Duplicate policy client name reported</td>
<td>If you use the configuration file import service on the policy server, you might encounter a duplicate policy client name for a policy client. The reason for this is that temporary names are generated in order to process a configuration file import. If a policy client tries to connect to the policy server while a configuration file import is in progress, it’s possible that the policy client name matches the generated temporary name. If this happens, issue a MODIFY UPDATE command on the policy client to cause it to reconnect to the policy server, once the configuration file import service has completed.</td>
<td>Message EZZ8781I followed by message EZZ8782I, along with messages in the log files indicating a duplicate policy client name was detected.</td>
</tr>
</tbody>
</table>

### Policy client retrieval problems

When acting as a policy client, Policy Agent retrieves policies for one or more policy types, on behalf of one or more stacks, from a policy server. The choice of local or remote policy retrieval can be made separately for each policy type, and for each configured stack. See [z/OS Communications Server: IP Configuration Guide](z/OS%20Communications%20Server%3A%20IP%20Configuration%20Guide) Policy Agent and policy applications for more information about policy client retrieval of remote policies.

If the policy client does not successfully retrieve policies, run Policy Agent on the policy client and policy server with the -d 128 startup option, and check the log files for error conditions. Retrieval problems are indicated by message EZZ8438I. Check the log files for the specific error encountered.

Table 55 on page 668 describes common policy client retrieval problems.
### Table 55. Common policy client retrieval problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
</table>
| Incorrect configuration on the policy client or policy server | - The policy server should be configured with one or more DynamicConfigPolicyLoad statements that match the client name. The DynamicConfigPolicyLoad statement determines the configuration files that get loaded after a policy client successfully connects. If a matching DynamicConfigPolicyLoad statement is not found, the policy server will attempt to load policies from a default file. Ensure that the correct set of DynamicConfigPolicyLoad statements is specified, and that the correct configuration files are specified on these statements.  
  - The policy client must be configured with a PolicyServer statement for each stack that will retrieve policies from the policy server. The ClientName specified on this statement is used to match a DynamicConfigPolicyLoad statement on the policy server. If the ClientName parameter is not specified, the default client name used is `remotesysname.tcpimage` where:  
    - `remotesysname` value is the policy client system name and `tcpimage` value is the policy client image name.  
    - Refer to [z/OS Communications Server: IP Configuration Guide](https://www.ibm.com/support/docview.wss?uid=swg27042967), Policy Agent and policy applications and [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com/support/docview.wss?uid=swg27042786), general configuration file statements section for more information. | Incorrect or no policies retrieved from the policy server. |
Table 55. Common policy client retrieval problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect regular expressions coded on the DynamicConfigPolicyLoad statement</td>
<td>The DynamicConfigPolicyLoad statements can be configured with regular expressions to match against policy client names. Regular expressions are very powerful, but also can be complex, and might not produce results that are intuitive. For example, the expression [a-z] matches any lower case alphabetic character, which means that any string containing at least one such character will match. As another example, the expression [^abc] means any character except a, b, or c matches. So the only strings that won't match are those containing ONLY the characters a, b, or c. Refer to z/OS Communications Server: IP Configuration Guide, Policy Agent and policy applications and z/OS Communications Server: IP Configuration Reference DynamicConfigPolicyLoad statement for more information.</td>
<td>Incorrect or no policies retrieved from the policy server.</td>
</tr>
<tr>
<td>Policy client not authorized to access policies on the policy server</td>
<td>The policy server must be configured with SERVAUTH profiles that allow the policy clients to access policies. The format of the SERVAUTH profiles is: EZB.PAGENT.sysname.image.ptype where: • sysname is the policy server system name • image is the policy client name <strong>Rule:</strong> The image portion of the profile name on the policy server must match or include the name of the policy clients. Each policy client name is configured or defaulted using the ClientName parameter on the PolicyServer statement. • ptype is the policy type (QOS, IDS, IPSEC, ROUTING, or TTLS) See z/OS Communications Server: IP Configuration Guide Policy Agent and policy applications for more information.</td>
<td>Incorrect or no policies retrieved from the policy server.</td>
</tr>
</tbody>
</table>

**Import requestor connection problems**

The import requestor connects to a Policy Agent. The IBM Configuration Assistant for z/OS Communications Server can be an import requestor. See z/OS Communications Server: IP Configuration Guide for information on the configuration file import service.

If the import requestor does not connect successfully, run Policy Agent with the -d 128 startup option, and check the log files for error conditions. Connection problems are indicated by message EZD1578I in the log. Check the log files for the specific error encountered.
Table 56 describes common import requestor connection problems.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect configuration on</td>
<td>You must configure the Policy Agent with the ServicesConnection statement specifying the port and TCP/IP stack name to which the import requestor will connect.</td>
<td>Message EZD1578I, along with messages in the log files, indicating the particular connection problem details.</td>
</tr>
<tr>
<td>Policy Agent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incorrect SSL configuration on</td>
<td>• If the import requestor is using SSL, you must configure the Policy Agent with a SAF keyring and the Security parameter set to Secure on the ServicesConnection statement.</td>
<td>Message EZD1578I, along with messages in the log files, indicating the particular connection problem details.</td>
</tr>
<tr>
<td>the Policy Agent or import</td>
<td>- The Policy Agent generates and installs an AT-TLS policy that allows the import requestors to establish SSL connections to the Policy Agent.</td>
<td></td>
</tr>
<tr>
<td>requestor</td>
<td>- You must configure a certificate in a SAF keyring that allows the import requestors to authenticate the server.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• If the import requestor is not using SSL, you must configure the Policy Agent with the Security parameter set to Basic on the ServicesConnection statement. You must not configure an AT-TLS policy that includes the port configured on the ServicesConnection statement.</td>
<td></td>
</tr>
</tbody>
</table>

See z/OS Communications Server: IP Configuration Guide, configuration file import service, for details about setting up the correct configuration.

See z/OS Communications Server: IP Configuration Guide, configuration file import service, for details about setting up the correct configuration.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Policy Agent did not issue message EZD1576I indicating it is ready for services connection requests.</td>
<td>If you are using secured connections for import requestors, and have AT-TLS policies configured on a policy server, the Policy Agent waits for the remote AT-TLS policies to be retrieved and installed before installing the generated AT-TLS policy for the port specified on the ServicesConnection statement. If the policy server is down or can not be contacted immediately, the generated AT-TLS policy can not be installed and the Policy Agent does not listen for import requestor connections.</td>
<td>Message EZD1576I is not issued, and import requestors cannot connect to the Policy Agent. Message EZD1578I, along with messages in the log files, indicating the particular connection problem details.</td>
</tr>
<tr>
<td></td>
<td>• Verify the policy server is available and the Policy Agent is active.</td>
<td>Message EZZ8438I, indicating errors in the local or remote AT-TLS policies for a TCP/IP image, where the secured connections for import requestor is requested. Message EZD1578I, along with messages in the log files indicating the particular connection problem details. Message EZD1576I is not issued, and import requestors cannot connect to the Policy Agent.</td>
</tr>
<tr>
<td></td>
<td>• You might consider using a backup policy server to handle policy client connections when the primary is not available.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The MODIFY SRVLSTN command could be used to force the generated AT-TLS policy to be installed before the remote AT-TLS policies are installed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Run the policy client and policy server with debug level 128 and check the Policy Agent log files to determine the cause of any connectivity problems.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See z/OS Communications Server: IP System Administrator’s Commands for information on the MODIFY command and z/OS Communications Server: IP Configuration Guide for AT-TLS data protection information.</td>
<td></td>
</tr>
</tbody>
</table>

The Policy Agent did not issue message EZD1576I indicating it is ready for services connection requests.

If you are using secured connections for import requestors, and have local or remote AT-TLS policies configured that contain errors, Policy Agent waits for the local or remote AT-TLS policies to be installed.

• Correct the configured AT-TLS policies and refresh policies.

• The MODIFY SRVLSTN command could be used to force the generated AT-TLS policy to be installed before the local or remote AT-TLS policies are installed.

See z/OS Communications Server: IP System Administrator’s Commands for information on the MODIFY command and z/OS Communications Server: IP Configuration Guide for AT-TLS data protection information.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
</table>
| **Import requestor does not successfully connect to the Policy Agent.** | If you are using secured connections for import requestors and the SAF keyring is correct but the connection from the import requestor fails, (indicating key ring problems) check the following:  
- If the key ring certificate has expired, then update the expiration date and issue the MODIFY SRVLSTN command for Policy Agent to reinstall the generated AT-TLS policy and to restart the listen for services requestor connections  
- If the contents of the key ring has changed, but the key ring name is unchanged, issue the MODIFY SRVLSTN for Policy Agent to reinstall the generated AT-TLS policy and to restart the listen for services requestor connections | Message EZD1576I is issued, but import requestors cannot connect to the Policy Agent. |
| **The Policy Agent is not listening on the port defined on the ServicesConnection statement.** | If the ServicesConnection statement is configured, the port specified on this statement may need to be reserved using the PORT statement in the TCP/IP profile. | Message EZD1578I, along with messages in the log files, indicating the particular connection problem details. |
| **Import requestor not authorized to access Policy Agent system** | The Policy Agent system must be configured with one or more user IDs and credentials for the set of import requestors that are authorized to connect.  
**Rule:** If you use a password for credentials, the password must match the password configured on the import requestor. If you use the IBM Configuration Assistant for z/OS Communications Server as the import requestor, the user ID and password are configured on the Flat File Import panel. | Message EZD1578I, along with messages in the log files, indicating the particular connection problem details. |
Import requestor retrieval problems

The import requestor retrieves import policies from the Policy Agent.

Table 57 describes common import requestor retrieval problems.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import requestor not authorized to access import policies on the Policy Agent.</td>
<td>The Policy Agent must be configured with SERVAUTH profiles that allow the import requestor to access import policies. The format of the SERVAUTH profiles is: EZB.PAGENT.sysname.image.ptype</td>
<td>Incorrect or no import policies retrieved from the Policy Agent.</td>
</tr>
</tbody>
</table>

where:

- sysname is the local system name
- image is the import request name

**Rule:** The image portion of the profile name on the policy server must match or include the import request name. If you use the IBM Configuration Assistant for z/OS Communications Server as the import requestor, the import request name is configured on the Flat File Import panel.

- ptype is the policy type (IDS, IPSEC, ROUTING, or TTLS)

See [z/OS Communications Server: IP Configuration Guide](#), configuration file import service, for details about setting up the correct configuration.

LDAP object retrieval problems

**Before you begin:** If you are having problems receiving policies from an LDAP server, run Policy Agent with the -d 1 or 2 startup options.

In Table 58 on page 674 select actions as indicated according the problem you are experiencing.
Table 58. LDAP object retrieval problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unable to connect to the LDAP server</td>
<td>Check the attributes specified on the ReadFromDirectory statement in the configuration file that relate to the LDAP server connection. These include the primary and backup server addresses and ports, the user ID and password, and SSL parameters.</td>
<td>Message EZZ8440I is issued to the console. If Policy Agent fails to connect to the LDAP server, check the log file for the specific error encountered. The Policy Agent keeps trying to connect to the server, using a sliding time window (one minute, then at five minute intervals, with the maximum time between connect attempts being 30 minutes). <strong>Tip:</strong> If a backup LDAP server is configured, the EZZ8440I message is only issued if neither the primary or backup server can be connected.</td>
</tr>
<tr>
<td>No objects, or incorrect objects, retrieved from the LDAP server</td>
<td>Check that the schema version specified on the ReadFromDirectory statement in the configuration file matches the version defined on the LDAP server. The different versions are distinguished by the set of supported object classes. Refer to the <a href="#">z/OS Communications Server: IP Configuration Guide</a> for supported schema object classes.</td>
<td>Missing or incorrect policies are displayed by the <code>pasearch</code> command, or the <code>NETSTAT SLAP</code> or <code>netstat -j</code> commands.</td>
</tr>
<tr>
<td>Wrong set of objects retrieved from the LDAP server</td>
<td>Check that the search and selection criteria specified on the ReadFromDirectory statement in the configuration file are correct. For version 1 policies, verify that the correct Base and SelectedTag attributes are used. For version 2 and later policies, check the <code>SearchPolicyBaseDN</code>, <code>SearchPolicyGroupKeyword</code>, <code>SearchPolicyKeyword</code>, and <code>SearchPolicyRuleKeyword</code> attributes.</td>
<td>Missing or incorrect policies are displayed by the <code>pasearch</code> command, or the <code>NETSTAT SLAP</code> or <code>netstat -j</code> commands.</td>
</tr>
<tr>
<td>LDAP DLL not found</td>
<td>Check that the LIBPATH environment variable is specified, and that it contains the directory in which the LDAP DLL (GLDCLDAP) resides. This is normally <code>/usr/lib</code>. Policy Agent accesses the LDAP DLL using the LIBPATH environment variable.</td>
<td>Policy Agent terminates unexpectedly with a CEEDUMP. The reason for termination in the CEEDUMP indicates that the LDAP DLL (GLDCLDAP) was not found.</td>
</tr>
</tbody>
</table>

674 z/OS V1R11.0 Comm Svr: IP Diagnosis Guide
Table 58. LDAP object retrieval problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1 policies not shared among multiple TCP/IP stacks</td>
<td>Policy Agent uses two attributes when it searches an LDAP server for version 1 policies that apply to a given TCP/IP image. One attribute is the TCP/IP image name and the other is a selector tag. The selector tag attribute can be defined such that LDAP scopes the search. The TCP/IP image name attribute is set by default to scope the search for a particular image. Each of the two attributes (TCPImageName and SelectorTag) is a multivalue field, meaning you can specify TCPImpageName/SelectorTag multiple times in one object defined to LDAP. Both multiple MVS images and multiple TCP/IP stacks can exist. If a policy object is to be used in multiple MVS LPARs, that object can have multiple SelectorTag attributes defined, one for each LPAR. If a policy object is to be used in multiple TCP/IP images, that object can have multiple TCPImageName attributes defined, one for each image.</td>
<td>Version 1 policies not shared among multiple TCP/IP stacks</td>
</tr>
</tbody>
</table>

LDAP object storage problems

Policies can be defined on an LDAP server using the appropriate definitions, known as schemas. The policies are defined as object classes with certain attributes, which are a superset of the attributes that can be defined in a local file using the PolicyAction and PolicyRule statements. Policy Agent acts as an LDAP client to communicate with and retrieve policies from an LDAP server. Policy Agent uses an LDAP DLL to perform its LDAP client functions.

**Before you begin:** If you are having problems initializing the LDAP server with the Policy Agent schema definitions or adding policy objects to the server, perform the following steps to diagnose LDAP object storage problems.

In Table 59 on page 676, select actions as indicated according to the problem you are experiencing.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
</table>
| Unable to add the Policy Agent schema definitions to an LDAPv3 server | The Policy Agent LDAPv3 schema definition files are shipped as the following sample files:  
  - pagent_schema.ldif  
  - pagent_v3schema.ldif  
  - pagent_schema_updates.ldif  
  - pagent_idsschema.ldif  
  - pagent_qosschema.ldif  
  - pagent_r5idsschema.ldif  
  - pagent_schema_r5updates.ldif  
  - pagent_r6qosschema.ldif  
  - pagent_schema_r6updates.ldif  
  - pagent_r8qosschema.ldif  
  - pagent_schema_r8updates.ldif  
  Some or all of these files need to be installed on the LDAP server in the proper order as an object in the server’s database, rather than as configuration information. This process is known as schema publication. Refer to RFCs 1804 and 2251. The files need to be specified on `ldapmodify` commands to modify the `cn:schema` entry in the server’s database, in the order as specified in `z/OS Communications Server: IP Configuration Guide`. Verify that the `<suffix>` value on the first noncomment line of these files has been changed to the suffix value defined for your LDAP server, as explained in the prologues in these files. | Symptoms can include error messages issued by the server. Because server implementations are different, check the documentation for your server for the types and locations of error or log messages. |
Table 59. LDAP object storage problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause/action</th>
<th>Symptom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unable to add policy objects to an LDAP server</td>
<td>Check the following: 1. Are the Policy Agent schema definitions installed on the LDAP server? 2. Are the correct object classes identified for any attributes you have defined in the object? For example, the ibm-policySubtreesAuxContainedSet attribute is defined for the ibm-policySubtreesPtrAuxClass object class. 3. Does the server recognize all of your objects?</td>
<td>Symptoms can include error messages issued by the server. Since server implementations are different, check the documentation for your server for the types and locations of error or log messages. A typical error message might indicate object class violation. There are several possible reasons for an LDAP server rejecting a policy object. The following symptoms correspond to the numbered actions in the cause and action column. 1. If the server does not know about policy attributes or object classes, then it fails any objects that contain them. 2. If you define a policy object with this attribute attached, but do not include the object class value, the server flags the object as an object class violation. 3. The symptoms for this are missing objects when you search the server or errors when adding the objects. Some servers can impose strict syntax rules on ldif files that contain objects.  • Lines that separate objects might need just a single newline character. If the separator lines contain other characters, the following object is processed as a continuation of the previous object. If the object file was transferred using FTP from a host, character translation might result in characters other than newlines separating objects. These additional characters must be removed.  • There must be no blanks at the ends of lines.</td>
</tr>
</tbody>
</table>

**Policy Agent and Sysplex distribution problems**

The Policy Agent sysplex distributor (SD) performance monitor function can be used to calculate outbound network performance information, such as TCP packet loss and timeout ratios, for applications being distributed to on SD target nodes. The calculated performance information is in the form of QoS weight fractions calculated for each DVIPA/Port service level. The QoS weight fractions are used to adjust the WLM weight: the higher the Qos fraction calculation, the lower the adjusted WLM weight. For more information on QoS fractions, refer to the section
Steps for diagnosing Policy Agent/Sysplex distribution problems

Before you begin: If you suspect problems with the calculated QoS weight fractions, run Policy Agent with debug level 4 or 8.

Perform the following steps to diagnose Policy Agent/Sysplex distribution problems. Select the steps as indicated by which problem you are experiencing.

- For debug level 4, Policy Agent displays a summary calculation for each DVIPA/Port XCF address and service level. The summary information includes the retransmit fraction, connection limit fraction, throughput fraction, and the final QoS fraction that resulted. For example:
  
  Calculating for DVIPA: 193.1.1.36, Port: 8000, XCF@: 193.1.1.36, SLName: 'Gold_Service'
  Fractions: rxmit: 0, connLimit: 100, thruput: 0 QoS used: 100

- For debug level 8, Policy Agent displays the intermediate values used to generate the above fractions. For example:
  
  Calculating for DVIPA: 193.1.1.36, Port: 8000, XCF@: 193.1.1.36, SLName: 'Gold_Service'
  Retransmit Fraction: 0 (Retransmit Bytes: 544, Timeouts: 1, Octets Sent: 81362424, Segments Sent: 143194)
  Connection limit Fraction: 100 (Max Connections: 3, Active Connections: 3)
  Throughput Fraction: 0 (Out Bytes: 81362424, Throughput: 10848, Conn Throughput: 3616 Profile Rate: 0, Min Rate 2000)
  QoS Fraction used : 100

Guideline: If the throughput fraction gets set to 100% for any service level, message EZZ8447I is issued. To see which service levels caused this message to be issued, run Policy Agent with debug level 8 and check the log file.

For more information see Chapter 11, “Diagnosing dynamic VIPA and sysplex problems,” on page 373

Memory allocation/leakage problems

Policy Agent allocates memory for many resources, such as:

- Policy rules and actions
- Sysplex Distributor lists and weight fraction arrays
- Policy performance data arrays
- LDAP search results

If it appears that Policy Agent is using too much memory, or memory leakage is suspected, use the following tools, possibly in conjunction with other tools outside the scope of Policy Agent, such as dump formatters and Language Environment memory tracing.

Use the -m startup option to keep track of all Policy Agent memory allocation and free requests. All memory allocations are recorded in a memory trace buffer, and all memory free requests find the corresponding entry and remove it. If this option is specified, Policy Agent automatically reports any memory leakage at termination time, because any entries left in the buffer after all memory free requests have been processed are by definition memory leaks. Note that if the memory trace buffer fills up, the memory trace function is dynamically turned off and no more memory tracing is performed. If this occurs, specify a larger value for the -m startup option when Policy Agent is restarted.
Use the MODIFY MEMTRC command to log a snapshot of Policy Agent memory allocations. This command dumps the contents of the memory trace buffer to the log file. As a result, it only has an effect when the -m startup option was specified.

Use debug level 16 to record memory allocation and free requests inline in the log file. This debug level is independent of the -m startup option. Note that using this debug level can result in significantly more information being recorded, so specify larger and/or more log files using the PAGENT_LOG_FILE_CONTROL environment variable.
Chapter 27. Diagnosing RSVP agent problems

The z/OS UNIX RSVP Agent provides end-to-end resource reservation services on behalf of applications. This topic provides information and guidance to diagnose z/OS UNIX RSVP Agent problems and contains the following sections:

- “Overview” on page 683
- “Policies and RSVP processing” on page 683
- “Gathering diagnostic information” on page 684
- “Diagnosing RSVP agent problems” on page 684

Overview

The RSVP Agent provides an RSVP Application Programming Interface (RAPI) for QoS-aware applications to use. Applications use RAPI to register their intent to use RSVP services, to describe their data traffic, and to explicitly request that network resources be reserved on their behalf. The RSVP Agent communicates with its peers (other RSVP Agents running on z/OS or other platforms) in the network, with QoS-aware sender and receiver applications, and with the TCP/IP stack to effect resource reservations. Refer to RFC 2205 for more information on RSVP, and to the z/OS Communications Server: IP Programmer’s Guide and Reference for more information on RAPI.

The following terms must be defined to understand RSVP processing:

**Quality of Service (QoS)**
The overall service that a user or application receives from a network, in terms of throughput, delay, and such.

**QoS-Aware Application**
An application that explicitly requests QoS services from the RSVP agent.

**Service Differentiation**
The ability of a network to provide different levels of QoS to different users or applications based on their needs.

**Service Level Agreement (SLA)**
A contract in business terms provided by a network service provider that details the QoS that users or applications are expected to receive.

**Service Policy**
Administrative controls for a network, in order to achieve the QoS promised by a given SLA.

**Integrated Services**
A type of service that provides end-to-end QoS to an application, using the methodology of resource reservation along the data path from a receiver to a sender.

**Differentiated Services**
A type of service that provides QoS to broad classes of traffic or users, for example all FTP traffic to a given subnet.

**Resource ReSerVation Protocol (RSVP)**
A protocol that provides for resource reservation in support of Integrated Services.
Reservation types, styles, and objects

There are two types of Integrated Services reservations used by the RSVP Agent:

**Controlled Load**
This reservation type is designed to make the network behave as though it were not loaded, even if one or more of the network elements are experiencing a heavy traffic load. Refer to RFC 2211 for more information on this service.

**Guaranteed**
This reservation type is designed to allow the network to compute the maximum delay data traffic receives from the network, based on the traffic specification and other known data. Refer to RFC 2212 for more information on this service.

In addition, there are three styles of reservation, depending on how the receiver desires to apply the reservation to its senders:

**WF (Wildcard Filter)**
This style applies a single reservation request to all senders.

**FF (Fixed Filter)**
This style pairs a given reservation request to a given sender. In this way, the receiver can apply a different reservation to each of its senders.

**SE (Shared Explicit)**
This style applies a single reservation to a list of senders. This differs from the WF style in that the list of senders is finite. Additional senders that appear in the future do not automatically inherit an SE style reservation.

Several objects are used in RSVP and RAPI to describe data traffic and reservations. These objects are as follows:

**Tspec (traffic specification)**
The Tspec is used to describe the sending application data traffic characteristics. It consists of an object known as a token bucket and other related values. A token bucket is a continually sustainable data rate, and the extent to which the rate can exceed the sustainable level for short periods of time. More detail concerning token buckets and other Integrated Services parameters and processing can be found in RFCs 2210, 2211, 2212, and 2215.

The Tspec contains these values:
- **r**: Token bucket rate, in bytes per second
- **b**: Token bucket depth, in bytes
- **p**: Peak rate, in bytes per second
- **m**: Minimum policed unit (minimum packet size to be considered), in bytes
- **M**: Maximum packet size (MTU), in bytes

**Rspec (guaranteed receiver specification)**
An Rspec consists of two values that further describe a reservation request when Guaranteed service is being used:
- **R**: Requested rate, in bytes per second
- **S**: Slack term, in microseconds
Flowspec (reservation specification)

The flowspec is the object used by a receiver application to indicate an actual reservation to be made. The actual makeup of the flowspec depends on the type of reservation. For Controlled Load, the flowspec takes the same form as the sender Tspec (although the form is the same, the receiver might specify different values than the sender). For Guaranteed, the flowspec takes the form of a Tspec followed by an Rspec.

Policies and RSVP processing

Policies can be defined with RSVP scope. The RSVP Agent obtains a service policy for which traffic is mapped (if any) from the Policy Agent when an application using RAPI indicates it is a sender (when the Tspec is first provided), or when it requests a reservation as a receiver (when the Rspec is first provided for Guaranteed service). At both of these times, if a service policy is defined that maps to the data traffic, the RSVP Agent uses values in the service policy to limit the request from the application. Specifically, the following are limited:

- Total number of RSVP flows.
  The MaxFlows keyword on the PolicyAction statement of the policy definition can be used to limit the total number of application flows that use RSVP services.

- Tspec token bucket values.
  The MaxRatePerFlow and MaxTokenBucketPerFlow keywords on the PolicyAction statement of the policy definition can be used to limit the r and b values, respectively, in the sender supplied Tspec.

- Rspec values.
  The MaxRatePerFlow keyword on the PolicyAction statement of the policy definition can be used to limit the R value in the receiver supplied Rspec.

- Reservation type.
  The FlowServiceType keyword on the PolicyAction statement of the policy definition can be used to limit the type of reservation requested. A Guaranteed type request is considered to be "greater than" a Controlled Load type request. So if an application requests Guaranteed, but the policy limits the type to Controlled Load, the reservation uses Controlled Load.

RSVP processing proceeds as follows.

When an application uses RAPI to indicate it is a sender, the RSVP Agent packages the sender Tspec (along with other information) in an RSVP PATH packet, and sends the packet to the final destination. The packet is sent using RAW sockets, with the IP Router Alert option set. This option causes each router that supports RSVP to intercept the PATH packet, for the purpose of remembering the PATH request, and to insert a "previous hop" object into the packet, which is then sent again to the final destination. This causes the packet to eventually arrive at the destination, with all RSVP routers in the data path aware of the RSVP flow.

At the destination, the RSVP Agent passes the PATH packet to the application, using RAPI. The receiver application uses the Tspec and other information to arrive at a reservation request (flowspec). The receiver application uses RAPI to pass this flowspec to the RSVP Agent. The RSVP Agent then sends an RSVP RESV packet (containing the flowspec and other information) to the previous hop.

Each router or host along the path back to the sender receives this RESV packet, uses the flowspec to install the appropriate reservation (if possible), and forwards
the RESV to its previous hop. In this way, each RSVP-capable router or host along
the data path installs the reservation according to its capabilities. At the sender, the
RSVP Agent passes the RESV packet information to the sender application, which
then has information that indicates the actual reservation in place. The sender
might choose to wait for the reservation to be in place, or might begin sending
data before this happens (although such data is treated by the network as though
no reservation were in place). Any router or host that is incapable of supporting
the requested reservation might send an error to the receiver, which is then free to
perhaps try a lesser reservation.

The z/OS UNIX RSVP agent can provide actual resource reservations on ATM
interfaces. The RSVP agent passes the reservation request to the TCP/IP stack,
where a bandwidth reserved SVC is established on the ATM link to support the
reservation request. The RSVP agent can also cause the Type of Service (TOS) byte
to be set for any given RSVP flow, by using the OutgoingTOS keyword on the
PolicyAction statement of a defined service policy.

Gathering diagnostic information

The RSVP Agent writes logging information to a log file. The level of logged
information is controlled by the LogLevel configuration statement. By default, only
error and warning messages are written. To gather more diagnostic information,
you can specify a LogLevel value. The maximum information is logged with a
LogLevel value of 511. Refer to the z/OS Communications Server: IP Configuration
Guide for more details on using LogLevel, as well as the location of the log file.

The following information can also be useful in diagnosing RSVP Agent problems:
• Output from the TSO NETSTAT SLAP or netstat -j commands
• Output from the pasearch command for RSVP scoped policies
• SNMP output from walks of the Network SLAPM2 Subagent MIB tables
• TCP/IP CTRACE output, using the INTERNET and IOCTL CTRACE options
• Policy Agent log output if RSVP scoped policies are defined

Diagnosing RSVP agent problems

Problems with the RSVP agent generally fall into one of the following categories:
• Initialization Problems
• Application Problems
• Service Policy Problems

Initialization problems

Before you begin: If the RSVP Agent does not complete initialization, run it with
LogLevel set to 511 and check the log file for error conditions.

Common problems are listed in Table 60.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause or action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSVP Agent not authorized to security product</td>
<td>The RSVP Agent must be authorized to a security product profile. Refer to the z/OS Communications Server: IP Configuration Guide for details on setting up the proper authorization.</td>
</tr>
</tbody>
</table>
### Table 60. Common RSVP initialization problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause or action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unable to read configuration file</td>
<td>Is the correct configuration file specified? Refer to the <a href="#">z/OS Communications Server: IP Configuration Guide</a> for the search order used to locate the configuration file. Does the file exist? Are the permission bits correctly set for a z/OS UNIX file system file?</td>
</tr>
<tr>
<td>Unable to associate with the TCP/IP stack</td>
<td>Is the associated TCP/IP stack started? The RSVP Agent uses the TCP/IP image name specified in the configuration file, or uses the standard resolver search order, to locate the name of the TCP/IP stack. The log file indicates the stack name being used.</td>
</tr>
<tr>
<td>Unable to initialize interfaces</td>
<td>The RSVP Agent needs to initialize each interface for which it is configured. A pair of &quot;mailboxes&quot; are created for each interface. Check for error messages while creating the &quot;rsvp&quot; and &quot;rsvp-udp&quot; mailboxes for each interface. An error received while trying to join a multicast group on an interface that is not multicast capable is expected, and looks like: WARNING:.....mailslot_create: setsockopt(MCAST_ADD) failed - EDC5121I Invalid argument.</td>
</tr>
</tbody>
</table>

### Application problems

#### Before you begin:
Determine if a Qos-aware application using RAPI is experiencing problems

If so, check the items listed in Table 61.

### Table 61. RSVP application problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause or action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAPI DLL not found</td>
<td>An application using RAPI must have access to the RAPI DLL at run time. This is normally accomplished with the LIBPATH environment variable. Check that the LIBPATH environment variable is specified and that it contains the directory in which the RAPI DLL (rapi.dll) resides, which should be /usr/lib.</td>
</tr>
<tr>
<td>Error RAPI_ERR_NORSVP received</td>
<td>If the application receives a RAPI(ERR_NORSVP) error code when calling a RAPI function, ensure that the RSVP Agent has been successfully started.</td>
</tr>
</tbody>
</table>

### Policy problems

#### Before you begin:
Determine if you are having problems with policies with RSVP scope. Policies with RSVP scope can be defined and made available by way of the Policy Agent.
If problems are encountered using such policies, check the items listed in Table 62.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause or action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSVP policies not being applied to data flows</td>
<td>If the limits imposed by defined RSVP-scoped policies are not taking effect, check that the Policy Agent has been successfully started. The Policy Agent must be active in order for the RSVP Agent to retrieve these policies. Check that the policies are correctly defined. For example, do not specify both inbound and outbound interfaces in a single policy condition because such a policy never maps to any traffic on an end host node. Also, check both the RSVP Agent and Policy Agent log files for errors dealing with obtaining policies.</td>
</tr>
</tbody>
</table>
| Policy values not being used or are incorrect | If the values being used in the policies to limit Tspec and Rspec values do not appear to be correct, or do not seem to be applied to RSVP data traffic, be aware that the service policy and Tspec/Rspec units of measure are different. Specifically, the following are different: If the Service Policy Unit is:  
  • MaxRatePerFlow: kilobits/second, the Tspec/Rspec Unit is r/R: bytes/second  
  • MaxTokenBucketPerFlow: kilobits, the Tspec/Rspec Unit is b: bytes  
  To arrive at the values to specify on the service policy, multiply the target Tspec/Rspec value by 8, then divide by 1000. For example, if the target Tspec b value is 6000, the corresponding MaxTokenBucketPerFlow value is 48 (6000 x 8 / 1000 = 48). See Chapter 26, “Diagnosing Policy Agent problems,” on page 653 for more information about Policy Agent. |

Example log file

Figure 91 on page 687 demonstrates some of the RSVP Agent processing. This log file was created using a LogLevel of 511.

Lines with numbers displayed like 1 are annotations that are described following the log.
Figure 91. RSVP Agent processing log (Part 1 of 6)
03/22 08:51:06 INFO :....mailslot_create: creating mailslot for RSVP
03/22 08:51:06 INFO :....mailbox_register: mailbox allocated for rsvp
03/22 08:51:06 INFO :....mailslot_create: creating mailslot for RSVP via UDP
03/22 08:51:06 INFO :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE :..entity_initialize: interface 127.0.0.1, entity for rsvp allocated and initialized
03/22 08:51:06 INFO :....mailslot_create: creating socket for querying route
03/22 08:51:06 INFO :....mailbox_register: no mailbox necessary for forward
03/22 08:51:06 INFO :....mailslot_create: creating mailslot for route engine - informational socket
03/22 08:51:06 TRACE :....mailslot_create: ready to accept informational socket connection
03/22 08:51:11 INFO :....mailbox_register: mailbox allocated for route
03/22 08:51:11 INFO :....mailslot_create: creating socket for traffic control module
03/22 08:51:11 INFO :....mailbox_register: no mailbox necessary for traffic-control
03/22 08:51:11 INFO :....mailslot_create: creating mailslot for RSVP client API
03/22 08:51:11 INFO :....mailbox_register: mailbox allocated for rsvp-api
03/22 08:51:11 INFO :....mailslot_create: creating mailslot for terminate
03/22 08:51:11 INFO :....mailbox_register: mailbox allocated for terminate
03/22 08:51:11 INFO :....mailslot_create: creating mailslot for dump
03/22 08:51:11 INFO :....mailbox_register: mailbox allocated for dump
03/22 08:51:11 INFO :....mailslot_create: creating mailslot for (broken) pipe
03/22 08:51:11 INFO :....mailbox_register: mailbox allocated for pipe
03/22 08:51:11 INFO :...main: rsvpd initialization complete
03/22 08:52:50 INFO :......rsvp_api_open: accepted a new connection for rapi
03/22 08:52:50 INFO :.......mailbox_register: mailbox allocated for mailbox
03/22 08:52:50 TRACE :......rsvp_event_mapSession: Session=9.67.116.99:1047:6 does not exist
03/22 08:52:50 EVENT :.....api_reader: api request SESSION
03/22 08:52:50 TRACE :......event_establishSessionSend: found outgoing if=9.67.116.98 through forward engine
03/22 08:52:50 TRACE :......rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
03/22 08:52:50 EVENT :.....api_reader: api request SENDER
03/22 08:52:50 INFO :........init_policyAPI: papi_debug: Entering
03/22 08:52:50 INFO :........init_policyAPI: papi_debug: papiLogFunc = 98681F0 papiUserValue = 0
03/22 08:52:50 INFO :........init_policyAPI: papi_debug: Exiting
03/22 08:52:50 INFO :........init_policyAPI: APIInitialize: Entering
03/22 08:52:50 INFO :........init_policyAPI: open_socket: Entering
03/22 08:52:50 INFO :........init_policyAPI: open_socket: Exiting
03/22 08:52:50 INFO :........init_policyAPI: APIInitialize: ApiHandle = 9BDDB80, connfd = 22
03/22 08:52:50 INFO :........init_policyAPI: APIInitialize: Exiting
03/22 08:52:50 INFO :........init_policyAPI: RegisterWithPolicyAPI: Entering

Figure 91. RSVP Agent processing log (Part 2 of 6)
03/22 08:52:50 INFO :.......init_policyAPI: RegisterWithPolicyAPI: Writing to socket = 22
03/22 08:52:50 INFO :.......init_policyAPI: ReadBuffer: Entering
03/22 08:52:51 INFO :.......init_policyAPI: ReadBuffer: Exiting
03/22 08:52:51 INFO :.......init_policyAPI: RegisterWithPolicyAPI: Exiting
03/22 08:52:51 INFO :.......init_policyAPI: Policy API initialized
03/22 08:52:51 INFO :.......rpapi_getPolicyData: RSVPPFindActionName: Entering
03/22 08:52:51 INFO :.......rpapi_getPolicyData: ReadBuffer: Entering
03/22 08:52:51 INFO :.......rpapi_getPolicyData: ReadBuffer: Exiting
03/22 08:52:51 INFO :.......rpapi_getPolicyData: RSVPPFindActionName: Result = 0
03/22 08:52:51 INFO :.......rpapi_getPolicyData: RSVPFindActionName: Exiting
03/22 08:52:51 INFO :......api_reader: appl chose service type 1
03/22 08:52:51 INFO :......rpapi_getSpecData: RSVPGetTSpec: Entering
03/22 08:52:51 INFO :......rpapi_getSpecData: RSVPGetTSpec: Result = 0
03/22 08:52:51 INFO :......rpapi_getSpecData: RSVPGetTSpec: Exiting
03/22 08:52:51 TRACE :......api_reader: new service=1, old service=0
03/22 08:52:51 INFO :......rserv_flow_stateMachine: state SESSIONED, event PATHDELTA
03/22 08:52:51 INFO :......rserv_flow_stateMachine: state PATHED
03/22 08:52:51 TRACE :......mailslot_send: sending to (9.67.116.99:0)
03/22 08:52:51 TRACE :......mailslot_send: sending to (9.67.116.99:1698)
03/22 08:52:51 INFO :........qosmgr_request: [CL r=90000 b=6000 p=110000 m=1024 M=2048]
03/22 08:52:51 INFO :........qosmgr_request: ioctl to add reservation successful
03/22 08:52:51 INFO :.........rpapi_Reg_UnregFlow: RSVPPutActionName: Entering

Figure 91. RSVP Agent processing log (Part 3 of 6)
Figure 91. RSVP Agent processing log (Part 4 of 6)
Figure 91. RSVP Agent processing log (Part 5 of 6)
Following are short descriptions of the numbered items in the trace:

03/22 08:54:24 INFO :......rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:54:24 TRACE :.......rsvp_action_nHop: constructing a PATH
03/22 08:54:24 TRACE :.......flow_timer_start: started T1
03/22 08:54:24 TRACE :......rsvp_flow_stateMachine: reentering state RESVED
03/22 08:54:24 TRACE :.......mailslot_send: sending to (9.67.116.99:0)
03/22 08:54:35 TRACE :......rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
03/22 08:54:35 EVENT :.....api_reader: api request SENDER_WITHDRAW
03/22 08:54:35 INFO :.......rsvp_flow_stateMachine: state RESVED, event PATHTEAR
24
03/22 08:54:35 TRACE :........traffic_action_oif: is to remove filter
03/22 08:54:35 INFO :.......qosmgr_request: Ioctl to remove reservation successful
03/22 08:54:35 INFO :..........rpapi_Reg_UnregFlow: RSVPRemActionName: Entering
03/22 08:54:35 INFO :..........rpapi_Reg_UnregFlow: ReadBuffer: Entering
03/22 08:54:35 INFO :..........rpapi_Reg_UnregFlow: ReadBuffer: Exiting
25
03/22 08:54:35 INFO :..........rpapi_Reg_UnregFlow: RSVPRemActionName: Result = 0
03/22 08:54:35 INFO :..........rpapi_Reg_UnregFlow: RSVPRemActionName: Exiting
26
03/22 08:54:35 TRACE :...........rsvp_action_nHop: constructing a PATHTEAR
03/22 08:54:35 TRACE :........flow_timer_stop: stopped T1
03/22 08:54:35 TRACE :........flow_timer_stop: Stop T4
27
03/22 08:54:35 TRACE :......rsvp_flow_stateMachine: entering state SESSIONED
03/22 08:54:35 TRACE :.......mailslot_send: sending to (9.67.116.99:0)
28
03/22 08:54:35 EVENT :......api_reader: api request CLOSE
03/22 08:54:35 INFO :.......rsvp_flow_stateMachine: state SESSIONED, event PATHTEAR
03/22 08:54:35 PROTOERR:......rsvp_flow_stateMachine: state SESSIONED does not expect event PATHTEAR
29
03/22 08:54:53 EVENT :..mailslot_sitter: process received signal SIGTERM
03/22 08:54:53 INFO :......check_signals: received TERM signal
03/22 08:54:53 INFO :.......term_policyAPI: UnRegisterFromPolicyAPI: Entering
03/22 08:54:53 INFO :.......term_policyAPI: ReadBuffer: Entering
03/22 08:54:53 INFO :.......term_policyAPI: ReadBuffer: Exiting
03/22 08:54:53 INFO :.......term_policyAPI: UnRegisterFromPolicyAPI: Result = 0
03/22 08:54:53 INFO :.......term_policyAPI: UnRegisterFromPolicyAPI: Exiting
03/22 08:54:53 INFO :.......term_policyAPI: APITerminate: Entering
03/22 08:54:53 INFO :.......term_policyAPI: APITerminate: Exiting
03/22 08:54:53 INFO :.......term_policyAPI: Policy API terminated
03/22 08:54:53 INFO :......dreg_process: deregistering process with the system
03/22 08:54:53 INFO :......dreg_process: attempt to dereg (ifaeddrg_byaddr)
03/22 08:54:53 INFO :......dreg_process: rc from ifaeddrg_byaddr rc =0
03/22 08:54:53 INFO :......terminator: process terminated with exit code 0

Figure 91. RSVP Agent processing log (Part 6 of 6)
The RSVP Agent is started.

The configuration file being used is reported.

The name of the TCP/IP stack that the RSVP Agent associates itself with is reported.

The name and IP address of the interfaces configured to the associated stack are reported. Note that the RSVP Agent gets notified by the stack of any interface additions, deletions, or changes after this point.

The interfaces are initialized one by one.

Some interface types are not enabled for multicasting. Therefore, when the RSVP Agent tries to enable multicasting, a warning is reported. Such interfaces can still be used for unicasting.

RSVP Agent initialization is complete.

An application makes its first RAPI call, initializing the RAPI interface with the RSVP Agent.

The type of RAPI request is SESSION, meaning a rapi_session() call was made.

The RSVP Agent determines what the application sends based on the specified destination address not being a local interface.

The outbound interface to use for the session is returned from the stack.

The application issues a rapi_sender() call, passing the Tspec.

The Policy Agent interface is initialized.

The policy action "CLCat2" is obtained from the Policy Agent for the specified flow.

The RSVP Agent constructs an RSVP PATH packet to be sent to the destination.

The flow enters the pathed state (PATHED), meaning a PATH packet has been sent for the flow.

An RSVP RESV packet is received from the RSVP Agent at the receiver node, specifying the reservation parameters.

The RSVP Agent installs the reservation request into the TCP/IP stack and registers the flow with the Policy Agent.

The type of reservation request is shown (CL, for Controlled Load) along with the reservation parameters (the r, b, p, m, M values in Tspec format).

The RESV packet values are passed to the sender application.

The flow enters the reserved state (RESVED), meaning the reservation has been put in place and the RESV packet has been forwarded to the previous hop (in this case the sender application).

A T1 timeout occurs, meaning a PATH refresh packet is sent. This occurs every 15 seconds.

A refreshed RESV packet is received from the RSVP Agent at the receiver node. This occurs every 30 seconds.

The application issues a rapi_release() call to end the RAPI session.

The reservation is removed from the TCP/IP stack and unregistered from the Policy Agent.
A PATHTEAR packet is constructed and sent, to tear down the flow along the data path.

The flow enters the sessioned state (SESSIONED), meaning that the flow has been torn down.

The application closes the API session, resulting in an error being reported because the state of the flow is SESSIONED. This error can be ignored.

A SIGTERM signal is received (due to a kill command issued from the UNIX shell), and the RSVP Agent shuts itself down.
Chapter 28. Diagnosing intrusion detection problems

This topic provides information and guidance to diagnose Intrusion Detection Service (IDS) problems, including traffic regulation management daemon (TRMD) related problems. It contains the following sections:

- "Overview"
- "Diagnosing IDS policy problems"
- "Diagnosing IDS output problems" on page 696
- "Diagnosing TRMD problems" on page 699
- "Documentation for the IBM Software Support Center" on page 700

Overview

The Intrusion Detection Services policy is installed into the stack by the Policy Agent (PAGENT). After the policy is installed, IDS detects, processes, and reports on events as requested by the policy. TRMD, part of IDS, handles reporting IDS statistics and events to syslogd. Problems might occur in the following areas:

- Policy installation
- Output to syslogd, the console, or the IDS trace missing or volume too high
- TRMD initialization

Diagnosing IDS policy problems

This section describes the commands used to diagnose IDS policy problems.

Step for determining which IDS policies are active in Policy Agent

Before you begin: If you are running multiple stacks, ensure that pasearch is reporting on the stack you are interested in.

- Use `pasearch` (refer to `z/OS Communications Server: IP System Administrator's Commands`) to see what IDS policies are active in Policy Agent.

See Chapter 26, “Diagnosing Policy Agent problems,” on page 653 if you do not see the IDS policies expected.

Step for determining how your IDS policies have been mapped by the stack

Before you begin: If you are running multiple stacks, ensure that your resolver configuration correctly identifies the stack you are interested in. Ensure that your IDS policies are correctly defined.

- Use `NETSTAT IDS` or `netstat -k` (refer to `z/OS Communications Server: IP System Administrator's Commands`) to see how your IDS policies have been mapped by the stack.

Refer to IDS policy considerations in the `z/OS Communications Server: IP Configuration Guide`.
Some IDS policies are not mapped until they are needed. Attack, Scan Global and Scan Event for protocol ICMP are mapped immediately when the policy is installed in the stack. Scan Event policies for protocols TCP and UDP are mapped on the first occurrence of a potentially countable event. TR policies for protocol TCP are mapped when a local application does a listen() and when a client completes the three-way connection handshake. TR policies for protocol UDP are mapped when an inbound datagram arrives for a bound port.

---

Diagnosing IDS output problems

The following describe diagnostic steps for some problems you might encounter.

**Steps for determining why IDS syslogd output is missing**

Perform the following steps to determine the cause for IDS syslogd missing.

1. Ensure that Policy Agent is running on this system.

2. Ensure that TRMD is running for this stack on this system. Consider using TCPIP PROFILE Autolog for TRMD.

3. Ensure that syslogd is running on this system.

4. Ensure that syslogd is configured for IDS output:
   - TRMD always writes to the syslog daemon facility.
   - Events are written to the syslog level configured in the relevant policy. Statistics are always written to INFO level.
   - If running multiple TRMDs, consider using trmd jobname prefix to separate IDS output by stack.

---

**IDS console output**

Under certain conditions, IDS suppresses console messages to avoid flooding the system console.

Scan detection is reported at most once per fast scan interval for a particular source IP address. If a scan is continually detected for the same source IP address, consider adding this address to your scan exclusion list (if this user is legitimately accessing resources). The installation also has the option of requesting notification to syslogd rather than to the console. The same criteria is used for reporting scans to syslogd as to the console.

IDS attack policy actions support the maximum event message parameter. If specified, this limits the number of times the same attack type is reported to the system console within any 5-minute time period.

Traffic regulation for protocol TCP suppresses console reporting of following three events that could occur repeatedly:
   - Only the first connection denied, when an application exceeds the TR TCP total connections limit, is reported during each port constrained period.
Only the first connection denied, when a source host exceeds the TR TCP percentage available limit, is reported until the number of connections by that source host to this application drops below 88% of the limit and at least 2 connections below the limit.

Connections that would exceed the TR TCP percentage of available connections per source host, but are allowed because of a higher value in QoS policy, are reported to syslogd only.

**IDS packet trace output**

Use the following references or recommended actions for IDS packet trace output:

- See “Intrusion Detection Services trace (SYSTCPIS)” on page 149 if message EZZ4210I CTRACE DEFINE FAILED FOR CTIIDS00 is issued at stack initialization.

- **Consider starting** the MVS external writer. See “Formatting packet traces using IPCS” on page 97 for information on formatting the IDS packet trace in a dump.

- For IDS attack policy, the tracing action allows packets associated with attack events to be traced. For all attack categories except flood, a single packet triggers an event and the packet is traced. To prevent trace flooding, a maximum of 100 attack packets per attack category are traced within a 5–minute interval. For the flood category, the first 100 packets discarded during the flood are traced.

**Unusual conditions**

Most messages issued by IDS relate to the detection of an IDS condition. However, the messages mentioned below should be investigated because they signal conditions which affect IDS normal processing that might result in IDS information being lost or delayed.

**Buffer overflow transferring message data between the stack and TRMD**

The following messages in syslog indicate that IDS events or statistics are being generated at a rate that is overflowing internal buffers used to relay the messages from the stack to TRMD. These messages are a warning that actual event or statistics messages are missing from the syslog. If these messages occur frequently, then IDS policy changes are necessary to reduce the amount of IDS logging, or the amount of statistics information, being generated.

EZZ9325I TRMD Log records missing: logtype,logmissing
EZZ9326I TRMD Statistics records missing: stattype,statmissing

**Repeated attacks of the same type at a high rate**

A message is issued in syslog to indicate that attack policy is in place and the attack type indicated is occurring repeatedly at a high rate. To avoid flooding syslog and conserve system resources, a maximum of 100 event messages per attack type are logged to syslogd within a 5–minute interval. This limit is always in effect. The following message indicates the number of duplicate attacks for which messages have been suppressed.

EZZ9327I TRMD Attack log records suppressed: attack_type,count

**Scan storage constrained**

The following is an example of a console message issued if scan detection attempted to obtain storage in order to track a potential scan event and could not obtain the required amount of storage.

EZZB761I IDS EVENT DETECTED
EZZB762I EVENT TYPE: SCAN STORAGE CONSTRAINED
EZZB763I CORRELATOR 0 - PROBEID @308FFF3
EZZB766I IDS RULE N/A
EZZB767I IDS ACTION N/A
Processing continues without adding the tracking information for this packet or for subsequent packets in the current internal interval (an internal interval is either 30 or 60 seconds). This could result in missing potential scan events.

The installation should attempt to determine the cause of the storage shortage. Scan detection itself can potentially consume large amounts of storage and should be looked at as part of the problem determination. The following are two ways to determine whether scan is consuming large amounts of storage:

- Console message EZZ8768I (EZZ8768I IDS SCAN STORAGE EXCEEDED nbrmeg MB, TRACKING nbrsip SOURCE IP ADDRESSES) is issued after scan detection acquires more than a megabyte of storage. This message is reissued at each power of 2 MB increments (for example, 1 MB, 2 MB, 4 MB, 8 MB, and so forth).
- The Netstat IDS command displays high level scan information. For example:

```
SCAN DETECTION:
  GLOBRULENAME: IDS-RULE4
  IMPRULENAME: IDS-RULE8
  TOTDETECTED: 1 DETCURRPLC: 1
  DETCURRINT: 0 INTERVAL: 30
  SRCIPSTRKD: 125 STRGLEV: 00000M
```

The SRCIPSTRKD field indicates the number of source IPs being tracked and the STRGLEV field indicates the number of megabytes of storage that scan is holding.

If scan processing is contributing to the storage shortage, consider changing the scan policy. If the installation has set the scan sensitivity to HIGH on high usage ports, consider reducing the sensitivity level or removing the port from scan detection until the storage constraint is resolved.

When scan starts to successfully obtain storage again, a SCAN STORAGE UNCONSTRAINED message is issued.

### Excessive processing time for scans

The following is an example of a console message issued as a result of excessive processing time for scans:

```
EZZ8761I IDS EVENT DETECTED
EZZ8762I EVENT TYPE: INTERFACE FLOOD DETECTION DISABLED
EZZ8763I CORRELATOR 0 - PROBEID 04070015
EZZ8766I IDS RULE N/A
EZZ8767I IDS ACTION N/A
```

If an installation repeatedly receives this message, scan processing is not able to complete its evaluation of the source IP addresses it is tracking in its normal interval (either 30 or 60 seconds). This could delay the detection of subsequent scans. This most likely indicates that a large number of source IP addresses are being monitored. If the policy is using high scan sensitivity, the installation should consider lowering the scan sensitivity level for high usage ports.

### Interface flood detection disabled

In order to track data for interface flood detection, private storage is obtained when IDS starts monitoring an interface. If the storage cannot be obtained, IDS is not able to detect an interface flood for the interface. A console message and a syslogd message are issued to report the condition.

The following is an example of the console message that is issued:

```
EZZ8761I IDS EVENT DETECTED
EZZ8762I EVENT TYPE: INTERFACE FLOOD DETECTION DISABLED
EZZ8763I CORRELATOR 0 - PROBEID 04070015
```
The following is an example of the syslogd message:

```
EZZ8658I TRMD ATTACK Interface Flood Detection Disabled:12/23/2002 20:39:35.00,
ifcname=OSAQDIO4L, dipaddr=5.72.107.78,correlator=20,probeid=04070015,
sensorhostname=MVS34.tcp.com
```

These messages indicate a storage constraint has prevented the initialization of interface flood detection for the interface specified in the message. Interface flood detection for other interfaces is not affected.

When the problem causing the storage constraint is resolved, the Interface Flood detection support can be activated by removing the IDS ATTACK FLOOD policy and then adding the IDS ATTACK FLOOD policy again, or by stopping and restarting the interface.

**Interface flood storage constrained**

The following message in syslogd indicates that private storage needed in order to collect informational data related to a possible interface flood condition could not be obtained:

```
EZZ8659I TRMD ATTACK Interface Flood storage constrained:timestamp,ifcname=ifcname,
dipaddr=dipaddr,correlator=correlator,probeid=04070016,sensorhostname=sensorhostname
```

The informational data provided by the EZZ8655I and EZZ8656I syslogd messages issued for the interface in the same time period might be incomplete. Collection of informational data for the interface that requires additional storage is temporarily suspended and resumes at the start of the next one-minute interval.

### Diagnosing TRMD problems

The most common type of TRMD problem is initialization.

The TRMD writes logging information to a log file. The level of logged information is controlled by the `-d` startup option. To gather more diagnostic information, you can start the TRMD with the `-d` startup option. The maximum information is logged with the `-d 3` option. Log output is directed to the syslog daemon (syslogd). Refer to the [z/OS Communications Server: IP Configuration Reference](#) for more details on using the `-d` startup option.

Problems with initialization of the TRMD include the following:

- **Starting TRMD from the console.**
  - TRMD might fail with an ABEND=S000 U4093 REASON=00000090 because an OMVS segment was not defined for the TRMD ID.

  Check the job output.

  ```
  IEF403I TRMD - STARTED - TIME=12.48.55
  ICH408I JOB(TRMD ) STEP(TRMD ) CL(PROCESS )
  OMVS SEGMENT NOT DEFINED
  IEE995I SYMPTOM DUMP OUTPUT
  USER COMPLETION CODE=4093 REASON CODE=00000090
  TIME=12.48.58 SEQ=00065 CPU=0000 ASID=002B
  PSW AT TIME OF ERROR 078D1000 8000AA7A ILC 2 INTC 0D
  ACTIVE LOAD MODULE ADDRESS=00007E70 OFFSET=00
  NAME=CEEBINIT
  DATA AT PSW 0000AA74 ~ 00181610 B10A1811
  GR 0: B4000000 1: B40000FD
  ```

Chapter 28. Diagnosing intrusion detection problems 699
Verify that an OMVS segment exists for TRMD by issuing the lu TSO command from a user ID that has authority to issue the LU command: LU trmd OMVS. If an OMVS segment does not exist, use the ALU command to update the user's OMVS data. For example, ALTUSER trmd OMVS(UID(0000) HOME('/')) PROGRAM('/bin/sh').

- The TCP/IP stack is not up and message EZZ8498I is received. Verify that the TCP/IP stack is up.

**Documentation for the IBM Software Support Center**

When contacting the IBM Software Support Center for problem resolution, some or all of the following information might be required:

- Gather TRMD debugging data by starting TRMD with the `trmd -d 3` command. See “Diagnosing TRMD problems” on page 699.
- Start CTRACE in the stack to gather related information. See “Component trace” on page 47.
- The output from the `pasearch -i` command. Refer to z/OS Communications Server: IP System Administrator’s Commands.
- The output from the Netstat IDS/-k command. Refer to z/OS Communications Server: IP System Administrator’s Commands.
Chapter 29. Diagnosing Application Transparent Transport Layer Security (AT-TLS)

AT-TLS transparently performs Transport Layer Security (TLS) on behalf of the application by invoking the z/OS System Secure Socket Layer (SSL) in the TCP transport layer. System SSL provides support for the TLSv1, SSLv3, and SSLv2 protocols. AT-TLS uses a policy-based configuration, and the Policy Agent application is required to define rules and actions to the TCP/IP stack for TCP connections using AT-TLS. Displays for AT-TLS policy are provided by `pasearch` and `Netstat`.

This topic describes how to diagnose AT-TLS problems and includes the following sections:

- "Common AT-TLS startup errors"
- "Steps for diagnosing AT-TLS problems"
- "AT-TLS traces" on page 703
- "AT-TLS return codes" on page 706
- "SIOCTTLSCTL ioctl return codes" on page 712

Common AT-TLS startup errors

The following list describes startup errors, possible causes, and actions to take.

- If message EZZ4248E is written to the console and not released, one of the following might have occurred:
  - Policy Agent has not been started.
  - Policy Agent configuration does not contain a TCPImage statement for this stack, or the stack policy configuration does not contain any local or remote AT-TLS policies.
  - Policy Agent is not permitted to create a socket with this stack. Ensure that the SERVAUTH class is active. Ensure that the EZB.INITSTACK.mvsname.tcpname resource profile is defined and that Policy Agent is permitted to it. If the EZB.STACKACCESS.mvsname.tcpname resource profile is defined, ensure that Policy Agent is permitted to it.

- If applications started after the stack fail to create a socket (errno EAGAIN, errno2 JrTcpNotActive), the stack is probably being configured for AT-TLS, and the application has been started before AT-TLS policy has been installed. If this is a required network infrastructure application, permit it to the EZB.INITSTACK.mvsname.tcpname resource profile in the SERVAUTH class. If it is not a required network infrastructure application, either start it after message EZZ4248E is released or modify the application to wait a short period of time and retry when the errno is EAGAIN.

- If message EZD1287I TTLS Error RC: 5020 Group Init is displayed, the TCP/IP stack was not able to load the System SSL DLL required for AT-TLS processing.

Steps for diagnosing AT-TLS problems

Perform the following steps to diagnose AT-TLS problems.
1. Issue `pasearch -t` to see all AT-TLS policies that are active in Policy Agent. Refer to [z/OS Communications Server: IP System Administrator’s Commands](z/OS_Communications_Server:_IP_System_Administrator%27s_Commands) for more information about the `pasearch -t` command. If you are running multiple stacks, ensure that `pasearch` is reporting on the stack you are interested in. If you do not see the AT-TLS policies that you expected, refer to [z/OS Communications Server: IP System Administrator’s Commands](z/OS_Communications_Server:_IP_System_Administrator%27s_Commands) for more information about displaying policy based networking information.

2. Issue `Netstat TTLS CONn connid` or `Netstat -x CONn connid` to determine whether the stack mapped a connection to AT-TLS policy and, if so, to which policy it was mapped. For more information about the `netstat` commands, refer to [z/OS Communications Server: IP System Administrator’s Commands](z/OS_Communications_Server:_IP_System_Administrator%27s_Commands). Ensure that your AT-TLS policies are correctly defined. Refer to the AT-TLS information in [z/OS Communications Server: IP Configuration Guide](z/OS_Communications_Server:_IP_Configuration_Guide) and the AT-TLS Policy statements in [z/OS Communications Server: IP Configuration Reference](z/OS_Communications_Server:_IP_Configuration_Reference) for more information about configuring AT-TLS policies.

3. In cases where AT-TLS connections do not map to any policy, verify that TCPCONFIG TTLS has been specified. Netstat configuration shows the current setting of AT-TLS. AT-TLS connection mapping is performed based on the following attributes:
   - Local IP Address
   - Remote IP address
   - Local Port
   - Remote Port
   - Direction
   - Job name
   - User ID
The AT-TLS policy rules are searched, starting with the highest priority rules, for the first match.

Then the internal SecondaryMap table is searched by process ID and the two IP addresses used on the connection. The SecondaryMap table contains entries for active connections that are mapped by the AT-TLS policy rule to a policy with the SecondaryMap attribute specified as On. If entries are found using both methods, the one found by the AT-TLS policy rule is used unless the one found by the SecondaryMap value has a higher priority.

If a TCP connection is not matching the expected rule, do one of the following:
   - Ensure that the AT-TLS policies are active and that no errors occurred. Message EZZ8438I is issued if Policy Agent encountered any errors while processing the AT-TLS policy. If errors occurred, review the Policy Agent logs for details on the error and correct the AT-TLS policy. You can use OBJERR to search the Policy Agent logs to find the errors.
   - Verify the rule and actions that the policy mapped to and the priority of the rule. You can use the `pasearch` command can be used to view the active AT-TLS policy. AT-TLS message EZD1281I is issued with all the parameters used to map to the AT-TLS policy, if trace level 4 is on.

4. If an error message was issued by AT-TLS, review the syslogd files for message EZD1286I or the TCP/IP joblog for message EZD1287I. The error message might provide information about correcting the problem.

5. If the error is recreatable, turn on an AT-TLS trace for the connection. Turn on the trace by coding a TTLSRule specific to the failing connection. Include a
TTLSCConnectionAction statement that has the Trace statement set to 255 (All). If configuring using the IBM Configuration Assistant for z/OS Communications Server, the trace level can be set in each Connectivity Rule.

6. If the problem cannot be resolved from the trace, perform a packet trace or a Ctrace with option TCP to provide additional debugging information and contact IBM service.

AT-TLS traces

By default, AT-TLS uses the syslog facility name daemon. Other TCP/IP functions, for example the SNMP agent, also specify the daemon facility name when writing records to syslogd. The job name and syslog facility name are the same. Filters cannot be used to direct the records to different output files. If you want AT-TLS records to go to a different output file, you can change the syslog facility name by configuring SyslogFacility Auth on the TTLSCGroupAdvancedParms statement to direct the messages from that group to the Auth facility instead. You can then set up filtering based on the job name and facility in the syslogd configuration file to direct AT-TLS records to a different output file.

If you are configuring using the IBM Configuration Assistant for z/OS Communications Server, you can modify the syslog facility name from the AT-TLS: Image Level Settings panel.

AT-TLS traces are enabled by setting the AT-TLS policy statement Trace to a nonzero value. A Trace statement can be configured on a TTLSCGroupAction, TTLSEnvironmentAction or TTLSCConnectionAction statement. Refer to the z/OS Communications Server: IP Configuration Reference for more details about AT-TLS policy statements. The Trace levels enable different AT-TLS messages to be issued. The sum of the numbers associated with each level of tracing desired is the value that should be specified.

If you are configuring using the IBM Configuration Assistant for z/OS Communications Server, you can set the default trace level on the AT-TLS: Image Level Settings panel, and you can override the trace level for each Connectivity Rule.

Table 63 lists the trace level, the generated AT-TLS messages, and the syslog priority.

Table 63. AT-TLS trace levels

<table>
<thead>
<tr>
<th>Trace level</th>
<th>Traced information</th>
<th>Syslog priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Error (to Joblog)</td>
<td>EZD1287I</td>
<td>NA</td>
</tr>
<tr>
<td>2 Error</td>
<td>EZD1286I</td>
<td>err</td>
</tr>
<tr>
<td>4 - Info</td>
<td>EZD1281I, EZD1283I</td>
<td>info</td>
</tr>
<tr>
<td>8 - Event</td>
<td>EZD1282I, EZD1283I</td>
<td>debug</td>
</tr>
<tr>
<td>16 - Flow</td>
<td>EZD1282I, EZD1283I, EZD1284I</td>
<td>debug</td>
</tr>
<tr>
<td>32 - Data</td>
<td>EZD1285I</td>
<td>debug</td>
</tr>
</tbody>
</table>

Tip: Setting the Trace level to 6 enables both error messages and info messages.
The information messages trace when an AT-TLS connection is mapped to a policy (EZD1281I) and when the secure connection is successfully negotiated (EZD1283I), including the security protocol and cipher used. Using syslogd's filtering parameters, a separate log file could be kept for AT-TLS info and error messages, enabling AT-TLS connections to be tracked.

Tip: Trace level 32 shows all the SSL headers sent and received.

Each secure connection is uniquely identified by its connection ID (ConnID). You can use the ConnID to follow a connection through the AT-TLS trace.

Sample AT-TLS trace

Figure 92 on page 705 shows an example trace of a generic server processing a secure connection. The standard syslogd prefix information has been removed from the trace.

Trace level 255 was used to generate this trace.

11:10:25 TCPCS3 EZD1281I TTLS Map CONNID: 00000025 LOCAL: 9.42.104.156..21 REMOTE: 9.27.154.171..1271


ACTIONS: grp_act1 env_act_serv **N/A** 1

11:10:28 TCPCS3 EZD1283I TTLS Event GRPID: 00000001 ENVID: 00000000 CONNID: 00000025 RC: 0

Connection Init

11:10:28 TCPCS3 EZD1282I TTLS Start GRPID: 00000001 ENVID: 00000001 CONNID: 00000000 Environment Create

ACTIONS: grp_act1 env_act_serv **N/A** 2

11:10:28 TCPCS3 EZD1283I TTLS Event GRPID: 00000001 ENVID: 00000002 CONNID: 00000000 RC: 0

Environment Master

Create 00000001

11:10:28 TCPCS3 EZD1284I TTLS Flow GRPID: 00000001 ENVID: 00000000 CONNID: 00000025 RC: 0 Call

GSK_ENVIRONMENT_OPEN - 7F1DB058


GSK_KEYRING_FILE - FTPDsafkeyring 3


GSK_CLIENT_AUTH_TYPE - FULL


GSK_SESSION_TYPE - SERVER


GSK_PROTOCOL_SSLV2 - ON


GSK_PROTOCOL_SSLV3 - ON


GSK_PROTOCOL_TLSV1 - ON


GSK_IO_CALLBACK -


GSK_SSL_HW_DETECT_MESSAGE - 1

11:10:28 TCPCS3 EZD1284I TTLS Flow GRPID: 00000001 ENVID: 00000002 CONNID: 00000025 RC: 0 Call

GSK_ENVIRONMENT_INIT - 7F1DB058


GSK_SSL_HW_DETECT_MESSAGE - NULL

11:10:28 TCPCS3 EZD1283I TTLS Event GRPID: 00000001 ENVID: 00000001 RC: 0

Environment Link 7F1DB058 00000002

11:10:28 TCPCS3 EZD1282I TTLS Start GRPID: 00000001 ENVID: 00000000 CONNID: 00000025 Initial Handshake

ACTIONS: grp_act1 env_act_serv **N/A** 4

11:10:28 TCPCS3 EZD1284I TTLS Flow GRPID: 00000001 ENVID: 00000001 CONNID: 00000025 RC: 0 Call

GSK_SECURE_SOCKET_OPEN - /7F0CA118

11:10:28 TCPCS3 EZD1284I TTLS Flow GRPID: 00000001 ENVID: 00000001 CONNID: 00000025 RC: 0 Set

GSK_FD - 00000025

11:10:28 TCPCS3 EZD1284I TTLS Flow GRPID: 00000001 ENVID: 00000001 CONNID: 00000025 RC: 0 Set

GSK_USER_DATA - 7F1DB330
The following information corresponds to the line numbers in Figure 92.

1. A TCP connection has mapped to an AT-TLS rule. The parameters used to search the AT-TLS rules are listed. The TTLSRule, TTLSGroupAction, TTLSEnvironmentAction, and TTLSConnectionAction names are also displayed. Note the ConnID for the connection. This ConnID appears in all future AT-TLS messages for this connection.

2. AT-TLS is creating an environment instance for the application.

3. AT-TLS is establishing the parameters for this environment. These parameters are obtained from the TTLSEnvironmentAction statement. System SSL calls are made to set up the parameters. This trace message is defining the key ring to be used by this environment.
4. AT-TLS has successfully set up the secure environment and is now initializing the secure connection. This initiates network flows with the remote partner.
5. Secure data has been received for this connection. During secure handshake, all the data is traced. For this trace example, some of the data has been removed.
6. Secure data is being sent for this connection.
7. The secure handshake has completed. The protocol negotiated (TLSv1) and the cipher suite negotiated (05) are displayed.
8. AT-TLS is sending a secure alert message, because the application closed the socket.
9. The secure connection is being closed.

**AT-TLS return codes**

AT-TLS error message EZD1286I is issued to syslogd to report any errors that occur on a AT-TLS connection when the trace level 2 (Error) is set. AT-TLS error message EZD1287I is issued to the TCP/IP joblog to report any errors that occur on a AT-TLS connection when the trace level 1 (Error) is set. These messages include the event that AT-TLS was processing and the return code indicating a failure. Return codes between 5001 and 5999 describe AT-TLS errors that can be corrected by the user. Return codes between 6001 and 6999 describe internal AT-TLS errors. Contact IBM with the error message and syslog information, if available. Any other return code is defined by System SSL. Refer to [Cryptographic Services System SSL Programming](https://www.ibm.com) for additional information on these return codes. See Table 65 on page 708 for information about these return codes.

Table 64 lists some common System SSL return codes and possible causes.

<table>
<thead>
<tr>
<th>Return code</th>
<th>Event</th>
<th>Possible cause and solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>Environment Init</td>
<td>The key ring cannot be opened because the user does not have permission. Check the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Look at message EZD1281 to verify the user ID being used for this connection and the TTLSEnvironmentAction statement mapped to this connection. If you are configuring using the IBM Configuration Assistant for z/OS Communications Server, you can specify the key ring on either the AT-TLS: Image Level Settings panel or on each Traffic Descriptor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Ensure that the correct key ring has been specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If using a RACF key ring, verify that all the steps in <a href="https://www.ibm.com">z/OS Communications Server: IP Configuration Guide</a> have been followed for this user ID.</td>
</tr>
</tbody>
</table>
### Table 64. Common System SSL return codes (continued)

<table>
<thead>
<tr>
<th>Return code</th>
<th>Event</th>
<th>Possible cause and solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>402</td>
<td>Connection Init</td>
<td>A SSL cipher suite could not be agreed upon between the client and server. Check the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If V2Ciphers or V3Ciphers are coded, verify that the remote end supports at least one of the cipher suites coded. If configuring using the IBM Configuration Assistant for z/OS Communications Server, the ciphers are selected for each Security Level.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Verify that the certificate being used for the connection supports the cipher suites. For example, V3 Cipher suite TLS_DH_DSS_WITH_DES_CBC_SHA(0C) requires a certificate defined with a Diffie-Hellman key.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For ciphers defined as exportable, verify that the proper FMIDs to support the encryption level are installed.</td>
</tr>
<tr>
<td>406</td>
<td>Connection Init</td>
<td>An I/O error occurred on the socket. This occurs if the TCP socket is closed underneath the SSL protocol, such as when a reset is received. Check the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Ensure that the remote partner is enabled for secure connections.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Determine whether the secure negotiation has completed. Use the AT-TLS Data trace level to determine this.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Verify that the TCP data flows were sent by the remote partner. Use a TCP/IP packet trace to verify this.</td>
</tr>
<tr>
<td>412</td>
<td>Connection Init</td>
<td>A common SSL protocol type cannot be agreed upon by both partners. This occurs if both partners do not support the same SSL protocol, as when the client supports only SSLv2 and the server supports only TLSv1. AT-TLS supports only SSLv2, SSLv3, and TLSv1. Check the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Determine the protocols supported by the remote partner.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Code a TTLSEnvironmentAdvancedParms statement, which enables the common protocols. If configuring using the IBM Configuration Assistant for z/OS Communications Server, use a Security Level with cipher levels supported by the remote partner.</td>
</tr>
<tr>
<td>422</td>
<td>Connection Init</td>
<td>A v3Cipher that is not valid has been found.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Check the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Determine whether the v3Cipher statement has been coded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Verify that the proper SSL FMIDs are installed to support the ciphers specified.</td>
</tr>
</tbody>
</table>
Table 64. Common System SSL return codes (continued)

<table>
<thead>
<tr>
<th>Return code</th>
<th>Event</th>
<th>Possible cause and solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>434</td>
<td>Connection Init</td>
<td>The certificate key is not compatible with the negotiated cipher suite. Ensure that the certificate being used supports the cipher suites coded with V2Ciphers or V3Ciphers. If configuring using the IBM Configuration Assistant for z/OS Communications Server, the ciphers are selected in each Security Level.</td>
</tr>
</tbody>
</table>

Table 65 lists some common AT-TLS return codes and possible causes.

Table 65. AT-TLS return codes

<table>
<thead>
<tr>
<th>Return code</th>
<th>Event</th>
<th>Possible cause and solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>Connection Init</td>
<td>ClientAuthType is set to Required or SAFCheck, but the client did not provide a certificate. Verify that the client supports client authentication and is configured to send its certificate during secure negotiation.</td>
</tr>
<tr>
<td>5002</td>
<td>Connection Init</td>
<td>ClientAuthType is set to SAFCheck, but the certificate supplied by the client is not defined to SAF subsystem. If using RACF, define the client’s certificate with the RACDCERT command. For more information about using the RACDCERT command, refer to z/OS Security Server RACF Security Administrator’s Guide.</td>
</tr>
</tbody>
</table>
| 5003        | Connection Init | Clear text data was received on the connection from the remote partner instead of secure data. The connection has been terminated. Check the following:  
  • Ensure that the remote client is enabled for secure connections.  
  • If the policy is defined with ApplicationControlled On, ensure that the application read all the cleartext data before starting the secure handshake. If configuring using the IBM Configuration Assistant for z/OS Communications Server, the Application Controlled setting is done in each Traffic Descriptor. |
<table>
<thead>
<tr>
<th>Return code</th>
<th>Event</th>
<th>Possible cause and solution</th>
</tr>
</thead>
</table>
| 5004       | Initial handshake | The first HandshakeTimeout interval expired without secure data being received from the remote partner. The timer is set for the number of seconds specified by the HandshakeTimeout value when the secure connection is initiated. When the first secure data is received from the remote partner, the timer is cancelled. Check the following:  
  • This can occur if both sides of the connection are configured to be the server in the secure handshake. Review the configuration to ensure that one side acts as the client. For AT-TLS, you can specify the HandshakeRole value in either the TTLEnvironmentAction or the TTLSConnectionAction statement. If configuring using the IBM Configuration Assistant for z/OS Communications Server, configure the Handshake Role value in each Traffic Descriptor.  
  • Increase the HandshakeTimeout value if the remote partner is not responding within the time interval. If configuring using the IBM Configuration Assistant for z/OS Communications Server, you can set the Timeout value in each Traffic Descriptor; you can override the value in each Connectivity Rule. |
| 5005       | Initial Handshake | The second HandshakeTimeout interval expired and the secure handshake is not finished. This interval is set to 10 times the HandshakeTimeout interval. The secure negotiation started and the initial secure message was received from the remote partner.  
  • If the remote partner is an interactive application, such as requiring the user to select a certificate, either increase the HandshakeTimeout value or have the user retry the connection.  
  • The HandshakeTimeout value might need to be increased if LDAP is being used to manage certificates. Increasing the value provides more time for the LDAP processing to occur. If configuring using the z/OS Network Configuration Assistant, the Handshake Timeout value can be set in each Traffic Descriptor and can be overridden in each Connectivity Rule. |
Table 65. AT-TLS return codes (continued)

<table>
<thead>
<tr>
<th>Return code</th>
<th>Event</th>
<th>Possible cause and solution</th>
</tr>
</thead>
</table>
| 5006        | Connection Init | The connection is using a TTLSEnvironmentAction statement that failed to initialize a System SSL environment.  
  - Use the syslog to determine why the System SSL environment failed to initialize.  
  - If the TTLSEnvironmentAction statement is in error, make the necessary corrections. A System SSL environment is initialized for the corrected TTLSEnvironmentAction statement and new connections use that environment.  
  - If a SAF configuration change is needed (such as changing a certificate in the key ring), make that change and then update the EnvironmentUserInstance parameter in the TTLSEnvironmentAction statement to reflect a changed action. A System SSL environment is initialized using the modified RACF configuration and new connections uses that environment.  
  If configuring using the z/OS Network Configuration Assistant to pick up changes made to a key ring, go to the AT-TLS Image Level Settings panel and click the Reaccess Key Rings button and update the Instance ID for the changed key ring. |
| 5007        | Connection Init | Application data was read during processing of ciphertext negotiation. Collect the syslogd output or joblog output and contact IBM.                                                                                   |
| 5008        | Connection Init | Application data was received after the local application closed the TCP connection. The data could not be presented to the application.  
  - Review the local and remote applications to ensure that the TCP sockets are being closed correctly in the application flow.  
  - If further diagnostic information is needed, set the trace level to 255, to trace the data flow and AT-TLS processing. |
| 5009        | Connection Init | AT-TLS was unable to obtain TCPIP private storage. Obtain a console dump of TCPIP and contact IBM                                                                                                                     |
| 5010        | Connection Init | AT-TLS was unable to obtain the ACEE for an application. Save the syslogd output and contact IBM                                                                                                                         |
| 5011        | Connection Init | AT-TLS does not have a Envar object for the applications ACEE. Save the syslogd output and contact IBM                                                                                                                   |
| 5012        | Connection Init | An internal AT-TLS error has occurred. Save the syslogd output and contact IBM                                                                                                                                       |
Table 65. AT-TLS return codes (continued)

<table>
<thead>
<tr>
<th>Return code</th>
<th>Event</th>
<th>Possible cause and solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>5013</td>
<td>Connection Init</td>
<td>AT-TLS was unable to clone the SAF environment for the application. Save the syslogd output and contact IBM.</td>
</tr>
<tr>
<td>5014</td>
<td>Connection Init</td>
<td>AT-TLS was unable to extract ACEE into ENVAR value. Save the syslogd output and contact IBM.</td>
</tr>
<tr>
<td>5015</td>
<td>Connection Init</td>
<td>AT-TLS was unable to process the connection because the connection had already terminated. Review the syslogd output to determine whether the connection was terminated by the remote partner. TTLS trace level 8 (flow) and 16 (event) can be used to gather more information.</td>
</tr>
<tr>
<td>5016</td>
<td>Connection Init</td>
<td>AT-TLS attempted to read ciphertext negotiation data, but an internal error occurred. Save the syslogd output and contact IBM</td>
</tr>
<tr>
<td>5017</td>
<td>Connection Init</td>
<td>The application tried to write data on a secure connection that has been closed by the remote application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Review the local and remote applications to ensure that the TCP sockets are being closed correctly in the application flow.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If further diagnostic information is needed, set the trace level to 255, to trace the data flow and AT-TLS processing.</td>
</tr>
<tr>
<td>5018</td>
<td>Connection Init</td>
<td>An internal error has occurred processing a TTLSGroupAction. Save the syslogd output and contact IBM.</td>
</tr>
<tr>
<td>5019</td>
<td>Connection Init</td>
<td>Task level security could not be created. BPX1TLS failed. Save the syslogd output and contact IBM.</td>
</tr>
<tr>
<td>5020</td>
<td>Connection Init</td>
<td>AT-TLS was unable to load the GSKSSL library. Ensure the SIEALNKE PDSE library is available to the TCPIP started task. See z/OS Cryptographic Services System SSL Programming for more information.</td>
</tr>
<tr>
<td>5023</td>
<td>Connection Init</td>
<td>AT-TLS called initACEE with a nested ENVR object and requested a managed ACEE. This is not supported. If AT-TLS was processing a data connection from the FTP server, ensure the AT-TLS policy has SecondaryMap On coded for the FTP control connection. A separate TTLSRule for the FTP data connection is not supported. Otherwise, save the syslogd output and contact IBM.</td>
</tr>
</tbody>
</table>

Return codes between 6001 and 6999 describe internal AT-TLS errors.

An internal AT-TLS error has occurred. Contact IBM with the error message and syslog information, if available.
**SIOCTTLSCTL ioctl return codes**

The SIOCTTLSCTL ioctl provides the interface for an application to query and control AT-TLS. Table 66 describes the error codes that can be returned on this ioctl, along with the conditions under which each can occur. Also included for each is an indication of whether the query data fields in the ioctl contains valid returned data.

*Table 66. SIOCTTLSCTL error codes*

<table>
<thead>
<tr>
<th>Errno</th>
<th>Errnojr</th>
<th>IOCTL request specified (1)</th>
<th>Condition causing Error</th>
<th>Valid Data? (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAcces</td>
<td>JrConnDeniedPolicy</td>
<td>INIT_CONNECTION, RESET_SESSION, RESET_CIPHER, STOP_CONNECTION</td>
<td>Mapped policy indicates that the application cannot request AT-TLS security for the connection (ApplicationControlled Off)</td>
<td>Yes</td>
</tr>
<tr>
<td>EAlready</td>
<td>JrAlreadyActive</td>
<td>INIT_CONNECTION, STOP_CONNECTION</td>
<td>An INIT_CONNECTION or STOP_CONNECTION request was previously received for the connection</td>
<td>Yes</td>
</tr>
<tr>
<td>EConnReset</td>
<td>JrTTLSSHandshakeFailed</td>
<td>Any</td>
<td>Initial handshake was in progress and socket is a blocking socket. Request blocked for handshake to complete. Handshake failed.</td>
<td>No</td>
</tr>
<tr>
<td>EInProgress</td>
<td>JrOK</td>
<td>INIT_CONNECTION, STOP_CONNECTION</td>
<td>Initial handshake or stop secure connection has been started and socket is a non-blocking socket.</td>
<td>Yes</td>
</tr>
<tr>
<td>EInval</td>
<td>JrInvalidVersion</td>
<td>Any</td>
<td>Bad ioctl version number specified.</td>
<td>No</td>
</tr>
<tr>
<td>EInval</td>
<td>JrSocketCallParmError</td>
<td>Any</td>
<td>Length of input data is not length of ioctl structure.</td>
<td>No</td>
</tr>
<tr>
<td>EInval</td>
<td>JrSocketCallParmError</td>
<td>Not valid</td>
<td>Request type specified is not valid.</td>
<td>No</td>
</tr>
<tr>
<td>EInval</td>
<td>JrSocketCallParmError</td>
<td>RETURN_CERTIFICATE</td>
<td>Certificate buffer pointer = 0 or certificate buffer length = 0.</td>
<td>No</td>
</tr>
<tr>
<td>EInval</td>
<td>JrSocketCallParmError</td>
<td>! RETURN_CERTIFICATE</td>
<td>Certificate buffer pointer != 0 or certificate buffer length != 0 and TTLS_Version is 1.</td>
<td>No</td>
</tr>
<tr>
<td>EMVSErr</td>
<td>JrUnexpectedErr</td>
<td>Any</td>
<td>Policy was not mapped prior to ioctl call and an error was encountered upon policy map during ioctl call.</td>
<td>No</td>
</tr>
<tr>
<td>ENoBufs</td>
<td>JrBuffTooSmall</td>
<td>RETURN_CERTIFICATE</td>
<td>The certificate buffer provided is too small.</td>
<td>Yes (4)</td>
</tr>
</tbody>
</table>
Table 66. SIOCTTLSCTL error codes (continued)

<table>
<thead>
<tr>
<th>Errno</th>
<th>Errnojr</th>
<th>IOCTL request specified (1)</th>
<th>Condition causing Error</th>
<th>Valid Data? (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENoBufs</td>
<td>JrBuffTooSmall</td>
<td>QUERY_ONLY</td>
<td>A TTLS_Version 2 request was issued, but the buffer was too small.</td>
<td>Yes (5)</td>
</tr>
<tr>
<td>ENotConn</td>
<td>JrGetConnError</td>
<td>Any</td>
<td>TCP connection is not yet in established state or has been reset.</td>
<td>No</td>
</tr>
<tr>
<td>EOpNotSupp</td>
<td>JrOptNotSupported</td>
<td>INIT_CONNECTION, RESET_SESSION, RESET_CIPHER, STOP_CONNECTION</td>
<td>Mapped policy indicates that AT-TLS is not enabled for the connection (TTLSEnabled Off).</td>
<td>Yes</td>
</tr>
<tr>
<td>EPerm</td>
<td>JrSocketCallParmError</td>
<td>INIT_CONNECTION with RESET_SESSION or RESET_CIPHER or STOP_CONNECTION, STOP_CONNECTION with RESET_SESSION or RESET_CIPHER, ALLOW_HSTIMEOUT without INIT_CONNECTION</td>
<td>Combination of requests specified is not permitted.</td>
<td>No</td>
</tr>
<tr>
<td>EPipe</td>
<td>JrUnexpectedErr</td>
<td>INIT_CONNECTION, RESET_CIPHER, STOP_CONNECTION</td>
<td>TCP connection is no longer in Established state. Two-way communication is not possible.</td>
<td>Yes</td>
</tr>
<tr>
<td>EProto</td>
<td>JrGetConnErr</td>
<td>RESET_SESSION, RESET_CIPHER</td>
<td>An INIT_CONNECTION request has not been received for the connection.</td>
<td>Yes</td>
</tr>
<tr>
<td>EProto</td>
<td>JrInvalidVersion</td>
<td>RESET_CIPHER, STOP_CONNECTION</td>
<td>Connection is secured using SSL version 2.</td>
<td>Yes</td>
</tr>
<tr>
<td>EProto</td>
<td>JrConnDeniedPolicy</td>
<td>ALLOW_HSTIMEOUT</td>
<td>The TTLS_ALLOW_HSTIMEOUT option was requested but the HandshakeRole is a client or the HandshakeTimeout value is 0.</td>
<td>Yes</td>
</tr>
<tr>
<td>EProtoType</td>
<td>JrSocketTypeNotSupported</td>
<td>Any</td>
<td>Socket is not a TCP socket.</td>
<td>No</td>
</tr>
<tr>
<td>EWWouldBlock</td>
<td>JrOK</td>
<td>Any</td>
<td>SSL handshake is in progress and socket is a non-blocking socket. (3)</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 66. SIOCTTLSCTL error codes (continued)

<table>
<thead>
<tr>
<th>Errno</th>
<th>Errnojr</th>
<th>IOCTL request specified</th>
<th>Condition causing Error</th>
<th>Valid Data?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td>(2)</td>
</tr>
</tbody>
</table>

Notes:
1. The entry **Any** indicates that any valid request or valid combination of request types was specified as follows:
   - **request_type**
     - The listed request_type value was specified alone or in any valid combination of request_type.
   - **request_type, request_type[1], request_type[2]**
     - One of the listed request types was specified alone or in any valid combination of request types.
   - **request_type with request_type**
     - The listed pair of request types was specified together.
   - **! request_type**
     - Any valid combination of request types that does not include the listed request_type was specified.
2. **Yes** indicates that query data fields in the ioctl control block contain valid returned data. **No** indicates that the query data fields are unmodified.
3. For a non-blocking socket, you can wait for the handshake to complete by issuing Select or Poll for Socket Writable.
4. Certificate is not returned because the buffer was not large enough to hold it.
5. Output data is returned for output requests which completely fit in the buffer provided.
Chapter 30. Diagnosing Defense Manager daemon problems

This topic describes how to diagnose Defense Manager daemon (DMD) problems, and contains the following sections:

- "Overview of diagnosing Defense Manager daemon problems"
- "Defense Manager daemon debug information" on page 718
- "TCP/IP services component trace for the Defense Manager daemon” on page 719
- "Enabling CTRACE at Defense Manager daemon startup” on page 720

Overview of diagnosing Defense Manager daemon problems

The DMD oversees the addition, modification and deletion of TCP/IP stack defensive filters. Problems with the DMD may be categorized as follows:

- DMD configuration problems
- DMD internal problems
- DMD problems interacting with an external component such as:
  - ipsec command
  - Secure Access Facility

The DMD provides log output using syslogd and internal trace information using component trace (CTRACE). The log output is sufficient for diagnosing most DMD problems and is the first place to look if you suspect a problem.

Table 67 on page 716 lists common DMD problems.
Table 67. Common defense manager daemon (DMD) problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipsec command cannot be used to update or delete a defensive filter installed in the stack.</td>
<td>The <code>ipsec -d display</code> command lists defensive filters that cannot be updated or deleted. An <code>ipsec -F update</code> command to update a defensive filter fails with message: <code>EZD1500I Defense Manager daemon reported an error - filter is not found</code>. An <code>ipsec -F delete</code> command to delete a defensive filter fails with message: <code>EZD1546I Defensive filter filtername was not found in stack stackname</code>.</td>
<td>Under normal circumstances the DMD's defensive filter files are in sync with the defensive filters in each running stack. However, there are several operational errors that can lead to a mismatch where defensive filters are installed in the stack but the DMD does not have any knowledge of them. The <code>ipsec -F display</code> command will display the defensive filters installed in the stack. However, the filters cannot be referenced by name to update or delete them. One possible cause of this mismatch is if the administrator removes the DmStackConfig statement from the DMD configuration while there are active defensive filters in the stack. If this was done to disable defensive filtering then only a partial disablement was done. No new defensive filters will be added to the stack but the existing ones remain. The user has 2 choices to complete the disablement: The MODIFY FORCE_INACTIVE command can be used to disable defensive filtering. Or the DmStackConfig statement can be added back to the configuration file with Mode Inactive. In both cases, any existing defensive filters will be deleted from the stack. If the DmStackConfig statement was removed inadvertently and defensive filtering should remain enabled for the stack, add the DmStackConfig statement back to the configuration file and refresh the configuration. All of the active defensive filters should be addressable again and new defensive filters can be added. Refreshing the DMD configuration with the wrong configuration file could result in the apparent deletion of a stack. For example, if DMD is started with configuration file <code>/etc/security/dmd.conf</code> which has a DmStackConfig statement for TCPCS2 and TCPCS3, then a refresh is done with configuration file <code>/etc/security/dmd.conf1</code> which only has a DmStackConfig statement for TCPCS2, it will appear that DmStackConfig has been deleted for TCPCS3. The configuration should be refreshed with the correct configuration file. Any active defensive filters should be addressable again. Another possible cause is that the DMD is started with the DefensiveFilterDirectory parameter in the DMD configuration file set to the wrong location. To correct the problem, stop the DMD. The DefensiveFilterDirectory parameter cannot be changed on a MODIFY REFRESH. Correct the DefensiveFilterDirectory value and restart the DMD. All of the active defensive filters should be addressable again.</td>
</tr>
<tr>
<td>The DMD cannot be started. The configuration file cannot be read.</td>
<td>The following message sequence is written to the MVS console: <code>EZD1601I THE DEFENSE MANAGER DAEMON INITIALIZATION SEQUENCE HAS BEGUN</code> <code>EZD1617I AN ERROR OCCURRED WHILE READING THE DEFENSE MANAGER DAEMON CONFIGURATION FILE /etc/security/dmd.conf</code> <code>- RETURN CODE 2</code> <code>EZD1604I THE DEFENSE MANAGER DAEMON SHUTDOWN SEQUENCE HAS BEGUN</code> <code>EZD1605I THE DEFENSE MANAGER DAEMON SHUTDOWN SEQUENCE HAS COMPLETED</code></td>
<td>The configuration file must exist and the DMD must have the right permissions to read it. If the configuration data is not in the default location (that is, <code>/etc/security/dmd.conf</code>), ensure that the environment variable DMD_FILE is set to the name of the MVS data set or z/OS UNIX file that contains the configuration data. If the DMD is defined with a non-zero UID, ensure that the DMD has permission to read the configuration file. This requires that the DMD user has both read access to the configuration file, as well as access to the directory containing the configuration file. Tip: The <code>/var/dm</code> directory must be set up to allow DMD to create, delete, read, and write files to it. You might want to create the configuration file in the <code>/var/dm</code> directory and use the DMD_FILE environment variable to specify the configuration file.</td>
</tr>
</tbody>
</table>
### Table 67. Common defense manager daemon (DMD) problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
</table>
| The DMD cannot be started. The `/var/dm` directory does not exist and cannot be created. | The following message sequence is written to the MVS console:  
**EZD1610I** THE DEFENSE MANAGER DAEMON INITIALIZATION SEQUENCE HAS BEGUN  
**EZD1604I** THE DEFENSE MANAGER DAEMON SHUTDOWN SEQUENCE HAS COMPLETED.  
The following message is written to syslog:  
**EZD1624I** The Defense Manager daemon socket directory `/var/dm` does not exist and cannot be created -  
_errno 111 EDC5111I Permission denied | The directory `/var/dm` must exist or the DMD must have the right permissions to create it. If the DMD needs to create the dm subdirectory, `/var` must exist.  
If the DMD is defined with a non-zero UID, create the `/var/dm` directory before starting the DMD. The directory should be owned by the DMD userid and the DMD should be able to create, delete, read, and write files to the directory. |
| The DMD cannot be started. The directory `/var/dm` exists but the DMD is not able to create a file in the directory. | The following message sequence is written to the MVS console:  
**EZD1610I** THE DEFENSE MANAGER DAEMON INITIALIZATION SEQUENCE HAS BEGUN  
**EZD1732I** AN ERROR OCCURRED WHILE TRYING TO ACCESS THE DMD DIRECTORY `/var/dm`  
_errno 111 EDC5111I Permission denied.  
**EZD1617I** AN ERROR OCCURRED WHILE READING THE DEFENSE MANAGER DAEMON CONFIGURATION FILE `/etc/security/dmd.conf`  
_Return code 3_  
**EZD1604I** THE DEFENSE MANAGER DAEMON SHUTDOWN SEQUENCE HAS BEGUN  
**EZD1605I** THE DEFENSE MANAGER DAEMON SHUTDOWN SEQUENCE HAS COMPLETED. | If the DMD is defined with a non-zero UID, the directory should be owned by the DMD userid and the DMD should be able to create, delete, read, and write files to the directory. |
| The DMD cannot be started. The defensive filter directory does not exist. | The following message sequence is written to the MVS console:  
**EZD1610I** THE DEFENSE MANAGER DAEMON INITIALIZATION SEQUENCE HAS BEGUN  
**EZD1621I** AN ERROR OCCURRED WHILE TRYING TO ACCESS DEFENSIVE FILTER DIRECTORY `/var/dm/filters`  
_errno 129 EDC5129I No such file or directory.  
**EZD1617I** AN ERROR OCCURRED WHILE READING THE DEFENSE MANAGER DAEMON CONFIGURATION FILE `/etc/security/dmd.conf`  
_Return code 3_  
**EZD1604I** THE DEFENSE MANAGER DAEMON SHUTDOWN SEQUENCE HAS BEGUN  
**EZD1605I** THE DEFENSE MANAGER DAEMON SHUTDOWN SEQUENCE HAS COMPLETED. | The defensive filter directory specified in the DMD configuration file on the DefensiveFilterDirectory keyword must be created before starting the DMD. The DMD must be able to create, delete, read, and write files to the directory.  
If the DMD is defined with a non-zero UID, the directory should be owned by the DMD userid and the DMD should be able to create, delete, read, and write files to the directory. |
### Table 67. Common defense manager daemon (DMD) problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Symptom</th>
<th>Cause/response</th>
</tr>
</thead>
<tbody>
<tr>
<td>The DMD cannot be started. Files cannot be written to the defensive filter directory.</td>
<td>The following message sequence is written to the MVS console:</td>
<td>The DMD must be able to create, read, write, and delete files from the defensive filter directory specified in the DMD configuration file on the DefensiveFilterDirectory keyword.</td>
</tr>
<tr>
<td></td>
<td>EZD1601I THE DEFENSE MANAGER DAEMON INITIALIZATION SEQUENCE HAS BEGUN</td>
<td>If the DMD is defined with a non-zero UID, the directory should be owned by the DMD userid and the DMD should be able to create, delete, read, and write files to the directory.</td>
</tr>
<tr>
<td></td>
<td>EZD1611I AN ERROR OCCURRED WHILE TRYING TO ACCESS DEFENSIVE FILTER DIRECTORY /var/dm/filters - ERRNO 111 EDC5111I Permission denied.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EZD1617I AN ERROR OCCURRED WHILE READING THE DEFENSE MANAGER DAEMON CONFIGURATION FILE /etc/security/dmd.conf - RETURN CODE 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EZD1604I THE DEFENSE MANAGER DAEMON SHUTDOWN SEQUENCE HAS BEGUN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EZD1605I THE DEFENSE MANAGER DAEMON SHUTDOWN SEQUENCE HAS COMPLETED</td>
<td></td>
</tr>
<tr>
<td>The DMD starts but the process ID is not written to a file.</td>
<td>The following message is written to the MVS console if the directory portion of the PID file location does not exist:</td>
<td>The directory portion of the PID file location specified by the DMD_PIDFILE environment variable or defaulted to /var/dm must be created before starting the DMD. The DMD must be able to write the PID file to the directory.</td>
</tr>
<tr>
<td></td>
<td>EZD1603I THE DEFENSE MANAGER DAEMON FAILED TO WRITE ITS PROCESS ID 61080992 TO /var/dm/dmd.pid - ERRNO 129 Errno DESCRIPTION EDC5129I No such file or directory.</td>
<td>If the directory does not exist, create it. If the DMD is defined with a non-zero UID, the directory should be owned by the DMD userid and the DMD should be able to create, delete, read, and write files to the directory.</td>
</tr>
<tr>
<td></td>
<td>The following message is written to the MVS console if the DMD does not have permission to write the PID file to the directory:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EZD1603I THE DEFENSE MANAGER DAEMON FAILED TO WRITE ITS PROCESS ID 29 TO /var/dm/dmd.pid - ERRNO 111 ERRNO DESCRIPTION EDC5111I Permission denied.</td>
<td></td>
</tr>
</tbody>
</table>

### Defense Manager daemon debug information

The SyslogLevel parameter in the DMD configuration file controls the level of DMD internal debug information that is sent to syslog. See [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com/support/docview.wss?uid=swg27046777) for more information.

### Abends during DMD processing

Messages and error-related information should be sent to the system console when an abend occurs during DMD processing. A dump of the error is needed unless the symptoms match a known problem. System dumps of the DMD include Language Environment data. The Language Environment IPCS verbexit LEDATA can be used to format this information. See [z/OS Language Environment Debugging Guide](https://www.ibm.com/support/docview.wss?uid=swg27013181) for more information. The following is a sample IPCS verbexit LEDATA command:

```plaintext
verbx ledata 'asid(68) tcb(007E5E88) cedump nthreads(*)'
```

**Tip:** In this example, the DMD asid is 0x68 and the address of the abended DMD TCB is 0x007E5E88.
DMD error codes

Several messages display a return code and reason generated by the DMD. These return codes and reasons are displayed by the ipsec command.

TCP/IP services component trace for the Defense Manager daemon

The DMD uses component trace support to trace internal operations. For short descriptions of other tracing procedures, such as displaying trace status, see Chapter 5, “TCP/IP services traces and IPCS support,” on page 47.

For detailed information, see the following:

- z/OS MVS Diagnosis: Tools and Service Aids for information about component trace procedures.
- z/OS MVS Initialization and Tuning Reference for information about the component trace SYS1.PARMLIB member.
- z/OS MVS System Commands for information about commands.
- z/OS MVS Programming: Authorized Assembler Services Guide for procedures and return codes for component trace macros.

CTRACE options

You can specify component trace options at DMD initialization or after the DMD has initialized.

Table 68 lists the DMD trace options.

Table 68. Defense manager daemon (DMD) trace options

<table>
<thead>
<tr>
<th>Trace event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Select all types of records. Be aware that this option slows performance.</td>
</tr>
<tr>
<td>MINIMUM</td>
<td>Select the DMD’s minimum level of tracing. This level includes the INIT, EXCEPT, and TERM categories.</td>
</tr>
<tr>
<td>INIT</td>
<td>Select the DMD initialization information.</td>
</tr>
<tr>
<td>TERM</td>
<td>Select the DMD termination information.</td>
</tr>
<tr>
<td>EXCEPT</td>
<td>Select the DMD exception information.</td>
</tr>
<tr>
<td>CONFIG</td>
<td>Select the DMD configuration information.</td>
</tr>
<tr>
<td>COMMANDS</td>
<td>Select processing of DMD commands from the console or command line.</td>
</tr>
<tr>
<td>LOGMSGS</td>
<td>Select the DMD syslog messages. These entries can be used to easily correlate system log messages to a specific point in the CTRACE log.</td>
</tr>
<tr>
<td>ROUTING</td>
<td>Select the DMD threading and request dispatching information.</td>
</tr>
<tr>
<td>SERIAL</td>
<td>Select the DMD serialization information.</td>
</tr>
<tr>
<td>EVENT</td>
<td>Select the DMD event information.</td>
</tr>
<tr>
<td>SOCKETS</td>
<td>Select the DMD socket information.</td>
</tr>
<tr>
<td>PERFORM</td>
<td>Select the DMD performance information.</td>
</tr>
<tr>
<td>REQUESTS</td>
<td>Select the DMD request/response information.</td>
</tr>
<tr>
<td>FLOW</td>
<td>Select the DMD code flow information.</td>
</tr>
<tr>
<td>STORAGE</td>
<td>Select the DMD storage information.</td>
</tr>
</tbody>
</table>
### Enabling CTRACE at Defense Manager daemon startup

A default minimum component trace is always started during DMD initialization. Use a parmlib member to customize the parameters that are used to initialize the trace. The default DMD component trace parmlib member is the SYS1.PARMLIB member CTIDMD00. The parmlib member name can be changed using the DMD_CTRACE_MEMBER environment variable.

**Rule:** The DMD reads the DMD_CTRACE_MEMBER environment variable only during initialization. Changes to DMD_CTRACE_MEMBER after server initialization have no affect.

**Restriction:** In addition to specifying the trace options, you can also change the DMD trace buffer size. The buffer size can be changed only at DMD initialization and has a maximum size of 256 MB.

If the CTIDMD00 member or the member that is specified in DMD_CTRACE_MEMBER is not found when starting the DMD, the following message is issued:

```
IEE5381 memberName MEMBER NOT FOUND IN PARMLIB
```

When this occurs, the DMD component trace is started with a buffer size of 1 MB and the MINIMUM tracing option.

---

**Table 68. Defense manager daemon (DMD) trace options (continued)**

<table>
<thead>
<tr>
<th>Trace event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTROL</td>
<td>Select the DMD control information.</td>
</tr>
<tr>
<td>DEBUG</td>
<td>Select the DMD debugging information.</td>
</tr>
<tr>
<td>VERBOSE</td>
<td>Select the DMD verbose debugging information.</td>
</tr>
</tbody>
</table>
TRACEOPTS
/* Optionallly start external writer in this file (use both */
/* WTRSTART and WTR with same wtr_procedure) */
/* WTRSTART(wtr_procedure)
/* -----------------------------------------------
/* ON OR OFF: PICK 1 */
/* -----------------------------------------------
ON
/* OFF */
/* BUFSIZE: A VALUE IN THE RANGE OF 128K TO 256M */
/* CTRACE buffers reside in the Defense Manager daemon's private */
/* storage which is in the region's address space. */
/* -----------------------------------------------
BUFSIZE(1M)
/* WTR(wtr_procedure)
/* -----------------------------------------------
/* OPTIONS: NAMES OF FUNCTIONS TO BE TRACED, OR "ALL"
/* -----------------------------------------------
/* OPTIONS( */
/* 'ALL ' */
/* 'MINIMUM' */
/* 'INIT' */
/* 'TERM ' */
/* 'EXCEPT ' */
/* 'CONFIG ' */
/* 'ROUTING ' */
/* 'COMMANDS' */
/* 'LOGMSG ' */
/* 'SERIAL ' */
/* 'EVENT ' */
/* 'SOCKETS ' */
/* 'REQUESTS' */
/* 'FLOW ' */
/* 'STORAGE ' */
/* 'CONTROL ' */
/* 'VERBOSE ' */
/* 'DEBUG ' */
/* 'PERFORM ' */
/* ) */
/* */
Steps for enabling the CTRACE at Defense Manager daemon startup

1. Edit the CTIDMD00 parmlib member and specify TRACEOPTS ON, the desired buffer size with the BUFSIZE() parameter and the desired CTRACE options. To direct the CTRACE to an external writer, also specify the name of the writer JCL procedure in the WTR() parameter. See Figure 93 on page 721.
2. Start the DMD.

Steps for disabling the CTRACE at Defense Manager daemon startup

1. To disable the CTRACE at DMD startup, edit the CTIDMD00 parmlib member and specify TRACEOPTS OFF.
2. Start the DMD.

Step for enabling the CTRACE after the Defense Manager daemon has started

Perform either of the following steps to enable the CTRACE after the DMD has started.

- Issue the following console commands to enable the CTRACE to an internal buffer:
  
  TRACE CT,ON,COMP=SYSTCPDM,SUB=(dmd_jobname)
  R xx,OPTIONS={option[,...]},END

- Issue the following console commands to enable the CTRACE to an external writer:
  
  TRACE CT,WTRSTART=writer_proc
  TRACE CT,ON,COMP=SYSTCPDM,SUB=(dm_jobname)
  R xx,OPTIONS={option[,...]},WTR=writer_proc,END

Step for disabling the CTRACE after the Defense Manager daemon has started

Perform either of the following steps to disable the CTRACE after the DMD has started.

- Issue the following console commands to disable the CTRACE to an internal buffer:
  
  TRACE CT,OFF,COMP=SYSTCPDM,SUB=(dm_jobname)

- Issue the following console commands to disable a CTRACE to an external writer:
  
  TRACE CT,OFF,COMP=SYSTCPDM,SUB=(dm_jobname)
  TRACE CT,WTRSTOP=writer_proc

Displaying the CTRACE status

To display the CTRACE status, issue the following console command:

D TRACE,COMP=SYSTCPDM,SUB=(dm_jobname)

Enabling CTRACE after Defense Manager daemon initialization

After DMD initialization, you must use the TRACE CT command to change the component trace options. Each time a new Component Trace is initiated, all prior trace options are turned OFF and the new options are put into effect. You can
specify the trace options with or without the parmlib member. See Chapter 5, "TCP/IP services traces and IPCS support," on page 47 for more information.

**Formatting Defense Manager daemon trace records**

You can format component trace records using IPCS panels or a combination of the IPCS panels and the CTRACE command, either from a dump or from external-writer files. See Chapter 5, "TCP/IP services traces and IPCS support," on page 47 for details.

Enter any combination of values as options to filter the CTRACE entries. The options must be entered using the following format: \texttt{TYPE(option[,option]...)}.

You can use any of the options listed in Table 68 on page 719 except ALL and MINIMUM.
Chapter 31. Diagnosing IP security and defensive filter problems

This topic describes how to diagnose IP security problems including problems with defensive filters. It contains the following sections:

- "Overview of diagnosing IP security and defensive filter problems"
- "Steps for diagnosing IP security problems" on page 726
- "Steps for diagnosing defensive filter problems" on page 727
- "Steps for diagnosing the cause for missing IP security or defensive filter syslogd output" on page 729
- "Steps for verifying IP security and defensive filter operation" on page 732
- "Tools for diagnosing IP security and defensive filter problems" on page 744

Overview of diagnosing IP security and defensive filter problems

IPSec configuration files are input to the Policy Agent to establish a TCP/IP stack IP filter policy, Key Exchange policy, and LocalDynVpn policy. These configuration files consist of a number of configuration statements and parameters documented in the z/OS Communications Server: IP Configuration Reference and can be configured manually into a flat file. Optionally, IBM provides a IBM Configuration Assistant for z/OS Communications Server, which provides wizards and a set of reusable objects (at a different level of abstraction than if configured manually). The IBM Configuration Assistant for z/OS Communications Server ultimately produces the Policy Agent configuration files on your behalf.

When diagnosing problems, it might be helpful to understand the relationship of the GUI level objects to the configuration file objects. Table 69 provides a brief mapping of these objects.

Table 69. GUI-level object mapping

<table>
<thead>
<tr>
<th>Policy Agent Object</th>
<th>IBM Configuration Assistant for z/OS Communications Server Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>IpDataOffer</td>
<td>Configured in security levels implementing dynamic tunnels</td>
</tr>
<tr>
<td>IpDynVpnAction</td>
<td>Security level implementing dynamic tunnels</td>
</tr>
<tr>
<td></td>
<td>A numeric suffix is appended to the Security Level name to guarantee uniqueness.</td>
</tr>
<tr>
<td>IpFilterRule</td>
<td>Connectivity rule</td>
</tr>
<tr>
<td></td>
<td>A numeric suffix is appended to the connectivity rule name to guarantee uniqueness.</td>
</tr>
<tr>
<td>IpManVpnAction</td>
<td>Security level implementing manual tunnels</td>
</tr>
<tr>
<td></td>
<td>A numeric suffix is appended to the security level name to guarantee uniqueness.</td>
</tr>
<tr>
<td>IpService</td>
<td>Configured in traffic descriptors</td>
</tr>
<tr>
<td></td>
<td>A numeric suffix is appended to the traffic descriptor name to guarantee uniqueness.</td>
</tr>
</tbody>
</table>
Table 69. GUI-level object mapping (continued)

<table>
<thead>
<tr>
<th>Policy Agent Object</th>
<th>IBM Configuration Assistant for z/OS Communications Server Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>IpTimeCondition</td>
<td>Defined within either Connectivity Rules or Security Levels implementing Manual Tunnels</td>
</tr>
<tr>
<td>KeyExchangeAction</td>
<td>Configured in connectivity rules</td>
</tr>
<tr>
<td></td>
<td>A numeric suffix is appended to the connectivity rule name to guarantee uniqueness.</td>
</tr>
<tr>
<td>KeyExchangeRule</td>
<td>Configured in Connectivity Rules</td>
</tr>
<tr>
<td></td>
<td>A numeric suffix is appended to the Connectivity Rule name to guarantee uniqueness.</td>
</tr>
<tr>
<td>LocalDynVpnRule</td>
<td>Configured in connectivity rules</td>
</tr>
<tr>
<td></td>
<td>Names are user specified.</td>
</tr>
</tbody>
</table>

The Policy Agent installs IP security policy into the stack and the IKE daemon. Specifically, IP filter policy is installed in the stack and Key Exchange policy and LocalDynVpn policy are installed in the IKE daemon. The stack enforces IP filter policy after it has been successfully installed. The IKE daemon enforces Key Exchange policy and LocalDynVpn policy after they have been successfully installed. The Traffic Regulation Management daemon (TRMD) reports IP security events to syslogd on behalf of the stack.

**Defensive filters:** Defensive filters are deny filters that can be added through the `ipsec` command, typically by an external security information and event manager that detects an attack. Defensive filters can only be installed in a TCP/IP stack that has IP security enabled. Defensive filters are given higher priority than IP security filters. That is, IP filter processing will first check a packet against any installed defensive filters for a match, before checking the IP security filters.

Problems can occur in the following areas:
- IP security policy installation
- IP security and defensive filter output to syslogd
- IP security operation
- Adding and managing defensive filters

**Steps for diagnosing IP security problems**

Perform the following steps to diagnose IP security problems.

1. Issue `pasearch -v a` to see all IP security policies that are active in Policy Agent. Refer to z/OS Communications Server: IP System Administrator’s Commands for more information about the `pasearch -v a` command. If you are running multiple stacks, ensure that pasearch is reporting on the stack you are interested in. See Chapter 26, “Diagnosing Policy Agent problems,” on page 653 if you do not see the IP security policies that you expected.

   **Tip:** IP security policies that are active in the Policy Agent might not be active in the stack. Issue `ipsec -f display` and locate the Source field to determine the source of the policy that is active in the stack. If the Source field indicates Stack Policy, then the policy that is active in the Policy Agent corresponds to the policy that is active in the stack.
Tip: Defensive filters are not defined in the policy agent configuration file so defensive filters are not displayed by pasearch.

2. Issue \texttt{ipsec -f display} to see how the stack mapped your IpFilterPolicy statement. Refer to \texttt{z/OS Communications Server: IP System Administrator's Commands} for more information about the \texttt{ipsec -f command}. If you are running multiple stacks, ensure that your resolver configuration correctly identifies the stack you are interested in. Ensure that your IP security policies are correctly defined. Refer to the IP security information in \texttt{z/OS Communications Server: IP Configuration Guide}.

Tip: When the command \texttt{ipsec -f display} is issued with a scope of \texttt{-c current}, any defensive filters installed in the stack will be displayed along with IP security filters.

---

### Steps for diagnosing defensive filter problems

Perform the following step to diagnose defensive filter problems.

1. Issue \texttt{ipsec -F display} to display active defensive filters. Refer to \texttt{z/OS Communications Server: IP System Administrator's Commands} for more information about the \texttt{ipsec -F command}. The defensive filters installed in a stack can be displayed or the global defensive filters can be displayed. If you are running multiple stacks and you want to display defensive filters installed in a specific stack, specify the \texttt{-p stacknameoption} or ensure that your resolver configuration correctly identifies the stack you are interested in. To display global defensive filters, specify the \texttt{-G} option.

Perform the following steps to determine why defensive filters are not being successfully added to a stack.

1. Ensure that IP security is enabled for the stack. Specify \texttt{IPSECURITY} on the \texttt{IPCONFIG} statement in the TCP/IP profile. In addition, specify \texttt{IPSECURITY} on the \texttt{IPCONFIG6} statement in the TCP/IP profile if support is needed for IPv6 defensive filters. See \texttt{z/OS Communications Server: IP Configuration Reference} for more information about the \texttt{IPCONFIG IPSECURITY} and \texttt{IPCONFIG6 IPSECURITY} statements.

2. Ensure that the Defense Manager daemon (DMD) is managing defensive filters for the stack. The TCP/IP stack name must be listed in the DMD configuration file to enable defensive filters for the stack. The mode specified on the \texttt{DmStackConfig statement for the stack must be Active or Simulate}. See \texttt{z/OS Communications Server: IP Configuration Reference} for more information about the DMD configuration file.

3. Ensure that the user has security product authorization to issue the \texttt{ipsec} command to add a defensive filter. See "ipsec command security" in \texttt{z/OS Communications Server: IP System Administrator's Commands} for more information on defining the necessary SERVAUTH profiles.

Perform the following step if administrative access is being denied by a defensive filter.

1. Exclude the administrator's IP address from defensive filter checking. Use the Exclude keyword on the \texttt{DmStackConfig statement in the DMD configuration file.}
Perform the following steps if a stack’s defensive filters are not blocking traffic.

1. Ensure that the filter’s mode is set to Block. The Action field on the `ipsec -F display` report should indicate Defensive Block. If the Action field indicates Defensive Simulate, issue `ipsec -F update` to change the filter’s mode. See z/OS Communications Server: IP System Administrator’s Commands for more information about the `ipsec -F` command.

2. Ensure that the stack’s mode in the DMD configuration file is set to Active. A mode of Simulate will override the individual filter’s setting. It will allow a packet to match a defensive filter, generate a message, and then continue to be processed. The mode must be set to Active to cause the individual filter’s mode setting to be honored. See z/OS Communications Server: IP Configuration Reference for more information about specifying a stack and its mode in the DMD configuration file. The MODIFY DISPLAY command can be issued for the DMD to display the active configuration settings. See z/OS Communications Server: IP System Administrator’s Commands for more information about the MODIFY command.

Perform the following step if a defensive filter is discarding traffic that should be permitted.

1. Delete the defensive filter if it is causing legitimate traffic to be discarded. Issue the `ipsec -F delete` command to delete the defensive filter from the stack. See z/OS Communications Server: IP System Administrator’s Commands for more information about the `ipsec -F` command.

Perform the following steps to disable defensive filtering for a stack.

1. Update the DMD configuration file to disable defensive filters for a stack. Specify a mode of Inactive for the stack on the DmStackConfig statement. See z/OS Communications Server: IP Configuration Reference for more information about the DMD configuration file.

2. Issue the MODIFY REFRESH command for the DMD. See z/OS Communications Server: IP System Administrator’s Commands for more information about the MODIFY command.

3. **Tip:** If you are unable to update your DMD configuration file, the MODIFY FORCE_INACTIVE command can be issued for the DMD to disable defensive filtering for the stack. A later MODIFY REFRESH will use the DMD configuration file. If you want defensive filtering to remain disabled, you should update the DMD configuration file as soon as possible.

4. **Tip:** Removing the DmStackConfig statement from the DMD configuration file will not delete existing defensive filters from the stack. If you removed the DmStackConfig statement, the defensive filters will remain in the stack until expiration. To remove the defensive filters from the stack immediately, add the DmStackConfig statement back to the DMD configuration file with a mode of Inactive or issue the MODIFY FORCE_INACTIVE command for the stack.

Perform one of the following actions to remove all defensive filters from a stack.

- Issue `ipsec -F delete -N all -p stackname` to delete all existing defensive filters from the stack. This will also delete them from the DMD’s persistent storage so the filters will not be reinstalled if the stack were to be stopped and restarted. Defensive filtering remains enabled for the stack and new filters can be added to
the stack. See z/OS Communications Server: IP System Administrator’s Commands for more information about the `ipsec -F` command.

- Disable defensive filtering for a stack as described earlier. This will remove all existing defensive filters from the stack and the DMD’s persistant storage. It will also prevent new defensive filters from being installed in the stack.

- **Tip:** The following actions will not remove defensive filters from the stack.
  - Stopping and restarting the stack. The DMD will reinstall defensive filters when the stack is restarted.
  - Stopping DMD. Existing defensive filters remain installed in the stack until expiration.

---

**Steps for diagnosing the cause for missing IP security or defensive filter syslogd output**

Perform the following steps to determine the cause for missing IP security or defensive filter syslogd output.

1. Ensure that Policy Agent is running on this system if IP security policy is defined.

2. Ensure that TRMD is running for this stack on this system. Consider using TCPIP PROFILE Autolog for TRMD. See “Diagnosing TRMD problems” on page 699 for more information.

3. Ensure that syslogd is running on this system.

4. Ensure that syslogd is configured for IP security and defensive filter output. TRMD always writes IP security and defensive filter log records to the syslog local4 facility.

<table>
<thead>
<tr>
<th>Table 70. IPSec messages logged by TRMD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Message</strong></td>
</tr>
<tr>
<td>EZD0827I Remote port translated</td>
</tr>
<tr>
<td>EZD0811I Decapsulation failed (reason codes 8 and 9)</td>
</tr>
<tr>
<td>All other IPSec messages logged by TRMD.</td>
</tr>
</tbody>
</table>

**Notes:**

- a. If IP security policy is configured to log permits and denies, TRMD sends those messages to syslogd using facility local4.
- b. If IKED is configured for logging, IKED messages are sent to syslogd using facility local4 and varied priorities.

<table>
<thead>
<tr>
<th>Table 71. Defensive filter messages logged by TRMD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Message</strong></td>
</tr>
<tr>
<td>Defensive filter messages logged by TRMD</td>
</tr>
</tbody>
</table>

**Notes:**

- a. If a defensive filter indicates that a filter match should be logged, TRMD sends those messages to syslogd using facility local4.
b. If the DMD is configured for logging, DMD messages are sent to syslogd using facility local4 and varied priorities.

Tips:

• If TRMD is logging Intrusion Detection Services (IDS) messages, as well as IP Security (IPSec) messages and defensive filter messages, consider using the facility to separate the IDS messages from the IPSec and defensive filter messages. IDS messages are written to the daemon facility. IPSec and defensive filter messages are written to the Local4 facility.

• If running multiple TRMDs, consider using the syslogd -u option when starting syslogd. The -u option causes the job name of the application writing the syslogd record to be included in the syslogd record.

• If running multiple TRMDs, consider using the trmd jobname prefix to separate IPSec output by stack.

Guidelines:

• Ensure that syslogd is configured to write TRMD and IKED messages for IP security.

• Ensure that syslogd is configured to write TRMD and DMD messages for defensive filters.

• For example, the following lines could be added to the syslogd configuration file to organize TRMD, IKED, and DMD messages:

```bash
*.local4.* /tmp/logs/filter.log
*.IKE+.local4.* /tmp/logs/IKED.log
*.DM+.local4.* /tmp/logs/DMD.log
*.trmd+.local4.* /tmp/logs/trmdfilt.log
*.trmd+ daemon.* /tmp/logs/ids.log
```

In the example, IKED, DMD, and TRMD IP security and defensive filter messages are all written to the log file /tmp/logs/filter.log. IKED messages are also written to the log file /tmp/logs/IKED.log. DMD messages are also written to the log file /tmp/logs/DMD.log. IP security and defensive filter TRMD messages are also written to the log file /tmp/logs/trmdfilt.log. If TRMD is logging IDS messages, those messages are written to /tmp/logs/ids.log.

• Ensure that the log files exist or syslogd is configured to create them using the -c option.

• Ensure that the log files are writable.

• Ensure that there is adequate space on the file system for writing to the log files.

Perform the following steps to reduce the amount of syslogd output for IP security and defensive filters.

a. Ensure that the logging levels for the IKE daemon are set appropriately in the IKE daemon configuration file.

   – IkeSyslogLevel - During day-to-day operation, this value should be set no higher than the default of 1. A higher value should be used for temporary diagnostic purposes only. IkeSyslogLevel can also be set to 0 to disable IKE syslog messages entirely.

   – PagantSyslogLevel - During day-to-day operation, this value should be set to the default of 0. A higher value should be used for temporary diagnostic purposes only.

b. Ensure that the logging levels for the DMD are set appropriately in the DMD configuration file.

   – SyslogLevel - During day-to-day operation, this value should be set no higher than 7. A higher value should be used for temporary diagnostic
purposes only. SyslogLevel can also be set to 1 for minimum logging or 0 to disable DMD syslog messages entirely.

c. Ensure that filter logging controls are set appropriately for IP security filters.

- Filter logging generates a message each time an inbound or outbound packet matches the filter. Exhaustive logging of IP traffic can have a negative effect on performance. Filter logging can be controlled at the individual rule level, including the ability to specify whether to log permitted traffic, denied traffic, or both.

- To disable filter logging for profile filter rules:
  - To disable logging for a configured filter rule, set NOLOG on the IPSECRULE or IPSEC6RULE statement.
  - To disable logging for the implicit filter rules that deny all traffic not permitted by a configured rule, set NOLOGIMPLICIT on the IPSEC statement.
  - To disable filter logging for all profile filter rules, set LOGDISABLE on the IPSEC statement.

- To disable filter logging for policy filter rules configured using the Policy Agent:
  - To disable logging for a configured filter rule, set IpFilterLogging No on the IpGenericFilterAction statement.
  - To disable logging for the implicit filter rules that deny all traffic that does not match a configured rule, set IpFilterLogImplicit No on the IpFilterPolicy statement.

- To disable filter logging for policy filter rules configured with the IBM Configuration Assistant for z/OS Communications Server:
  - To disable logging for a configured filter rule, set filter logging to No for the Connectivity Rule.
  - To disable logging for the implicit filter rules that deny all traffic that does not match a configured rule, select Do NOT log implicit deny events on the IPSec: Stack Level Settings panel.
  - To disable filter logging for all policy filter rules, select Disable all filter logging on the IPSec: Stack Level Settings panel.

- The following messages are controlled by the configured filter logging settings described above:
  - EZD0814I Packet permitted
  - EZD0815I Packet denied by policy
  - EZD0821I Packet denied, no tunnel
  - EZD0822I Packet denied, tunnel inactive
  - EZD0832I Packet denied by NAT Traversal Processing
  - EZD0833I Packet denied, tunnel mismatch

d. Ensure that filter logging controls are set appropriately for defensive filters.

- Filter logging generates a message each time an inbound or outbound packet matches the defensive filter. Exhaustive logging of IP traffic can have a negative effect on performance. Filter logging can be controlled at the individual defensive filter rule level.
To disable filter logging for a defensive filter rule, use the `ipsec -F update` command with `log no` specified.

The following messages are controlled by the defensive filter’s log setting:
- EZD1721I Packet denied by defensive filter
- EZD1722I Packet would have been denied by defensive filter

Ensure that IP security and defensive filter messages being logged by the TRMD daemon are being handled appropriately.

The TCP/IP stack invokes the TRMD daemon to log IP security and defensive filter messages to syslog. The filter logging messages described above are logged by the TRMD daemon. TRMD also logs messages that are not associated with a specific filter. For example, when a tunnel is successfully negotiated, TRMD logs message "EZD0818I Tunnel added". Also, when an IP security policy update is processed, TRMD logs message "EZD0816I IPSec Policy updated". When a defensive filter is added to the stack, TRMD logs message "EZD1723I Defensive filter added".

There is no explicit configuration option to turn off logging for TRMD messages that are not associated with a specific filter. However, the syslog configuration file can be updated to exclude some or all TRMD messages. See Table 70 on page 729 for information on the syslog priority used to log TRMD messages. See Table 71 on page 729 for information on the syslog priority used to log defensive filter TRMD messages.

Include the following line in your syslog configuration file to exclude IP security TRMD messages logged with a priority of debug. IP security and defensive filter TRMD messages with a priority of info or higher would be written to /tmp/trmdlog. Messages with a priority of debug would not be written to the file.

```
*.TRMD*.local4.info /tmp/trmdlog
```

Include the following line in your syslog configuration to exclude all IP security and defensive filter TRMD messages.

```
*.TRMD*.local4.none /tmp/trmdlog
```

All messages with job name TRMD* would be selected. Then all TRMD messages using facility local4 would be excluded. In effect this excludes all IP security and defensive filter TRMD messages from being written to /tmp/trmdlog.

---

**Steps for verifying IP security and defensive filter operation**

Figure 94 on page 733 shows the decisions involved for IP security operation.
Before you begin: Identify the characteristics of the IP traffic for which IP security operation is to be verified. The characteristics of IP traffic that are subject to IP security control are described by the IpFilterRule or IPSECRULE (for IPv4) or...
IPSEC6RULE (for IPv6) statement. Refer to z/OS Communications Server: IP Configuration Reference for more information about the IpFilterRule, IPSECRULE and IPSEC6RULE statements.

Perform the following steps to verify IP security and defensive filter operation.

1. Use the Netstat CONFIG/-f command to determine whether the TCP/IP stack is configured for IP security for IPv4, IPv6, or both. For information about the Netstat command, refer to z/OS Communications Server: IP System Administrator's Commands.

   Do one of the following:
   - If the stack is not configured for IP security for the IP protocol that you want, proceed to step 2.
   - If the stack is configured for IP security for the IP protocol that you want, proceed to step 3.

2. If you want IP security enabled for IPv4, configure the stack for IPv4 IP security using the IPCONFIG IPSECURITY statement in the TCP/IP profile. If you want IP security enabled for IPv6, configure the stack for IPv6 IP security using the IPCONFIG6 IPSECURITY statement in the TCP/IP profile. Refer to z/OS Communications Server: IP Configuration Reference for more information about the IPCONFIG IPSECURITY and IPCONFIG6 IPSECURITY statements. Refer to z/OS Communications Server: IP Configuration Guide for general information about IP security concepts, including IP filtering.

3. Use the MODIFY command to display the configuration values for the Defense Manager daemon (DMD). For more information about the MODIFY command, see z/OS Communications Server: IP System Administrator’s Commands.

   If you want defensive filtering enabled, do one of the following:
   - If the stack name is not listed in the DMD configuration, proceed to step 4.
   - If the stack name is listed in the DMD configuration but does not have the desired mode, proceed to step 5.
   - If the stack name is listed in the DMD configuration with the desired mode, proceed to step 6.

   Otherwise, if you do not want defensive filtering, proceed to step 6.

4. Update the DMD configuration file to include a DmStackConfig statement for the stack for which you want defensive filtering. Specify the defensive filtering mode, Active or Simulate, on the DmStackConfig statement with the stack name. See z/OS Communications Server: IP Configuration Reference for more information on the DMD configuration file. Proceed to step 6.

5. Update the DMD configuration file to specify the defensive filtering mode, Active or Simulate, on the DmStackConfig statement. See z/OS Communications Server: IP Configuration Reference for more information on the DMD configuration file.

6. Use the ipsec -t command to determine which IP filter applies to the identified IP packet. At the top of the ipsec -t command output, note whether Source indicates Stack Profile or Stack Policy.

Limited IP filter controls can be configured using the IPSECRULE statement (for IPv4) and the IPSEC6RULE statement (for IPv6) in the TCP/IP profile. Full
IP security capability, including manual and dynamic IPSec protection, requires use of the Policy Agent for IP security policy configuration.

Locate the Type field in the `ipsec -t` command output to determine the type of filter. If the Type field indicates Defensive, then the filter is a defensive filter. Defensive filters are not configured but are added to the stack by the `ipsec` command. Typically, this is done by an external security information and event manager that detects an attack. However, the `ipsec` command can be issued manually by a user with the appropriate authority to add a defensive filter.

**Tip:** The `ipsec -t` command can return multiple filter rules because the actual packet filtering compares more attributes than might be supplied as input on the `ipsec -t` command. To minimize this effect, supply as much information as possible on the `ipsec -t` command.

If the returned filter rules include a defensive filter, take the following actions:

- Locate the exclusion list at the top of the `ipsec -t` command output and determine if there are any IP addresses listed. Traffic from IP addresses in the exclusion list will bypass defensive filters.
- Locate the Action field in the `ipsec -t` command output to determine the mode of the defensive filter. If the Action field indicates Defensive Block, the filter is discarding traffic. If the Action field indicates Defensive Simulate, only filter logging is done, packets continue to be processed.
- If the defensive filter rule is blocking traffic that should be allowed, determine the user that added the filter by inspecting the syslog messages. Locate the "EZD1723I Defensive filter added" defensive filter message that corresponds to this defensive filter. The userid of the user that added the filter is included in the message.

If none of the filters that are returned by the `ipsec -t` command include the desired action for the identified IP packet, then correct the IP filter configuration. Refer to the `z/OS Communications Server: IP Configuration Guide` for general information about configuring IP filters.

7. Locate the Type field in the `ipsec -t` command output to determine whether IPSec protection is configured for the identified IP packet. If the Type field indicates Generic or Defensive, then IPSec protection is not configured for the identified IP packet. See "Steps for verifying IP security policy or defensive filter enforcement" on page 740 to verify that the configured policy is enforced for the IP traffic characterized by the identified IP packet.

8. Locate the Type field in the `ipsec -t` command output to determine whether manual or dynamic IPSec protection is configured for the identified IP packet. If the Type field indicates Manual, then see "Steps for verifying manual IPSec protection." If the Type field indicates Dynamic or Dynamic Anchor, then see "Steps for verifying dynamic IPSec protection" on page 737.

**Steps for verifying manual IPSec protection**

Figure 95 on page 736 shows the decisions involved for verifying manual IPSec protection.
Before you begin: Complete the steps in “Steps for verifying IP security and defensive filter operation” on page 732 in order to identify the name of an IpFilterRule for which manual IPSec protection is to be verified.

Perform the following steps to verify manual IPSec protection.

1. Verify that manual filters that correspond to the identified IpFilterRule are installed in the stack by using the `ipsec -f display -n` command. Two filters of type Manual (1 inbound and 1 outbound) are installed in the stack for an IpFilterRule that is configured with IpManVpnAction. If the manual filter rules are not installed in the stack, then correct the IP filter policy. Note that an IpFilterRule might be inactive (not installed) in the stack due to an IpTimeCondition. For information about the `ipsec` command, refer to z/OS Communications Server: IP System Administrator’s Commands. Refer to z/OS.
2. Obtain the IpManVpnAction name by locating the VpnActionName field in the `ipsec -f` command output. This is the name of the IpManVpnAction policy configuration statement. Obtain the manual tunnel ID by locating the TunnelID field in the `ipsec -f` display command output. The Tunnel ID for a manual tunnel has a value of M, followed by a positive integer.

3. Verify that the manual tunnel is active.
   Use the `ipsec -m display -a` command, supplying the manual tunnel ID. Locate the State field in the `ipsec -m` command output and confirm that it indicates Active. If the manual tunnel is not active, then activate the tunnel using the `ipsec -m activate` command. You might consider updating the IpManVpnAction policy configuration statement to specify `Active yes`, if it is not already specified. A setting of `Active yes` causes the manual tunnel state to be set to active when the manual tunnel is installed in the stack, without the additional step of issuing `ipsec -m activate`.
   If you are using the IBM Configuration Assistant for z/OS Communications Server to configure, you can choose to automatically activate manual tunnels within each Connectivity Rule.

4. Contact the remote security endpoint’s network administrator to ensure that the manual tunnel has been activated remotely. In order for traffic to flow through a manual tunnel the remote security endpoint must also activate the manual tunnel.

5. Verify that IpManVpnAction is enforced. Refer to “Steps for verifying IP security policy or defensive filter enforcement” on page 740.

**Steps for verifying dynamic IPSec protection**

Figure 96 on page 738 shows the decisions involved for verifying dynamic IPSec protection.
Before you begin: Complete the steps in “Steps for verifying IP security and defensive filter operation” on page 732 in order to identify the name of an IpFilterRule for which dynamic IPSec protection is to be verified.

Perform the following steps to verify dynamic IPSec protection.

1. Verify that the IKE daemon is active. See “Steps for verifying server operation” on page 31.
   Tip: The IKE daemon binds to UDP ports 500 and 4500.

2. Use the `ipsec -f display -n` command to verify that dynamic anchor filters that corresponds to the identified IpFilterRule are installed in the stack. Two filters of type Dynamic Anchor (1 inbound and 1 outbound) are installed in the stack for an IpFilterRule that is configured with an IpDynVpnAction. If the dynamic anchor filter rules are not installed in the stack, then correct the IP filter policy. Note that an IpFilterRule might be inactive (not installed) in the stack due to
For information about the `ipsec` command, refer to [z/OS Communications Server: IP System Administrator's Commands](https://publib.boulder.ibm.com/infocenter/com Unterschiedliche Standorte/). Refer to [z/OS Communications Server: IP Configuration Reference](https://publib.boulder.ibm.com/infocenter/com Unterschiedliche Standorte/) for more information about the `IpDynVpnAction` and `IpTimeCondition` statements. If IP filter rules are not installed, also check the following:

- Verify that policy agent is active.
- If policy agent is active, verify that the following messages appeared after IKED was started:
  
  `EZD1058I IKE STATUS FOR STACK stackname IS UP`
  
  `EZD1068I IKE POLICY UPDATED FOR STACK stackname`

  If these messages did not appear, check the Policy Agent log for errors.

3. Use the `ipsec -f display -n` command to verify that the dynamic filters are installed in the stack. When the IKE daemon completes a dynamic tunnel negotiation, it installs two dynamic filters to more specifically control the IP traffic that can be permitted through the dynamic tunnel.

   The dynamic filters are identified with a Type field of Dynamic in the `ipsec` command output.

   Do one of the following:
   - If no dynamic filters are installed in the stack with the identified `IpFilterRule` name, then proceed to step 5.
   - If the dynamic filters are installed in the stack, then proceed to step 4.

4. Verify that the dynamic tunnel that corresponds to the dynamic filters is active.

   The IKE daemon installs a dynamic tunnel and corresponding inbound and outbound dynamic filters into the stack.

   Follow these steps to perform verification:
   a. Locate the dynamic tunnel ID in the `TunnelID` field of the `ipsec -f` command output.

      **Tip:** Be sure to look for the `TunnelID` identified on the filter rule with type Dynamic, rather than the filter rule with type Dynamic Anchor.
   b. Use the `ipsec -y display -a` command, supplying the dynamic tunnel ID.
   c. Locate the `State` field in the `ipsec -y` command output and confirm that it indicates Active. If the dynamic tunnel is not active, then check the IKE `svslogd` output for errors. Otherwise, see “Steps for verifying IP security policy or defensive filter enforcement” on page 740.

5. If no dynamic filters have been installed in the stack, then the dynamic tunnel activation might not have been started.

   Consider whether or not you need to take an action to activate the tunnel.
   - If you intend to manually start the tunnel, then you must issue the `ipsec -y activate` command. If you intend for the tunnel to be automatically activated, you must configure your `LocalDynVpnPolicy` to include a `LocalDynVpnRule` with `AutoActivate` specified.
   - If you intend for the tunnel to be activated on-demand by outbound traffic, then you must configure `AllowOnDemand` Yes on either your `IpFilterPolicy`
or on an IpLocalStartAction associated with the IpFilterRule identified in step 2 on page 738 and you must also set the outbound traffic flow to trigger the activation.

- If the tunnel is intended to be activated by the remote security endpoint, then you must configure your KeyExchangePolicy properly, and the remote security endpoint must initiate the tunnel negotiation. If you know that you have not yet taken a required action to activate the tunnel, do so now. Otherwise, proceed to the next step.

Refer to z/OS Communications Server: IP Configuration Guide for more information about activating dynamic tunnels.

6. Use the `ipsec -y display -b` command to display all dynamic tunnels known to the IKE daemon. In the `ipsec` command output, search for a dynamic tunnel with an IpFilterRule name that matches the identified IpFilterRule name. If there is no such dynamic tunnel, proceed to step 8. Otherwise, proceed to step 7.

7. If the state of the dynamic tunnel that was identified in step 6 is not DONE, then see “Interpreting IKE daemon phase 2 SA states” on page 941. Otherwise, check the syslogd output for errors.

8. Use the `ipsec -k display` command to see whether there is an applicable IKE tunnel negotiation in progress. If not, check the log for errors. Otherwise, proceed to step 9.

9. If the IKE tunnel state is not DONE, then note the role (initiator or responder) of the IKE tunnel and see “Interpreting IKE daemon phase 1 SA states” on page 936. Otherwise, check the syslogd output for errors.

Steps for verifying IP security policy or defensive filter enforcement

Figure 97 on page 741 shows the decisions involved for verifying IP security policy enforcement.
Before you begin: Complete the steps in "Steps for verifying IP security and defensive filter operation" on page 732 in order to identify the name of an IpFilterRule or IPSECRULE or IPSEC6RULE for which IP security policy enforcement is to be verified or the name of a defensive filter.
Perform the following steps to verify IP security policy enforcement.

1. Start TRMD for the stack if it is not already active. The Traffic Regulation Management Daemon (TRMD) is required to log IP filter permits and denies. Refer to [z/OS Communications Server: IP Configuration Reference](#) for information about starting TRMD.

2. Display the identified filter rule using the `ipsec -f display -n` command if it is an IP security filter. Display the identified filter rule using the `ipsec -f display -N` command if it is a defensive filter. Use the instructions in the following lists to temporarily activate logging for the filter if it is not already active. See [z/OS Communications Server: IP Configuration Reference](#) for information about the `IpFilterPolicy`, `IpFilterRule`, `IpGenericFilterAction`, IPSEC, IPSECRULE, and IPSEC6RULE statements.
   - If the displayed filter Type field is Defensive, do the following:
     - If the displayed filter Logging field is not ALL, update the defensive filter’s log setting with the `ipsec -F update` command. See [z/OS Communications Server: IP System Administrator’s Commands](#) for information on the ipsec command.
   - If the `ipsec` command header output indicates Stack Policy, do the following:
     - If the `ipsec` command header output indicates Logging NO, temporarily specify FilterLogging on the IpFilterPolicy statement. If you are using the IBM Configuration Assistant for z/OS Communications Server to configure, select Enable filter logging on the IPSec: Stack Level Settings panel.
     - If the displayed filter Logging field does not indicate ALL, specify IpFilterLogging Yes on the IpGenericFilterAction referenced by the IpFilterRule. If you are using the IBM Configuration Assistant for z/OS Communications Server to configure, set filter logging to Yes in the each Connectivity Rule.
     - Use the MODIFY command with the Policy Agent to activate your changes, if any. Refer to [z/OS Communications Server: IP System Administrator’s Commands](#) for more detailed information on the MODIFY command.
   - If the `ipsec` command header output indicates Stack Profile, do the following:
     - If the `ipsec` command header output indicates Logging NO, specify LOGENABLE on the IPSEC statement.
     - If the displayed filter Logging field does not indicate ALL, specify LOG on the IPSECRULE or IPSEC6RULE statement.
     - Use the VARY TCPIP,OBEYFILE command to activate your changes, if any.

3. After IP filter logging is active, check the syslog to determine whether the IP traffic that is characterized by the filter rule is being permitted or denied. Message EZD0814I is issued when an IP packet is permitted. Message EZD0815I, EZD0821I, EZD0832I, EZD0822I, EZD0833I, or EZD1721I is issued when an IP packet is denied. If the traffic is denied, proceed to step 7. Otherwise, proceed to step 4.
4. If the IP traffic is being permitted, but that is not desired, correct the filter configuration. Refer to "z/OS Communications Server: IP Configuration Guide" for information about configuring IP filtering.

5. Determine whether the IP traffic is subject to IPSec protection by locating the vpnaction field in the EZD0814I message. If the vpnaction field is not N/A then the IP traffic is subject to IPSec protection. If IPSec protection is not applied, then proceed to step 8. Otherwise, proceed to step 6.

6. Determine the properties of the IPSec tunnel by first locating the tunnelID field in the EZD0814I message. Apply the following criteria to evaluate the tunnelID:
   - If the first character of the tunnelID is M, use the `ipsec -m display -a` command to display the corresponding manual tunnel. If the displayed manual tunnel does not have the desired characteristics, correct the IpManVpnAction statement. Refer to "z/OS Communications Server: IP Configuration Reference" for information about the IpManVpnAction statement.
     If you are using the IBM Configuration Assistant for z/OS Communications Server to configure, the IpManVpnAction corresponds to a Security Level implementing Manual Tunnels. If Security Level does not contain the desired characteristics, correct the Security Level. Refer to the Configuration Assistant online help for additional information.
   - If the first character of the tunnelID is Y, use the `ipsec -y display -a` command to display the corresponding dynamic tunnel. If the displayed dynamic tunnel does not have the desired characteristics, correct the IpDynVpnAction statement. Refer to "z/OS Communications Server: IP Configuration Reference" for information about the IpDynVpnAction statement.
     If you are using the IBM Configuration Assistant for z/OS Communications Server to configure, the IpDynVpnAction corresponds to a Security Level implementing dynamic tunnels. If Security Level does not contain the desired characteristics, correct the Security Level. Refer to the Configuration Assistant online help for additional information.

7. Data traffic cannot be initiated to the remote data endpoint in certain cases when NAT traversal support is being used. Message EZD0832I is issued when an attempt is made to initiate data traffic if either of these conditions is true:
   - The remote security endpoint is acting as a security gateway and a NAT was detected between the local security endpoint and the remote security endpoint.
   - The remote security endpoint is behind a NAT device performing port translation
     If not the "cannot initiate case" message, proceed to 9.

8. If the IP traffic is not being protected with IPSec, but you want IPSec protection, correct the filter configuration. Refer to "z/OS Communications Server: IP Configuration Guide" for information about configuring IP filtering.

9. If the IP traffic is being denied, determine what type of filter is denying the traffic.
If the IP traffic is being denied by a defensive filter, message EZD1721I is issued.

- Determine the user that added the defensive filter by locating the "EZD1723I Defensive filter added" message that corresponds to the deny message. The userid of the user that added the filter is included in message EZD1723I.
- Delete the defensive filter if it is denying traffic that should be permitted. Use the `ipsec -F delete` command to delete a defensive filter. See z/OS Communications Server: IP System Administrator’s Commands for information on the ipsec command.

If the IP traffic is being denied by an IP security filter, correct the filter configuration to change this situation. See z/OS Communications Server: IP Configuration Guide for information about configuring IP filtering.

---

### Steps for verifying IPSec processing on zIIP

If attempting to use the zIIP IPSECURITY support (to direct IPSec AH|ESP protocol processing to zIIP), issue the Netstat STATS/-S command while the IPSec workload is running. The inbound and outbound 'Packets Handled by zIIP' counters will be rising if IPSec workload is in fact being processed on zIIP(s). If these counters are not rising while IPSec traffic is flowing, verify (a) GLOBALCONFIG ZIIP IPSECURITY parameters are specified in the TCPIP profile (use Netstat Config/-f to verify); and (b) zIIP(s) are configured to the z/OS image (use MVS D M=CPU command to verify).

### Determining the Workload Manager service class associated with IPSec workload being processed on zIIP

To verify that the new independent enclave is being used with an appropriate WLM service class issue the SDSF ENC command or view the RMF Workload Activity report. For more information regarding the SDSF function of viewing enclaves, see z/OS SDSF Operation and Customization. For additional information regarding the RMF Workload Activity report, see z/OS RMF Report Analysis.

---

### Tools for diagnosing IP security and defensive filter problems

This section describes tools used to diagnose IP security and defensive filter problems.

### Using the `ipsec` command

You can use the `ipsec` command to display information about:

- IP filter rules
- Security associations
- Port translation
- SECCLASS definitions
- Matching IP filter rules for a specified traffic pattern
- Network security information of an IKE daemon’s active NSS IPSec clients
- NSS IPSec clients connected to NSS servers

By default, `ipsec` commands are directed to the local system. Optionally, `ipsec` commands may be directed to remote systems (NSS IPSec clients) using the `ipsec -z` option.
**Restriction:** Management of defensive filters (**ipsec -F**) is only provided through the local ipsec command. Remote management using a NSS server is not supported.

**ipsec -f display**
The **ipsec -f** display command displays information about the current set of filter rules in use by a stack. The current set of filter rules will include any installed defensive filters.

You can use the options listed in Table 72 to define the display.

**Table 72. ipsec -f display command options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>-p</td>
<td>Directs the command to a stack other than the local default stack.</td>
</tr>
<tr>
<td>-z</td>
<td>Directs the command to a NSS IPSec client.</td>
</tr>
<tr>
<td>-c profile</td>
<td>Displays information about the set of filter rules defined on the IPSEC statement in the TCP/IP profile.</td>
</tr>
<tr>
<td>-c policy</td>
<td>Display information about the set of filter rules defined in the Policy Agent IPSec Configuration file.</td>
</tr>
</tbody>
</table>

Several different types of filter rules exist. The **ipsec -F** command applies only to defensive filters.

Filter rules that are disallowed due to time conditions do not appear in the output of **ipsec -f** display command. The **pasearch** command must be used to obtain information about such filter rules. When working with a NSS client, the **pasearch** command needs to be issued on the system where the client is executing. Use the **ipsec -x** command to determine where the NSS IPSec client is executing.

Several different types of filter rules exist. By default, the **ipsec -f** display output includes information about generic, defensive, dynamic anchor, dynamic, NATT anchor, and NATT dynamic filter rules. You can use the -h option to display information about filter rules of type NRF. NAT resolution filter (NRF) rules are present when the remote security endpoint is behind a NAT. Refer to **z/OS Communications Server: IP Configuration Guide** for an explanation of filter types.

**ipsec -F display**
The **ipsec -F display** command displays information about defensive filters.

You can use the options listed in Table 73 to define the display.

**Table 73. ipsec -F display command options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>-P</td>
<td>Directs the command to a stack other than the local default stack.</td>
</tr>
<tr>
<td>-G</td>
<td>Directs the command to the Defense Manager daemon (DMD) to display global defensive filters.</td>
</tr>
</tbody>
</table>

Several different types of filter rules exist. The **ipsec -F display** output only includes defensive filters.
ipsec -F update and ipsec -F delete

The `ipsec -F update` command can be used to update a defensive filter. The `ipsec -F delete` command can be used to delete a defensive filter.

You can use the options listed in Table 74 to define the update and delete values.

Table 74. ipsec -F update or delete command options

<table>
<thead>
<tr>
<th>Option</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>-p</td>
<td>Directs the command to a stack other than the local default stack.</td>
</tr>
<tr>
<td>-G</td>
<td>Directs the command to the Defense Manager daemon (DMD) to manage global defensive filters.</td>
</tr>
<tr>
<td>-N</td>
<td>Provides the name of the defensive filter to be updated or deleted.</td>
</tr>
</tbody>
</table>

ipsec -m display

The `ipsec -m display` command displays information about manual tunnels installed in the stack. Use the -p option to direct the command to a stack other than the default stack or the -z option to direct the command to a NSS IPSec client. Manual tunnels can be either active or inactive. A manual tunnel must be active before traffic matching a filter rule that uses the manual tunnel can be permitted.

Manual tunnels that are not allowed to be used due to time conditions do not appear in the output of `ipsec -m display` command. Use the `pasearch` command to obtain information about such manual tunnels.

ipsec -k display

The `ipsec -k display` command displays information about IKE tunnels for the default stack. This information is obtained from the IKE daemon. Use the -p option to direct the IKE daemon to return information about a different stack or the -z option to direct the command to a NSS client. An IKE tunnel must be in place before a dynamic IPSec (phase 2) security association can be negotiated by the IKE daemon.

At times, multiple ISAKMP (phase 1) security associations that correspond to the same IKE tunnel can occur. By default only information about the most current ISAKMP security association for an IKE tunnel is displayed. Use the -c option to display information about all ISAKMP security associations corresponding to an IKE tunnel.

Security associations for use by a dynamic tunnel are negotiated under the protection of an ISAKMP security association. Specify the -e option to display information about IPSec security associations that were negotiated or are in the process of being negotiated under the protection an ISAKMP security association.

ipsec -y display

The `ipsec -y display` command displays information about dynamic tunnels installed in the default stack. Use the -p option to direct the command to another stack or the -z option to direct the command to a NSS IPSec client. A dynamic tunnel must be active before traffic matching a filter rule utilizing an IpDynVpnAction can be permitted.
At times, there might be multiple IPSec security associations that correspond to the same dynamic tunnel. By default, only information about the most current IPSec security association for a dynamic tunnel is displayed. Use the -c option to display information about all IPSec security associations that correspond to a dynamic tunnel.

The stack only knows about IPSec security associations that have been successfully negotiated. The IKE daemon knows about IPSec security associations that have been successfully negotiated as well as those currently being negotiated. At times, it is helpful to see information about IPSec security associations that are in the process of being negotiated. The -b option obtains information about IPSec security associations from the IKE daemon rather than the stack.

When a stack is a target for a distributed DVIPA it might contain IPSec security associations for a dynamic tunnel that was negotiated on behalf of the distributing stack. Such security associations are known as shadow security associations. The -s option obtains information about shadowed security associations.

**ipsec -i**

Use the `ipsec -i` command to display the SECCLASS value assigned to interfaces defined to the default stack. The -p option directs the command to another stack or the -z option to direct the command to a NSS IPSec client. The SECCLASS option of LINK or INTERFACE statement is used to assign a security classification to an interface. The LINK or INTERFACE statement is specified in the TCP/IP profile. SECCLASS can be specified as a filtering criteria on certain IP filter rules.

**ipsec -t**

Use the `ipsec -t` command to locate active filter rules for the default stack that match a specified traffic pattern. The -p option directs the command to another stack or the -z option to direct the command to a NSS IPSec client.

**ipsec -o**

Use the `ipsec -o` command to display the default stack’s port translation table. The -p option directs the command to another stack or the -z option to direct the command to a NSS IPSec client. Port translation is performed as needed for TCP and UDP connections that use a dynamic security association with a remote security endpoint that resides behind a NAT.

**ipsec -w**

Use the `ipsec -w` command to display network security information for each of an IKE daemon’s active NSS IPSec clients.

**ipsec -x**

Use the `ipsec -x` command to display a list of the NSS IPSec clients connected to the NSS server. Each NSS IPSec client represents a remote system made up of an IKE daemon and a TCP/IP stack. Use the client name with the -z option to direct any of the other `ipsec` commands to a specific client.

**Using the pasearch command**

You can use the `pasearch` commands listed in Table 75 to display information about the IPSec policy loaded by the Policy Agent for the stack:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pasearch -v a</td>
<td>Displays all IPSec policy</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>pasearch -v f</td>
<td>Displays IpFilterPolicy</td>
</tr>
<tr>
<td>pasearch -v k</td>
<td>Displays KeyExchangePolicy</td>
</tr>
<tr>
<td>pasearch -v l</td>
<td>Displays LocalDynVpnPolicy</td>
</tr>
</tbody>
</table>

The -p option can be used to obtain policy for a specific stack. Additional pasearch options can be used to obtain a more condensed display. Refer to [z/OS Communications Server: IP System Administrator's Commands](#) for a complete description of the pasearch command syntax.

**Using syslog messages**

The IKE daemon uses syslogd to write informational messages to the local4 facility. These messages contain information about the following:

- The state of the IKE daemon
- Successful and unsuccessful phase 1 and phase 2 negotiations
- Information about phase 1 and phase 2 negotiation failures

Additional IKE daemon debug information can be enabled by setting the IkeSyslog and PagentSyslogLevel parameters in the IKE configuration file. See “IKE daemon debug information” on page 349 for more details and sample IKE daemon syslog output.

If you are using the IBM Configuration Assistant for z/OS Communications Server to configure, the IKE Syslog level and the Policy Agent API Syslog level can be set from the IPSec: IKE Daemon Settings panel.

The Defense Manager daemon (DMD) uses syslogd to write informational messages to the local4 facility. DMD debug information can be enabled by setting the SyslogLevel parameter in the DMD configuration file. See “Defense Manager daemon debug information” on page 718 for more information.

The stack utilizes the TRMD daemon to write informational messages. The TRMD daemon uses syslogd to write these messages to the local4 facility. To enable many of these messages, IP filter logging must be turned on at both an IP filter policy level and an individual filter rule level. See “Steps for verifying IP security policy or defensive filter enforcement” on page 740 for details about enabling IP filter logging.
Chapter 32. Diagnosing OMPROUTE problems

This topic provides information and guidance to diagnose OMPROUTE problems, and contains the following sections:

- "Overview"
- “Definitions” on page 752
- "Diagnosing OMPROUTE problems” on page 752
- “OMPROUTE traces and debug information” on page 759
- “Starting OMPROUTE tracing and debugging from an MVS cataloged procedure or AUTOLOG” on page 760
- “TCP/IP services component trace for OMPROUTE” on page 772
- “Commands to enable, disable, and display the status of the OMPROUTE CTRACE” on page 776

Overview

For IPv4, OMPROUTE implements the Open Shortest Path First (OSPF) protocol described in RFC 1583, "OSPF Version 2" as well as the Routing Information Protocols (RIP) described in RFC 1058, "Routing Information Protocol" (RIP Version 1) and in RFC 1723, "RIP Version 2–Carrying Additional Information" (RIP Version 2).

For IPv6, OMPROUTE implements the IPv6 OSPF protocol described in RFC 2740, "OSPF for IPv6", as well as the IPv6 RIP protocol described in RFC 2080, "RIPng for IPv6".

OMPROUTE provides an alternative to the static TCP/IP BEGINROUTES or GATEWAY definitions. When configured properly, the MVS host running with OMPROUTE becomes an active OSPF or RIP router in a TCP/IP network. The dynamic routing protocols are used to dynamically maintain the host routing table. For example, OMPROUTE can determine that a new route has been created, that a route is temporarily unavailable, or that a more efficient route exists.

OMPROUTE has the following characteristics:

- It is a z/OS UNIX application. It requires the z/OS UNIX file system to operate.
- OMPROUTE can be started from an MVS procedure, from the z/OS shell, or from AUTOLOG. Refer to the z/OS Communications Server: IP Configuration Guide for information about OMPROUTE.
- The OMPROUTE subagent provides an alternative to DISPLAY commands for displaying IPv4 Open Shortest Path First (OSPF) protocol configuration and state information. The subagent implements the Management Information Base (MIB) variables defined in Request for Comment (RFC) 1850. The OMPROUTE subagent is controlled by statements in the OMPROUTE configuration file. For details, refer to the z/OS Communications Server: IP Configuration Reference.
- OMPROUTE needs to be started by a RACF authorized user ID.
- OMPROUTE needs to be in an APF authorized library.
- A one-to-one relationship exists between an instance of OMPROUTE and a TCP/IP stack. OSPF/RIP support on multiple TCP/IP stacks requires multiple instances of OMPROUTE.
• All IPv4 dynamic routes are deleted from the routing table upon initialization of OMPROUTE if there are IPv4 interfaces configured to OMPROUTE as RIP or OSPF interfaces.

• All IPv6 dynamic routes (with the exception of routes learned using the IPv6 Router Discovery protocol) are deleted from the routing table upon initialization of OMPROUTE if there are IPv6 interfaces configured to OMPROUTE as OSPF or RIP interfaces.

• IPv4 Internet Control Message Protocol (ICMP) redirects are ignored when OMPROUTE is active and there are IPv4 interfaces configured to OMPROUTE as RIP or OSPF interfaces.

• IPv6 ICMP redirects are ignored when OMPROUTE is active and there are IPv6 interfaces configured to OMPROUTE as OSPF or RIP interfaces.

• OMPROUTE does not make use of the BSD Routing Parameters. Instead, the maximum transmission unit (MTU), subnet mask, and destination address parameters for IPv4 interfaces are configured using the OSPF_Interface, RIP_Interface, and Interface statements in the OMPROUTE configuration file. Also, for IPv6, OMPROUTE does not update the stack’s MTU sizes but learns them from the stack instead.

**Restriction:** If using NCPROUTE, the BSD routing parameters in the BSDROUTINGPARMS TCP/IP configuration statement must be defined for the host-to-NCP channel interfaces, and the parameter values must match the corresponding values on the RIP_INTERFACE or INTERFACE statements in the OMPROUTE configuration file; otherwise, connection problems occur between NCPROUTE and its NCP clients.

• OMPROUTE uses the MVS operator console, SYSLOGD, STDOUT, and CTRACE for its logging and tracing:
  – The MVS operator console and SYSLOGD are used for major events such as initialization, termination, and error conditions.
  – STDOUT and z/OS UNIX file system files are used for detailed tracing and debugging.
  – CTRACE is used for the following purposes:
    - Tracing the receipt and transmission of OSPF/RIP packets
    - Tracing subagent/SNMP agent packets
    - Tracing communication between OMPROUTE and the TCP/IP stack
    - Detailed tracing and debugging

For details on using TCP/IP Services Component trace support with OMPROUTE, see "TCP/IP services component trace for OMPROUTE" on page 772 and Chapter 5, "TCP/IP services traces and IPCS support," on page 47.

• If you want to communicate a routing protocol over an interface, configure the interface to OMPROUTE using the OSPF_INTERFACE, RIP_INTERFACE, IPV6_OSPF_INTERFACE, or IPV6_RIP_INTERFACE configuration statement.

• IPv4 interfaces that are not involved in the communication of the RIP or OSPF protocol (except VIPA interfaces) must be configured to OMPROUTE using the INTERFACE configuration statement, unless it is a non-point-to-point interface and all default values are acceptable as specified on the INTERFACE statement. All IPv4 interfaces known to the TCP/IP stack should be defined to OMPROUTE with the correct subnet mask and MTU values. For IPv4 interfaces that are not defined to OMPROUTE, OMPROUTE assigns default subnet mask and MTU values to the interfaces, with possibly undesirable results.
IPv6 interfaces that are not involved in the communication of the OSPF or RIP protocol defaults to IPv6 generic interfaces when Global_Options Ignore_Undefined_Interfaces is coded to No (default value). The IPv6_Interface statement can be used if the IPv6 (generic) interface default values are not acceptable or you want to define additional IPv6 prefixes on the IPv6_Interface statement. If Global_Options Ignore_Undefined_Interfaces is coded to Yes, code IPv6_INTERFACE statements for all IPv6 Interfaces not involved in communication of OSPF or RIP that you want OMPROUTE to recognize.

OMPROUTE uses a standard message catalog. The message catalog must be in the z/OS UNIX file system. The directory location for the message catalog path is set by the environment variables NLSPATH and LANG.

If you want OMPROUTE to completely ignore IPv4 and IPv6 interfaces that are not defined to it, code the GLOBAL_OPTIONS statement with IGNORE_UNDEFINED_INTERFACES=YES in the OMPROUTE configuration file. For details, refer to the z/OS Communications Server: IP Configuration Guide.

OMPROUTE is enhanced with Virtual IP Addressing (VIPA) to handle network interface failures by switching to alternate paths. The virtual routes are included in the OSPF and RIP advertisements to adjacent routers. Adjacent routers learn about virtual routes from the advertisements and can use them to reach the destinations at the MVS host.

OMPROUTE allows for the generation of multiple, equal-cost routes to a destination, thus providing load-balancing support.

During a temporary shortage in storage, such as CSM ECSA or CSM data space conditions or reaching the TCP/IP defined limits for ECSA or private storage, OMPROUTE temporarily suspends the route timeout processing. This is done in an attempt to prevent the loss of routing information on the local host.

OMPROUTE works best without non-replaceable static routes, and the use of non-replaceable static routes (defined using the BEGINROUTES or GATEWAY TCP/IP configuration statement) is not recommended. Non-replaceable static routes might interfere with the discovery of a better route to the destination as well as inhibit the ability to switch to another route if the destination should become unreachable by way of the static route. For example, if you define a non-replaceable static host route through one interface and that interface becomes unreachable, OMPROUTE does not define a route to that same host through an alternate interface.

If you must define static routes, all static routes are considered to be of equal cost and non-replaceable static routes are not replaced by OSPF or RIP routes. Use extreme care when working with static routes and OMPROUTE. Set IMPORT_STATIC_ROUTES = YES on the AS_Boundary Routing or IPv6_AS_Boundary_Routing configuration statement, or both. Alternatively, set SEND_STATIC_ROUTES = YES on the RIP_Interface or IPv6_RIP_Interface configuration statement, or both. This allows the static routes to be advertised to other routers.

You can define static routes as replaceable. Unlike non-replaceable static routes, replaceable static routes are always replaced by dynamic routes learned by OMPROUTE. In other words, a replaceable static route is used only if no dynamic route is known to the destination. Replaceable static routes can be thought of as last resort routes to reach a destination when no dynamic route is known.
Definitions
OMPROUTE must be defined correctly to TCP/IP. For detailed information about TCP/IP definitions, refer to the information on configuring OMPROUTE in the z/OS Communications Server: IP Configuration Reference.

Diagnosing OMPROUTE problems
Problems with OMPROUTE are generally reported under one of the following categories:
- Abends
- OMPROUTE connection problems
- Routing failures
- Adjacency failures

These categories are described in the following sections.

Abends
An abend during OMPROUTE processing should result in messages and error-related information being sent to the system console. A dump of the error is needed unless the symptoms match a known problem. If a dump was not taken, ensure the Language Environment run-time options TRAP(ON,NOSPIE) TERMTHDACT(UAIMM) are set for OMPROUTE.

OMPROUTE connection problems
OMPROUTE connection problems are reported when OMPROUTE is unable to connect to TCP/IP or to one of the ports required for OSPF or RIP communication. These problems are generally caused by an error in the configuration or definitions in TCP/IP.

In a common INET environment (multiple stacks), OMPROUTE attempts to connect to a stack whose name is determined by the TCPIPjobname keyword in the resolver configuration data set or file. If OMPROUTE cannot determine the TCPIPjobname, it uses a default of INET. If OMPROUTE cannot communicate with the stack pointed to by TCPIPjobname or is unable to initialize its required ports, it issues an error message describing the problem and then terminates.

For details on diagnosing problems while attempting to connect to the SNMP agent, see "SNMP connection problems" on page 610.

Routing failures
Routing problems are usually the result of outages in a network and a lack of alternative routing paths available for recovery. Refer to "Steps for verifying IP routing to a destination when not using policy-based routing (PBR)" on page 32 and "Steps for diagnosing problems with IP routing to a destination when using policy-based routing (PBR)" on page 34 for help with diagnosing routing failures.
Table 76 describes command terms used in this section.

**Table 76. OMPROUTE command terms**

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NETSTAT ROUTE</td>
<td>Refers to the Netstat ROUTE/-r command and the netstat route commands used on other platforms.</td>
</tr>
<tr>
<td>OMPROUTE RTTABLE</td>
<td>Refers to the D TCPIP,tcpipjobname,OMPROUTE,RTTABLE command for displaying OMPROUTE IPv4 route tables.</td>
</tr>
<tr>
<td>OMPROUTE RT6TABLE</td>
<td>Refers to the D TCPIP,tcpipjobname,OMPROUTE,RT6TABLE command for displaying OMPROUTE IPv6 route tables.</td>
</tr>
<tr>
<td>PING</td>
<td>Refers to z/OS UNIX ping, TSO PING, and the ping commands used on other platforms.</td>
</tr>
<tr>
<td>Traceroute</td>
<td>Refers to z/OS UNIX traceroute, TSO TRACERTE, and the traceroute commands used on other platforms.</td>
</tr>
</tbody>
</table>

**Analyzing routing failures**

**Guidelines:** When analyzing routing failures, follow these guidelines:

- Make sure that the address used in attempting to contact the remote host is a valid IP address.
- Make sure routing is possible in both directions. For most TCP/IP communication, two-way routing is required. The origin must have routes to reach the destination, and the destination must have routes to reach the origin. If NETSTAT ROUTE at the origin shows correct routing, you must also use NETSTAT ROUTE at the destination to verify that it can send replies back to the origin. If there are intermediate hops between the source and destination, all routing tables must have routing information. For example, if the origin node routing table indicates that the first hop to reach the destination is router A, then the router A routing table must also have a valid, active route to the destination, and so on. This also applies to the return route.
- Also, this is affected by SOURCEVIPA. If SOURCEVIPA is enabled at the origin of the communication, then the destination and all intermediate hops must be able to route back to the VIPA.
- If the NETSTAT ROUTE output on the source, the destination, or an intermediate hop does not show the expected routes, do one or more of the following:
  - Make sure that the routers involved in providing routing information are operational and participating in the correct routing protocol.
  - Make sure that the necessary physical connections are active.
  - Use the OMPROUTE DISPLAY commands described in the [z/OS Communications Server: IP System Administrator's Commands](https://www.ibm.com/support/knowledgecenter/en/SSFKQJ_7.1.0/com.ibm.zos.zos/cman_ipadm_ref.htm) to determine if anything in the configuration or current state of OMPROUTE has caused the unexpected NETSTAT ROUTE information.

**Documenting routing failures**

You should gather documentation described in “Documentation for the IBM Support Center” on page 43 for initial diagnosis of all routing failures. If dynamic
Routing is being provided by OMPROUTE and the expected dynamic routes have not been installed in the stack route table, the following documentation should also be available:

- MVS system log
- SYSLOGD
- The data set containing OMPROUTE trace and debug information. If OMPROUTE trace and debug information is being redirected to the OMPROUTE CTRACE internal buffer, this buffer is included in a dump of the OMPROUTE address space. For details, see "OMPROUTE traces and debug information" on page 759 and "TCP/IP services component trace for OMPROUTE" on page 772.
- Output from OMPROUTE RTTABLE or RT6TABLE commands. If using policy-based routing, collect output for the appropriate route tables.
- Output from any other OMPROUTE DISPLAY commands used.

**Adjacency failures**

OMPROUTE adjacency failures are reported when OMPROUTE is unable to establish adjacency with a neighboring router, or loses an established adjacency, over one of its network interfaces. The following error messages are used to report adjacency failures:

For IPv4: EZZ7921I OSPF Adjacency Failure, neighbor neighbor, old state state, new state state, event event

For IPv6: EZZ7954I IPv6 OSPF Adjacency Failure, neighbor neighbor, old state ostate, new state nstate, event event

In addition to the adjacency failure message, if the futile neighbor state loop detection is enabled in OMPROUTE (Max_Adj_Attempt parameter on OSPF and IPV6OSPF statements), the following error messages will be issued to report futile neighbor state loops for the adjacency attempts:

For IPv4: EZZB157I jobname IPv4 OSPF detected futile neighbor state loop with neighbor neighbor on interface interface after threshold_value adjacency attempts

For IPv6: EZZB157I jobname IPv6 OSPF detected futile neighbor state loop with neighbor neighbor on interface interface after threshold_value adjacency attempts

If OMPROUTE is configured with redundant parallel interfaces (primary and backup) attached to the same LAN segment, OMPROUTE will try to form adjacency with a neighboring designated router over the alternate redundant interface when the futile neighbor state loop has been detected on the problematic interface. The following informational messages will be issued to report the interface changes:

For IPv4: EZZB158I jobname IPv4 OSPF could not establish adjacency on interface interface1 - attempting to establish adjacency on interface interface2

For IPv6: EZZB158I jobname IPv6 OSPF could not establish adjacency on interface interface1 - attempting to establish adjacency on interface interface2

For information the Max_Adj_Attempt parameter, see the OMPROUTE chapter in z/OS Communications Server: IP Configuration Reference.

For information on futile neighbor state loops, see the section on "Network design considerations with z/OS Communications Server" in IP Configuration Guide.
Documenting adjacency failures

The following documentation should be available for initial diagnosis of adjacency failures:

- MVS system log
- SYSLOGD
- The data set containing OMPROUTE trace and debug information, unless trace and debug information is being redirected to the OMPROUTE CTRACE internal buffer, which is automatically included in a dump of the OMPROUTE address space. For details, see “OMPROUTE traces and debug information” on page 759.
- TCP/IP and OMPROUTE CTRACE. For information about generating an OMPROUTE Component Trace, see “TCP/IP services component trace for OMPROUTE” on page 772.
- Output from appropriate OMPROUTE DISPLAY commands as described in z/OS Communications Server: IP System Administrator’s Commands.

If using IPv4, use the following commands to display OSPF configuration, interfaces, neighbors, and routing table:

- DISPLAY TCPIP,,OMPROUTE,OSPF,,LIST,ALL
- DISPLAY TCPIP,,OMPROUTE,OSPF, INTERFACES
- DISPLAY TCPIP,,OMPROUTE,OSPF, INTERFACE,NAME=if_name
- DISPLAY TCPIP,,OMPROUTE,OSPF,NEIGHBOR
- DISPLAY TCPIP,,OMPROUTE,OSPF,NEIGHBOR,IPADDR=ip-addr
- DISPLAY TCPIP,,OMPROUTE,RTTABLE

If using IPv6, use the following commands to display OSPF configuration, interfaces, and neighbors, and routing table:

- DISPLAY TCPIP,,OMPROUTE,IPV6OSPF,ALL
- DISPLAY TCPIP,,OMPROUTE,IPV6OSPF, INTERFACES
- DISPLAY TCPIP,,OMPROUTE,IPV6OSPF, INTERFACE,NAME=if_name [Note: ID=if-id can be used instead of NAME parameter]
- DISPLAY TCPIP,,OMPROUTE,IPV6OSPF,NEIGHBOR
- DISPLAY TCPIP,,OMPROUTE,IPV6,NEIGHBOR,ID=router-id [Note: IFNAME=if_name can be used instead of ID parameter]
- DISPLAY TCPIP,,OMPROUTE,RT6TABLE

- When applicable, the dumps of TCPIP and VTAM address and data spaces. A SLIP on OMPROUTE adjacency failing message (for example, EZZ7921I ) may be used to capture the OMPROUTE trace and debug information as well as the dumps of TCPIP and VTAM address and data spaces at the time of the error.

Analyzing adjacency failures

An adjacency failure is determined by the neighbor event code as reported by OMPROUTE in the informational message provided. The events associated with adjacency failures are:
Table 77. Neighbor event code associated with adjacency failures

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Sequence number mismatch</td>
<td>OMPROUTE has received a sequence number mismatch in a database descriptor packet. A neighbor may be attempting to restart the adjacency for some reason resulting in sequence number mismatches. This event usually indicates that the neighbor has not been receiving hello packets from OMPROUTE, has experienced event 12 on its side, and is trying to restart.</td>
</tr>
<tr>
<td>8</td>
<td>Bad link state request</td>
<td>OMPROUTE has received a bad Link State request (LSA) packet from a neighbor. A subverted router in the network may have modified the contents of a LSA to result in maximum sequence number or maximum age attacks. From the LSA floods, the neighbors will replace the good LSA with the bad LSA as newer in their databases until they naturally ages out usually in a maximum of 1 hour.</td>
</tr>
<tr>
<td>12</td>
<td>No hellos seen recently</td>
<td>OMPROUTE has not received hello packets from a neighbor for a full DEAD_ROUTER_INTERVAL and as a consequence, OMPROUTE assumes that the neighbor is down.</td>
</tr>
<tr>
<td>15</td>
<td>Failure to thrive</td>
<td>OMPROUTE was trying to establish adjacency with a neighbor but failed to complete within the DB_EXCHANGE_INTERVAL.</td>
</tr>
</tbody>
</table>

Along with the neighbor event code, the old neighbor state indicates the highest neighbor state that OMPROUTE has reached for the adjacency attempt and the new neighbor state reflects the changed state. Starting at the 2-way state for bidirectional communication, the following neighbor states are used by OMPROUTE in attempts to reach full adjacency with a neighboring designated router:

Table 78. Neighbor state associated with adjacency failures

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2-way communication</td>
<td>The neighbor and OMPROUTE have seen and acknowledged each other’s hello packets.</td>
</tr>
<tr>
<td>16</td>
<td>Database exchange start</td>
<td>OMPROUTE is negotiating master and slave roles with a neighbor.</td>
</tr>
<tr>
<td>32</td>
<td>Database exchange</td>
<td>OMPROUTE is exchanging database information with a neighbor in the form of database descriptor packets.</td>
</tr>
<tr>
<td>64</td>
<td>Loading</td>
<td>OMPROUTE is requesting newer pieces from a neighbor’s database that are more up-to-date in the form of link state request packets.</td>
</tr>
<tr>
<td>128</td>
<td>Full</td>
<td>OMPROUTE has established full adjacency with a neighbor.</td>
</tr>
</tbody>
</table>
The order of the neighbor transit states for establishing adjacency is 8, 16, 32, 64, and 128. Whenever a problem is detected for some reason between those states before reaching full adjacency, OMPROUTE will reset the neighbor state to 2-way (8) and repeat the process on a continuous basis even to the point where it becomes futile. A futile neighbor state loop is seen as a successive repetitive pattern of transit states and ones that do not appear to reach full adjacency. For example, typical patterns are: 8-16, 8-16 or 8-16-32, 8-16-32, etc. After each adjacency failure, OMPROUTE will continue to attempt to establish adjacency with a neighbor over the same network interface. If futile neighbor state loop detection is enabled and if there are redundant parallel interfaces (primary or backup) attached to the same LAN segment available, OMPROUTE will suspend the problematic interface and retry the adjacency attempt over the alternate interface. The other option is to use the MODIFY OMPROUTE commands to manually suspend and activate an alternate redundant parallel OSPF interface so that adjacency with the neighbor will be attempted over that interface.

For information on futile neighbor state loops, see the section on "Network design considerations with z/OS Communications Server" in z/OS Communications Server: IP Configuration Guide. For details on the MODIFY OMPROUTE commands, see z/OS Communications Server: IP System Administrator's Commands.

OMPROUTE can drop adjacencies under the following conditions:

- Other workloads on the z/OS machine keeps OMPROUTE from dispatching enough CPU cycles:
  - Dumps are being taken while OMPROUTE is running. All address spaces are marked non-dispatchable during a dump processing. If the dump takes longer than a DEAD_ROUTER_INTERVAL, the adjacencies will fail.
  - There are too many other address spaces running higher priority than OMPROUTE. Because OMPROUTE is a time-sensitive application and manages the TCPIP’s routing table, OMPROUTE should be one less than TCPIP’s dispatching priority. If using WLM goal modes, OMPROUTE should be in the same service class as TCPIP’s.

- Not enough dispatching for OMPROUTE:
  - The dispatching priority for OMPROUTE is too low. Either increase OMPROUTE’s dispatching priority or increase the DEAD_ROUTER_INTERVAL values.
  - OMPROUTE is running as a BPXBATCH program. Because there are other applications using the BPXBATCH program, they might steal CPU cycles from OMPROUTE. Change OMPROUTE not to use the BPXBATCH program.

- Increased workload in OMPROUTE:
  - OMPROUTE is a designated router or a backup designated router. OMPROUTE will perform link state database management related tasks which can contribute to high workloads. These high workloads can impact OMPROUTE in processing of inbound hello packets necessary to maintain adjacencies with its neighbors. Either increase the DEAD_ROUTER_INTERVAL values or change the ROUTER_PRIORITY values to reduce the likelihood of OMPROUTE becoming elected as a designated router. A z/OS machine is not designed to be a full-fledged router and it is best to offload this work of link state database management to the neighboring network routers. That is, configure the network routers on the attached LAN segment to be elected as designated routers when possible.
  - OMPROUTE is running with too much tracing. OMPROUTE debug trace using file I/O can contribute to adjacency failures (for example, missed hello packets). Use the OMPROUTE CTRACE method when possible.
- OMPROUTE routing table is very large. Once OMPROUTE has more than 1000-2000 routes, adjacency failures (for example, missed hello packets) might occur because of the increased workload from processing routing table and link state updates. Configure OMPROUTE to use stub areas when possible and try to keep z/OS out of backbone areas.

- OMPROUTE has too many adjacencies This may be notable when using XCF in a sysplex environment. Because of increased workload from adjacency communications, adjacency failures (for example, missed hello packets) might occur. Determine if it is necessary for XCF interfaces to be configured to use OSPF.

- Network hardware problem:
  - Attached switch or router not functioning correctly.
  - Poor or faulty network cable connections.
    A network hardware problem that is beyond detection by TCP/IP or OMPROUTE can contribute to adjacency failures and futile neighbor state loops. If futile neighbor state loop detection is enabled and if there are redundant parallel interfaces attached to the same LAN segment, OMPROUTE will attempt adjacency with the neighbor using an alternate interface. When necessary, use the MODIFY OMPROUTE commands to manually suspend and activate an alternate redundant parallel OSPF interface so that adjacency with the neighbor will be attempted over that interface. OMPROUTE might circumvent the network hardware problem using the alternate interface.

For the symptom of missed inbound hello packets, if it has been determined that the neighbor is sending the hello packets and the TCP/IP stack is forwarding them to the OMPROUTE application at the upper layer, there is an OMPROUTE option that might help alleviate this problem. In the OMPROUTE standard environmental variable file, use "OMPROUTE_OPTIONS=hello_hi" to force OMPROUTE to process the inbound hello packets at a higher priority. The other condition is when the TCP/IP stack may not be getting dispatched as often enough to forward the inbound hello packets to OMPROUTE. In this case, ensure that appropriate dispatching priorities are assigned to the TCP/IP stack and OMPROUTE.

To track adjacency problems, do the following:
- Issue the command to display the OSPF interfaces and analyze the following fields from the report:
  - STATE for the current interface state
  - #NBRS for the total number of neighbors whose hellos have been received, plus those that have been configured.
  - #ADJS for the total number of neighbors in state exchange or greater. These are the neighbors with whom the router has synchronized or is in the process of synchronization.
- Issue the command to display a detailed OSPF interface and analyze the following fields from the report:
  - DESIGNATED ROUTER to determine if OMPROUTE is a designated router or not.
  - BACKUP DR to determine if OMPROUTE is a backup designated router or not.
  - DR PRIORITY to determine the interface router priority. A higher value indicates that this OMPROUTE is more likely to become the designated router. A value of 0 indicates that OMPROUTE will never become the designated router.
- #NEIGHBORS for total number of neighbors whose hellos have been received, plus those that have been configured.
- #ADJACENCIES for total number of neighbors in state Exchange or greater. These are the neighbors with whom the router has synchronized or is in the process of synchronization.
- #FULL ADJS for total number of full adjacencies. This is the number of neighbors whose state is Full (and therefore with which the router has synchronized databases).
- #MCAST FLOODS for the total number of link state updates that flooded the interface (not counting retransmissions).

* Issue the command to display the OSPF neighbors and analyze the following field from the report:
  - STATE for the current neighbor state.
* Issue the command to display a detailed OSPF neighbor and analyze the following fields from the report:
  - NEIGHBOR STATE for the current neighbor state.
  - DR PRIORITY to determine the neighbor’s router priority.
  - #ADJ RESETS for the total number of transitions to state ExStart from a higher state.
  - #NBR LOSSES for total number of times the neighbor has transitioned to the down state.

**OMPROUTE traces and debug information**

There are many TCP/IP traces that can be useful in identifying the cause of OMPROUTE problems. OMPROUTE’s use of the MVS Component Trace support is also useful (see “TCP/IP services component trace for OMPROUTE” on page 772). This section describes the OMPROUTE internal traces. OMPROUTE internal tracing and debugging can be started when OMPROUTE is started. Also, the MODIFY command can be used to start, stop, or alter OMPROUTE tracing and debugging after OMPROUTE has been started.

This section describes each of these methods.

**Starting OMPROUTE tracing and debugging from the z/OS UNIX System Services shell**

If OMPROUTE is started from the z/OS UNIX System Services shell command line (using the omproute command), you can specify the following parameters to indicate the level of tracing or debugging desired.

- **-tn and -6tn (where n is a supported trace level)**
  
  These options specify the OMPROUTE external tracing levels, with -tn covering both OMPROUTE initialization and IPv4 routing protocols and -6tn covering IPv6 routing protocols. These options provide information about the operation of the routing application and can be used for many purposes, such as debugging a configuration, education on the operation of the routing application, verification of test cases, and so on. The following trace levels are supported:
  
  - 1 = Informational messages
  - 2 = Formatted packet trace

- **-sn (where n is a supported debug level)**
This option specifies the internal debugging level for the OMPROUTE subagent. It provides internal debugging information needed for debugging problems. The following level is supported:
- 1 = Internal debugging messages. This turns on DPIdebug(2).

- **-dn and -6dn (where n is a supported debug level)**
  These options specify the OMPROUTE internal debugging levels, with -dn covering both OMPROUTE initialization and IPv4 routing protocols and -6dn covering IPv6 routing protocols. These options provide internal debugging information needed for debugging problems. The following levels are supported:
  - 1 = Internal debugging messages.
  - 2 = Unformatted hexadecimal packet trace
  - 3 = Function entry or exit trace
  - 4 = Task add or run

**Guidelines:**
- The -tn, -6tn, -dn, and -6dn options affect OMPROUTE performance. As a result, you might have to increase the Dead Router Interval on OSPF and IPv6 OSPF interfaces to prevent neighbor adjacencies from collapsing.
- The trace and debug levels are cumulative; each level includes all lower levels. For example, -t2 provides formatted packet trace and informational messages. You can enter more than one parameter by inserting a space after each parameter, for example, omproute -t1 -d2, which is the trace level most often requested by support. For more information, refer to APAR II12026.
- Parameters can be specified in mixed case.

**Starting OMPROUTE tracing and debugging from an MVS cataloged procedure or AUTOLOG**
The OMPROUTE tracing and debugging are controlled by parameters on PARM= when OMPROUTE is started from an MVS cataloged procedure or AUTOLOG. For example:

```
//OMPROUTE EXEC PGM=OMPROUTE,REGION=10M,TIME=NOLIMIT,
// Parm=('POSIX(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/-t2 -d1')
```

For a description of the parameters that can be specified, see ["Starting OMPROUTE tracing and debugging from the z/OS UNIX System Services shell"](on page 759).

**Starting OMPROUTE tracing and debugging using the MODIFY command**
Whether you start OMPROUTE from the z/OS UNIX System Services shell or from a MVS cataloged procedure, you can use the MODIFY command to start logging or tracing, to stop logging or tracing, and to change the level of logging or tracing.

The syntax for these MODIFY commands follows:
- **MODIFY procname,TRACE=trace-level**
  Use the TRACE command to change the trace level for OMPROUTE initialization as well as IPv4 routing protocols.
  - TRACE=0 turns off OMPROUTE tracing.
  - TRACE=1 gives all the informational messages.
  - TRACE=2 gives the informational messages plus formatted packet tracing.
• **MODIFY** `procname, TRACE6=trace-level`
  Use the TRACE6 command to change the trace level for IPv6 routing protocols.
  - TRACE6=0 turns off OMPROUTE tracing.
  - TRACE6=1 gives all the informational messages.
  - TRACE6=2 gives the informational messages plus formatted packet tracing.
• **MODIFY** `procname, DEBUG=debug-level`
  Use the DEBUG command to change the debug level for OMPROUTE initialization as well as IPv4 routing protocols.
  - DEBUG=0 turns off OMPROUTE debugging.
  - DEBUG=1 gives internal debug messages.
  - DEBUG=2 gives the same as DEBUG=1 plus hexadecimal packet tracing.
  - DEBUG=3 gives the same as DEBUG=2 plus module entry and exit.
  - DEBUG=4 gives the same as DEBUG=3 plus task add and run.
• **MODIFY** `procname, DEBUG6=debug-level`
  Use the DEBUG6 command to change the debug level for IPv6 routing protocols.
  - DEBUG6=0 turns off OMPROUTE debugging.
  - DEBUG6=1 gives internal debug messages.
  - DEBUG6=2 gives the same as DEBUG6=1 plus hexadecimal packet tracing.
  - DEBUG6=3 gives the same as DEBUG6=2 plus module entry and exit.
  - DEBUG6=4 gives the same as DEBUG6=3 plus task add and run.
• **MODIFY** `procname, SADEBUG=trace-level`
  Use the SADEBUG command to start and stop message logging for the OMPROUTE subagent and to stop DPI tracing:
  - SADEBUG=0 stops message logging for the OMPROUTE subagent and issues DPIdebug(0) to stop DPI tracing.
  - SADEBUG=1 generates all messages by the OMPROUTE subagent and DPIdebug(2).

**Destination of OMPROUTE trace and debug output**

If the OMPROUTE CTRACE with option DEBUGTRC (or option ALL) is not enabled, then output from OMPROUTE tracing and debugging is written to the debug output destination. The debug output destination is based on the OMPROUTE_DEBUG_FILE and OMPROUTE_IPV6_DEBUG_FILE environment variables. If OMPROUTE was started without tracing enabled and OMPROUTE_DEBUG_FILE/OMPROUTE_IPV6_DEBUG_FILE is not defined and tracing is started later using the MODIFY command, the trace output destination is `$TMP/omproute_debug`, where `$TMP` is the value of the TMP environment variable.

When OMPROUTE_DEBUG_FILE is defined, the first trace file created for OMPROUTE initialization and IPv4 routing protocol tracing is named using the value coded on OMPROUTE_DEBUG_FILE. When OMPROUTE_IPV6_DEBUG_FILE is defined, the first trace file created for IPv6 routing protocol tracing is named using the value coded on OMPROUTE_IPV6_DEBUG_FILE. When either of these first files is full, the extensions are changed to 00N, where N is in the range of 1 to the number of files specified in the OMPROUTE_DEBUG_FILE_CONTROL environment variable (default 4). The current file is always the file named using the value coded on OMPROUTE_DEBUG_FILE/OMPROUTE_IPV6_DEBUG_FILE and the oldest file is...
the highest N value. This eliminates the danger of OMPROUTE filling the z/OS UNIX file system when tracing is active for a long time.

The size and number of debug files created can be controlled by the OMPROUTE_DEBUG_FILE_CONTROL environment variable. This allows you to adjust how much OMPROUTE trace data is saved. You tailor this parameter to your network complexity or available z/OS UNIX file system storage capacity. Refer to the z/OS Communications Server: IP Configuration Guide for details on this environment variable.

If the OMPROUTE CTRACE with option DEBUGTRC (or option ALL) is enabled, then output from OMPROUTE tracing and debugging is sent to the CTRACE facility. The OMPROUTE CTRACE facility can write trace records to an internal buffer or to an external writer. When the OMPROUTE CTRACE with option DEBUGTRC (or option ALL) is active, the normal debug output destinations are ignored. If the CTRACE is disabled, and a trace level is modified, then OMPROUTE once again follows the above rules for determining the debug output destination.

Sample OMPROUTE trace output

Figure 98 on page 763 is a sample OMPROUTE initialization and IPv4 routing protocol trace with descriptions for some of the trace entries:
EZZ78001 OMPROUTE starting
EZZ7845I Established affinity with TCPCS8
EZZ7817I Using defined OSPF protocol 89
EZZ7883I Processing interface from stack, address 9.169.100.18, name CTC2, index 2, flags 451
EZZ7883I Processing interface from stack, address 9.67.100.8, name CTC1, index 1, flags 451
EZZ8023I The RIP routing protocol is Enabled
EZZ8036I The IPv6 RIP routing protocol is Enabled
EZZ7937I The OSPF routing protocol is Enabled
EZZ8050I Updating BSD Route Parms for link CTC1, MTU 1024, metric 1, subnet 255.255.255.0, destination 0.0.0.0
EZZ8057I Added network 9.67.100.0 to interface 9.67.100.8 on net 0 interface CTC1, table EZBMAIN
EZZ7827I Adding stack route to 9.67.100.0, mask 255.255.255.0 via 0.0.0.0, link CTC1, metric 1, type 1, table EZBMAIN
EZZ8057I Added network 9.67.100.7 to interface 9.67.100.8 on net 0 interface CTC1, table EZBMAIN
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0 interface CTC1
EZZ7891I Joining multicast group 224.0.0.5 on interface 9.67.100.8
EZZ7913I State change, interface 9.67.100.8, new state 16, event 1

EZZ7875I No IPv4 Default Route Installed for table EZBMAIN
EZZ8001I OMPROUTE subagent Starting
EZZ7898I OMPROUTE Initialization Complete
EZZ8101I OMPROUTE subagent Initialization Completed
EZZ7908I Received packet type 1 from 9.167.100.13
EZZ8011I Sending request to address 9.67.100.7
EZZ8015I Sending packet to 9.67.100.7
EZZ8015I Sending packet to 9.169.100.14
EZZ8015I Sending packet to 9.67.100.7
EZZ8012I Sending broadcast response to address 9.67.100.255 in 1 packets with 1 routes
EZZ8015I Sending packet to 9.169.100.14
EZZ8012I Sending broadcast response to address 9.169.100.255 in 1 packets with 1 routes
EZZ7908I Received packet type 1 from 9.67.100.7
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0 interface CTC1
EZZ7891I State change, neighbor 9.67.100.7, new state 4, event 1
EZZ7891I State change, neighbor 9.67.100.7, new state 8, event 3
EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8 org 9.67.100.8
EZZ7919I State change, neighbor 9.67.100.7, new state 16, event 14
EZZ7910I Sending multicast, type 2, destination 224.0.0.5 net 0 interface CTC1
EZZ7908I Received packet type 2 from 9.67.100.7
EZZ7919I State change, IPv4 neighbor 9.67.100.7, new state 32, event 5
EZZ7910I Sending multicast, type 3, destination 224.0.0.5 net 0 interface CTC1
EZZ7908I Received packet type 2 from 9.67.100.7
EZZ7908I Received packet type 4 from 9.67.100.7

Figure 98. Sample OMPROUTE Trace Output (Part 1 of 6)
16 EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id 9.67.100.7 org 9.67.100.7
EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id 9.67.100.8 org 9.67.100.8
EZZ7927I from 9.67.100.7, self update: typ 1 id 9.67.100.8 org 9.67.100.8
EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id 9.167.100.13 org 9.100.13
EZZ7928I from 9.67.100.7, new LS advertisement: typ 5 id 9.67.100.0 org 9.67.100.8
EZZ7927I from 9.67.100.7, self update: typ 5 id 9.67.100.0 org 9.67.100.8
EZZ7928I from 9.67.100.7, new LS advertisement: typ 5 id 9.169.100.0 org 9.67.100.8
EZZ7927I from 9.67.100.7, self update: typ 5 id 9.169.100.0 org 9.67.100.8
EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8 org 9.67.100.8
EZZ7910I Sending multicast, type 4, destination 224.0.0.5 net 0 interface CTC1
EZZ7910I Sending multicast, type 3, destination 224.0.0.5 net 0 interface CTC1
EZZ7908I Received packet type 4 from 9.67.100.7
EZZ7928I from 9.67.100.7, new LS advertisement: typ 5 id 9.169.100.14 org 9.67.100.8
EZZ7927I from 9.67.100.7, self update: typ 5 id 9.169.100.14 org 9.67.100.8
EZZ7910I Sending multicast, type 2, destination 224.0.0.5 net 0 interface CTC1
EZZ7908I Received packet type 2 from 9.67.100.7
EZZ7919I State change, neighbor 9.67.100.7, new state 128, event 6
EZZ7908I Received packet type 5 from 9.67.100.7
EZZ7910I Sending multicast, type 5, destination 224.0.0.5 net 0 interface CTC1
EZZ8015I sending packet to 9.169.100.14
EZZ8012I sending broadcast response to address 9.169.100.255 in 1 packets with 1 routes
EZZ8015I sending packet to 9.67.100.7
EZZ8012I sending broadcast response to address 9.67.100.255 in 1 packets with 1 routes
EZZ8015I sending packet to 9.169.100.14
EZZ8012I sending broadcast response to address 9.169.100.255 in 1 packets with 1 routes
EZZ7908I Received packet type 4 from 9.67.100.7
EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id 9.67.100.7 org 9.67.100.7
EZZ7910I Sending multicast, type 5, destination 224.0.0.5 net 0 interface CTC1
EZZ7934I Originating LS advertisement: typ 5 id 9.169.100.14 org 9.67.100.8
EZZ7934I Originating LS advertisement: typ 5 id 9.169.100.0 org 9.67.100.8
EZZ7934I Originating LS advertisement: typ 5 id 9.67.100.0 org 9.67.100.8
EZZ7910I Sending multicast, type 4, destination 224.0.0.5 net 0 interface CTC1

Figure 98. Sample OMPROUTE Trace Output (Part 2 of 6)
Figure 98. Sample OMPROUTE Trace Output (Part 3 of 6)
Figure 98. Sample OMPROUTE Trace Output (Part 4 of 6)
Figure 98. Sample OMPROUTE Trace Output (Part 5 of 6)
Following are brief explanations of numbered items in the trace:

1. OMPROUTE initializing (trace level 1 was specified at startup: -t1).
2. OMPROUTE learns of TCP/IP stack IPv4 interfaces.
IPv6 tracing is in the file pointed to by the OMPROUTE_IPV6_DEBUG_FILE environment variable.

Direct routes are added for each TCP/IP stack IPv4 interface.

OSPF Hello packet sent out OSPF interface.

OSPF Interface transitions to state “point-to-point.”

RIP Requests & Responses begin being sent out RIP interface.

OSPF Hello packet received from OSPF neighbor.

OSPF neighbor transitions to state “Init.”

OSPF neighbor transitions to state “2-Way.”

OSPF neighbor transitions to state “ExStart.”

OSPF Database Description packet sent out OSPF interface.

OSPF Database Description received from OSPF neighbor.

OSPF neighbor transitions to state “Exchange.”

OSPF Link State Request packet sent out OSPF interface.

OSPF Link State Update packet received from OSPF neighbor.

Link State Advertisements from received Update packet are processed.

OSPF Link State Update packet sent out OSPF interface.

OSPF neighbor transitions to state “Full.”

OSPF Link State Acknowledgment packet received from OSPF neighbor.

OSPF Link State Acknowledgment packet sent out OSPF interface.

OSPF Dijkstra calculation is performed.

Learned route is added to TCP/IP stack IPv4 route table.

Adjacency establishment begins with router at other end of OSPF Virtual Link.

Request received to display OSPF Interface configuration information.

Request received to change IPv4 tracing level to 2 (adds formatted packets).

Request received to change IPv6 tracing level to 2 (adds formatted packets to trace output in the file pointed to by the OMPROUTE_IPV6_DEBUG_FILE environment variable).

Formatted OSPF packet.

Formatted RIP packet.

Request received to change tracing level back to 1(-t1).

OMPROUTE learns of stopped TCP/IP IPv4 interface.

Routes over stopped interface are deleted.

Neighbor over stopped interface transitions to state “Down.”

Stopped interface transitions to state “Down.”

The following sample shows OMPROUTE IPv6 routing protocol trace with descriptions for some of the trace entries:
EZ79771 Processing IPv6 interface from stack, address 1977::7, name MPCPTPV67, index 16, flags 811, flags2 0
EZ79771 Processing IPv6 interface from stack, address fe80::542c:ed1e:1362:4d26, name MPCPTPV67, index 16, flags 811, flags2 2
EZ79771 Processing IPv6 interface from stack, address 7:7:7:7:7:7:7:7, name VIPA16, index 18, flags 4001, flags2 0

EZ780571 Added network 1977::7 to interface fe80::542c:ed1e:1362:4d26 on net 16 interface MPCPTPV67, table EZBMAIN
EZ780571 Added network 7:7:7:: to interface 7:7:7:7:7:7:7:7 on net 18 interface VIPA16, table EZBMAIN

EZ78271 Adding stack route to ::, prefixlen 0 via fe80::7cb6:c5d5:6593:c076, link MPCPTPV67, metric 0, type 136, table EZBMAIN

EZ80111 send request to address ff02::9
EZ80151 sending packet to ff02::9
EZ80215 sending IPv6RIP response to address ff02::9 from fe80::542c:ed1e:1362:4d26 in 1 packets with 6 routes
EZ80041 response received from host fe80::846e:70a6:8ca6:48b7
EZ78271 Adding stack route to ::, prefixlen 0 via fe80::846e:70a6:8ca6:48b7, link MPCPTPV67, metric 0, type 136, table EZBMAIN
EZ780111 Deleting stack route to ::, prefixlen 0 via fe80::7cb6:c5d5:6593:c076, link MPCPTPV67, metric 0, type 136, table EZBMAIN
EZ80101 update route to net :: at metric 9 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ78061 Changing stack route to ::, prefixlen 0 via fe80::846e:70a6:8ca6:48b7, link MPCPTPV67, metric 9, type 136, table EZBMAIN
EZ80101 update route to net 1967::6 at metric 2 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ80101 update route to net 1977::7 at metric 1 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ80101 update route to net 6:6:6:: at metric 2 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ80101 update route to net 1946::6 at metric 3 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ80101 update route to net 9::67:120:4 at metric 3 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ80101 update route to net 9::67:120:6 at metric 3 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ80101 update route to net 9::67:120:7 at metric 3 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ78271 Adding stack route to 1946::6 at metric 2 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ80101 update route to net 1946::6 at metric 3 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ78271 Adding stack route to 1946::6 at metric 3 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ80101 update route to net 1946::6 at metric 3 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ78271 Adding stack route to 1946::4 at metric 3 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN
EZ78271 Adding stack route to 1946::4 at metric 3 hops via router fe80::846e:70a6:8ca6:48b7, table EZBMAIN

--- IPv6 RIP Packet Sent (MPCPTPV67) --- Type: Response
Destination_Addr: ::
Prefix Length: 0  metric: 16
Destination_Addr: 9::67:120:3
Prefix Length: 128  metric: 5
Destination_Addr: 1977::7
Prefix Length: 128  metric: 1
Destination_Addr: 7:7:7::
Prefix Length: 128  metric: 1
Destination_Addr: 9::67:120:7
Prefix Length: 48  metric: 1
Destination_Addr: 9::67:120:3

Prefix Length: 128 metric: 1
Destination_Addr: 1967::
  Prefix Length: 16 metric: 1
Destination_Addr: 1967::6
  Prefix Length: 128 metric: 16
  Prefix Length: 128 metric: 16
Destination_Addr: 6:6:6::
  Prefix Length: 48 metric: 1
Destination_Addr: 1946::6
  Prefix Length: 128 metric: 16
Destination_Addr: 9::67:120:4
  Prefix Length: 128 metric: 16
Destination_Addr: 1946::4
  Prefix Length: 128 metric: 16
Destination_Addr: 9::67:120:3
  Prefix Length: 128 metric: 16
Destination_Addr: 1977::7
  Prefix Length: 128 metric: 16
Destination_Addr: 7:7:7:7:7:7:7:7:
  Prefix Length: 48 metric: 1
Destination_Addr: 1946::4
  Prefix Length: 128 metric: 16
Destination_Addr: 9::67:120:4
  Prefix Length: 128 metric: 2
Destination_Addr: 1946::4
  Prefix Length: 128 metric: 2

EZZ8021I sending IPv6RIP response to address ff02::9 from
fe80::542c:ed1e:1362:4d26 in 1 packets with 16 routes

EZZ8004I response received from host fe80::846e:70a6:8ca6:48b7
-- IPv6 RIP Packet Received (MPCPTPV67) -- Type: Response
  Destination_Addr: ::
  Prefix Length: 0 metric: 10
  Destination_Addr: 1967::6
    Prefix Length: 128 metric: 1
  Destination_Addr: 1967::
    Prefix Length: 16 metric: 1
    Prefix Length: 128 metric: 1
  Destination_Addr: 6:6:6::
    Prefix Length: 48 metric: 1
  Destination_Addr: 1946::6
    Prefix Length: 128 metric: 16
  Destination_Addr: 9::67:120:4
    Prefix Length: 128 metric: 16
  Destination_Addr: 1946::4
    Prefix Length: 128 metric: 16
  Destination_Addr: 9::67:120:3
    Prefix Length: 128 metric: 16
  Destination_Addr: 1977::7
    Prefix Length: 128 metric: 16
  Destination_Addr: 7:7:7:7:7:7:7:7:
    Prefix Length: 48 metric: 1
  Destination_Addr: 1946::4
    Prefix Length: 128 metric: 16
  Destination_Addr: 9::67:120:4
    Prefix Length: 128 metric: 2
  Destination_Addr: 1946::4
    Prefix Length: 128 metric: 2

Figure 99. Sample IPv6 OMPROUTE Trace Output

Following are brief explanations of numbered items in the trace:
OMPROUTE learns of TCP/IP stack IPv6 interface addresses. Note that each home address on an IPv6 interface is described separately; OMPROUTE uses the interface name to assign addresses to a specific interface.

Direct routes are added for each non-link-local TCP/IP stack IPv6 home address. When an interface's home address is needed in a message, its link-local address is used unless it is a VIPA that does not have a link-local address.

IPv6 RIP Requests and Responses begin being sent out IPv6 RIP interface. Note use of link-local address when interface is being identified by address only.

IPv6 RIP Response received and associated routes added to IPv6 route table. Note that source address is always link-local.

Request received to change IPv6 tracing level to 2 (adds formatted packets). The operator command to set the tracing level appears in the IPv4 trace, because modify commands run on the IPv4 thread.

Formatted IPv6 RIP packet.

TCP/IP services component trace for OMPROUTE

z/OS Communications Server provides Component Trace support for the OMPROUTE application. This section describes how to specify OMPROUTE trace and formatting options. For short descriptions of other tracing procedures, such as displaying trace status, see Chapter 5, “TCP/IP services traces and IPCS support,” on page 47. Also, see “Commands to enable, disable, and display the status of the OMPROUTE CTRACE” on page 776.

For detailed descriptions, refer to the following information:
- [z/OS MVS Diagnosis: Tools and Service Aids](#) for information about Component Trace procedures
- [z/OS MVS Initialization and Tuning Reference](#) for information about the SYS1.PARMLIB member
- [z/OS MVS System Commands](#) for information about trace commands
- [z/OS MVS Programming: Authorized Assembler Services Guide](#) for information about procedures and return codes for CTRACE macros
- [z/OS MVS IPCS Commands](#)
- [z/OS MVS IPCS User's Guide](#)

Specifying trace options

You can specify Component Trace options at OMPROUTE initialization or after OMPROUTE has initialized.

Specifying options at initialization

A default minimum Component Trace is always started during OMPROUTE initialization. A parmlib member can be used to customize the parameters used to initialize the trace. The default OMPROUTE Component Trace parmlib member is the SYS1.PARMLIB member CTIORA00. The parmlib member name can be changed by use of the OMPROUTE_CTRACE_MEMBER environment variable.

Tip: Besides specifying the trace options, you can also change the OMPROUTE trace buffer size. The buffer size can be changed only at OMPROUTE initialization.
The maximum OMPROUTE trace buffer size is 100 MB.

**Guideline:** Use of a large internal CTRACE buffer or an external writer is recommended when using the DEBUGTRC option.

**Requirement:** The OMPROUTE REGION size in the OMPROUTE catalog procedure must be large enough to accommodate a large buffer size.

If the CTIORA00 member is not found when starting OMPROUTE, the following message is issued:

```
IEE5381 CTIORA00 MEMBER NOT FOUND in SYS1.PARMLIB
```

When this occurs, the OMPROUTE component trace is started with a buffer size of 1 MB and the MINIMUM tracing option.

The following figure shows the SYS1.PARMLIB member CTIORA00.
DESCRIPTION = This parmlib member causes component trace for
the TCP/IP OMPROUTE application to be initialized
with a trace buffer size of 1M.

This parmlib member only lists those TRACEOPTS
values specific to OMPROUTE. For a complete list
of TRACEOPTS keywords and their values see
z/OS MVS INITIALIZATION AND TUNING REFERENCE.

$MAC(CTIORA00),COMP(OSPF ),PROD(TCPIP ): Component Trace
SYS1.PARMLIB member

*******************************************************************************/

TRACEOPTS
  /* Optionally start external writer in this file (use both */
  /* WTRSTART and WTR with same wtr_procedure) */
  /* WTRSTART(wtr_procedure) */
  /* ON OR OFF: PICK 1 */
  ON
  /* OFF */
  /* BUFSIZE: A VALUE IN RANGE 128K TO 100M */
  /* CTRACE buffers reside in OMPROUTE Private storage */
  /* which is in the regions address space. */
  /* BUFSIZE(IM) */
  WTR(wtr_procedure)
  /* OPTIONS: NAMES OF FUNCTIONS TO BE TRACED, OR "ALL" */
  OPTIONS('ALL ')

Figure 100. SYS1.PARMLIB member CTIORA00 (Part 1 of 2)
Table 79 describes the available trace options.

<table>
<thead>
<tr>
<th>Trace event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Select all types of records. Be aware that this option slows performance.</td>
</tr>
<tr>
<td>MINIMUM</td>
<td>Select OMPROUTE’s minimum level of tracing. Specifying MINIMUM is the same as specifying ROUTE.</td>
</tr>
<tr>
<td>ROUTE</td>
<td>Select information exchange and routing updates between the OMPROUTE application and the z/OS TCP/IP Services stack.</td>
</tr>
<tr>
<td>PACKET</td>
<td>Select all inbound and outbound packet flows. This is the same as specifying OPACKET, RPACKET, and IPACKET.</td>
</tr>
<tr>
<td>RPACKET</td>
<td>Select inbound and outbound packet flows for the IPv4 RIP and IPv6 RIP protocols.</td>
</tr>
<tr>
<td>OPACKET</td>
<td>Select inbound and outbound packet flows for the IPv4 OSPF and IPv6 OSPF protocols.</td>
</tr>
<tr>
<td>IPACKET</td>
<td>Select inbound packets sent from z/OS TCP/IP with information regarding route or interface changes.</td>
</tr>
<tr>
<td>SPACKET</td>
<td>Trace inbound and outbound packets sent between the SNMP agent and the OMPROUTE subagent.</td>
</tr>
<tr>
<td>DEBUGTRC</td>
<td>Redirects IPv4 trace (-t), IPv4 debug (-d), IPv6 trace (-6t) and IPv6 debug (-6d) output to the CTRACE facility.</td>
</tr>
</tbody>
</table>

**Guideline:** Use of a large internal CTRACE buffer or an external writer is recommended when using the DEBUGTRC option.

**Specifying options after initialization:** After OMPROUTE initialization, you must use the TRACE CT command to change the component trace options. Each time a new Component Trace is initiated, all prior trace options are turned OFF and the new options are put into effect.

You can specify the trace options with or without the PARMLIB member. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47.
Formatting OMPROUTE trace records

You can format component trace records using IPCS panels or a combination of the IPCS panels and the CTRACE command, either from a dump or from external-writer files. (For details, see Chapter 5, “TCP/IP services traces and IPCS support,” on page 47.) Any combination of the following values can be entered as options to filter the CTRACE entries. The options must be entered using the format:

```
TYPE(option[,option]...)
```

- ROUTE
- OPACKET
- RPACKET
- IPACKET
- SPACKET
- DEBUGTRC

You cannot use the following as options when formatting OMPROUTE component traces:

- ALL
- MINIMUM
- PACKET

Commands to enable, disable, and display the status of the OMPROUTE CTRACE

Steps for enabling the CTRACE at OMPROUTE startup

Restriction: OMPROUTE must have read access to the SYS1.PARMLIB data sets.

To enable the CTRACE at OMPROUTE startup:

1. Edit CTIORA00 parmlib member (or the member specified in OMPROUTE_CTRACE_MEMBER environment variable) and specify TRACEOPTS ON, the desired buffer size by way of the BUFSIZE() parameter, and the desired CTRACE options. To direct the CTRACE to an external writer, also specify the name of the writer JCL procedure in the WTR() parameter. Refer to the example CTIORA00 member.

2. Start OMPROUTE with a trace level enabled.

Steps for disabling the CTRACE at OMPROUTE startup

To disable the CTRACE at OMPROUTE startup, edit CTIORA00 or the member specified in OMPROUTE_CTRACE_MEMBER environment variable and specify TRACEOPTS OFF.

Steps for enabling the CTRACE after OMPROUTE has started

To enable the CTRACE after OMPROUTE has started:

1. Do one of the following:
   - Issue the following console commands to enable a CTRACE to an internal buffer:
2. If DEBUGTRC or ALL is included in the CTRACE options, issue one of the following commands to modify the trace level:

```
F,omproute_jobname,TRACE=x
F,omproute_jobname,DEBUG=x
F,omproute_jobname,TRACE6=x
```

or

```
F,omproute_jobname,DEBUG6=xx
```

**Requirement:** This is required even if the OMPROUTE trace is already active.

---

### Steps for disabling the CTRACE after OMPROUTE has started

To disable the CTRACE after OMPROUTE has started:

1. Issue the following console commands to disable a CTRACE to an internal buffer:

   ```
   TRACE CT,OFF,COMP=SYSTCPRT,SUB=(omproute_jobname)
   ```

   or

   Issue the following console commands to disable a CTRACE to an external writer:

   ```
   TRACE CT,OFF,COMP=SYSTCPRT,SUB=(omproute_jobname)
   TRACE CT,WTRSTOP=writer_proc
   ```

2. If DEBUGTRC or ALL is included in the CTRACE options, issue one of the following commands to modify the trace level:

   ```
   F,omproute_jobname,TRACE=x
   F,omproute_jobname,DEBUG=x
   F,omproute_jobname,TRACE6=x
   ```

   or

   ```
   F,omproute_jobname,DEBUG6=xx
   ```

---

### Step for displaying the CTRACE status

To display the CTRACE status, issue the following console command:

```
D TRACE,COMP=SYSTCPRT,SUB=(omproute_jobname)
```
Chapter 33. Diagnosing NCPROUTE problems

The NCPROUTE protocol provides a standardized interface, through which a server program on one host (NCPROUTE) can manage the routing tables and respond to SNMP route table requests for another program (Network Control Program).

This topic contains the following sections:
- “Definitions” on page 782
- “Diagnosing NCPROUTE problems” on page 783
- “NCPROUTE traces” on page 792

Figure 101 shows the NCPROUTE environment.
Prior to ACF/NCP V7R1, static route tables were used for routing IP datagrams over connected networks. However, the static routes had a drawback in that they were not able to respond to network topology changes. By implementing the RIP protocol between a host and NCP clients, the NCPROUTE server is able to provide dynamic IP routing for NCP clients. In effect, the NCP clients become active RIP routers in a TCP/IP network.

Multiple NCP units (374x family of communications controllers) can connect to the same NCPROUTE server on one host. This means that NCPROUTE can manage multiple routing tables for each NCP client. SNALINK is used as the connection vehicle to establish LU0 sessions between NCPROUTE and NCP clients. Each NCP client can have one or more LU0 sessions with NCPROUTE, provided that one session is used as primary and others as secondary for backup.

The NCPROUTE server reacts to network topology changes on behalf of NCP clients by maintaining each NCP client routing table, processing and generating RIP and SNMP datagrams, and performing error recovery procedures.

The NCPROUTE protocol is based on the exchange of protocol data units (PDUs).

The following list describes the eight types of PDUs:

- **Hello PDU**: Sent from an NCP client to initiate a session with NCPROUTE.
- **Acknowledge PDU**: Sent from NCPROUTE to acknowledge receipt of a Hello datagram. NCPROUTE is ready to manage the routing tables for an NCP client.
- **Status PDU**: Sent from an NCP client to inform NCPROUTE of a status change with an interface. Interfaces can become inactive or active.
- **Delete Route Request PDU**: Sent from NCPROUTE to request deletion of a route that is no longer known to the network from an NCP client routing table. This PDU can also be sent from an NCP client as a response informing NCPROUTE that the delete route request failed.
- **Add Route Request PDU**: Sent from NCPROUTE to request addition of a route that is discovered by NCPROUTE to an NCP client routing table. This PDU can also be sent from an NCP client as a response informing NCPROUTE that the add route request failed.
- **Change Route Request PDU**: Sent from NCPROUTE to request changing the value of a metric for a route currently active in an NCP client routing table.
- **Transport PDU**: Sent from an NCP client to request NCPROUTE to retransmit RIP broadcasts sent from other routers and to process Simple Network Management Protocol (SNMP) requests sent from SNMP clients in the network. This PDU can also be sent from NCPROUTE as a response to retransmit RIP broadcasts or as a response to an SNMP query request. The Transport PDU contains encapsulated RIP and SNMP commands for additional processing.
- **Inactive Interface List PDU**: Sent from an NCP client to inform NCPROUTE of currently inactive interfaces.

NCPROUTE uses the RIP messages for retransmitting of and responding to RIP updates and trace requests. A message might be unicasted, broadcasted, or multicasted, depending on the network interface capabilities in an NCP client.

There are four types of RIP messages that can be encapsulated in a Transport PDU. They are listed in Table 80 on page 781.
Table 80. Types of RIP messages

<table>
<thead>
<tr>
<th>Message type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>There are two types of Request messages:</td>
</tr>
<tr>
<td></td>
<td><strong>REQUEST TO</strong></td>
</tr>
<tr>
<td></td>
<td>Sent from NCP by NCPROUTE over a network interface to request routing table from one or more neighboring RIP routers.</td>
</tr>
<tr>
<td></td>
<td><strong>REQUEST FROM</strong></td>
</tr>
<tr>
<td></td>
<td>NCP received from one or more neighboring RIP routers as a request to transmit all or part of this NCP's routing table as supplied by NCPROUTE.</td>
</tr>
<tr>
<td>Response</td>
<td>There are two types of Response messages:</td>
</tr>
<tr>
<td></td>
<td><strong>RESPONSE TO</strong></td>
</tr>
<tr>
<td></td>
<td>Sent from NCP by NCPROUTE as a response to a request from a neighboring RIP router or sent from NCP by NCPROUTE for advertisements of RIP updates at periodic intervals over a network interface. The message contains all or part of this NCP's routing table as supplied by NCPROUTE.</td>
</tr>
<tr>
<td></td>
<td><strong>RESPONSE FROM</strong></td>
</tr>
<tr>
<td></td>
<td>NCP received from a neighboring RIP router as a response to a request from NCP by NCPROUTE or received from one or more neighboring RIP router for advertisements of RIP updates at periodic intervals. The message contains all or part of a neighboring router's routing table.</td>
</tr>
<tr>
<td>TraceOn</td>
<td>NCP received a request from a neighboring RIP router to enable the actions trace provided by NCPROUTE.</td>
</tr>
<tr>
<td>TraceOff</td>
<td>NCP received a request to a neighboring RIP router to disable tracing provided by NCPROUTE.</td>
</tr>
</tbody>
</table>

NCPROUTE communicates with the SNMP agent over the Distributed Program Interface (DPI) to process the SNMP commands. In this configuration, NCPROUTE becomes the SNMP subagent to provide values of registered MIB variables to the SNMP agent.

There are four types of SNMP commands that can be encapsulated within a Transport PDU:

- **Get Request**: NCP received a request from a client to obtain one or more MIB variable values from an SNMP agent.
- **Get Next Request**: NCP received a request from a client to obtain the next variable value in the MIB tree from an SNMP agent.
- **Get Response**: Sent from NCP to its client as a response to an SNMP request.
- **Set Request**: NCP received a request from a client to set or change the value of one or more MIB variables in an SNMP agent. This command is not supported by NCPROUTE.

Refer to [z/OS Communications Server: IP User’s Guide and Commands](#) for detailed information about the SNMP commands.

Table 81 describes the MIB variables registered for use by NCPROUTE:

<table>
<thead>
<tr>
<th>MIB variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipRouteDest</td>
<td>Destination IP address of this route</td>
</tr>
<tr>
<td>ipRouteMetric1</td>
<td>Primary routing metric for this route</td>
</tr>
<tr>
<td>ipRouteMetric2</td>
<td>Alternative routing metric for this route</td>
</tr>
<tr>
<td>ipRouteMetric3</td>
<td>Another alternative routing metric for this route</td>
</tr>
<tr>
<td>ipRouteMetric4</td>
<td>Another alternative routing metric for this route</td>
</tr>
<tr>
<td>ipRouteNextHop</td>
<td>IP address of the next hop of this route</td>
</tr>
<tr>
<td>ipRouteType</td>
<td>Type of route</td>
</tr>
<tr>
<td>ipRouteProto</td>
<td>Routing mechanism by which this route was learned</td>
</tr>
<tr>
<td>ipRouteMask</td>
<td>Mask value for this route</td>
</tr>
</tbody>
</table>

Refer to [z/OS Communications Server: IP User’s Guide and Commands](#) for detailed information about the MIB variables.

**Definitions**

NCPROUTE must be defined correctly to both NCP and TCP/IP. UDP port 580 must be reserved for NCPROUTE. Routes to the NCP clients must be defined on the GATEWAY or the BSDROUTINGPARMS statement for NCPROUTE connectivity.

Refer to [z/OS Communications Server: IP Configuration Reference](#) for detailed information about TCP/IP and NCPROUTE server definitions.

Internet interfaces (token ring and Ethernet) and NCST logical units for communication with the TCP/IP host must be defined for each NCP client through NCP generation.

**Guideline**: If you use SNMP to query routing information of NCP clients, the SNMP query engine and agent must be configured correctly. For NCPROUTE to communicate with the SNMP agent, the MVS host name or IP address and community name must be defined in the NCPROUTE profile, SEZAINST(NCPRPROF). The SNMP agent community name must also be defined in the hlq.PW.SRC data set for proper verification.

Refer to [z/OS Communications Server: IP Configuration Reference](#) for detailed information about SNMP definitions.
Diagnosing NCPROUTE problems

Problems with NCPROUTE are generally reported under one of the following categories:
- Abends
- Connection problems
- Analyzing routing failures
- Incorrect output
- Session outages

Use the information provided in the following sections for problem determination and diagnosis of errors reported against NCPROUTE.

Abends

An abend during NCPROUTE processing should result in messages and error related information being sent to the system console. A dump of the error is needed unless the symptoms match a known problem.

Documentation

Code a SYSUDUMP DD or SYSABEND DD statement in the cataloged procedure used to start NCPROUTE to ensure that a useful dump is obtained in event of an abend.

Analysis

Refer to z/OS Problem Management or to Chapter 3, “Diagnosing abends, loops, and hangs,” on page 25 for information about debugging dumps produced during NCPROUTE processing.

Connection problems

NCPROUTE connection problems are reported when NCPROUTE is unable to connect to TCP/IP, when NCP clients are unable to connect to the NCPROUTE server, when SNALINK LU0 is unable to connect between the NCPROUTE server and an NCP client, and when NCPROUTE is unable to connect to an SNMP agent. Generally, this type of problem is caused by an error in the configuration or definitions (either in VTAM, TCPIP, SNALINK, SNMP, NCP, or NCPROUTE).

Documentation

The following documentation should be available for initial diagnosis of NCPROUTE connection problems:
- Documentation for NCPROUTE connection failure
  - TCP/IP console log
  - hlq.PROFILE.TCPIP data set
  - TCPIP.DATA data set
  - NCPROUTE cataloged procedure
- Documentation for NCP client connection failure
  - NCPROUTE console log
  - NCPROUTE.PROFILE data set
  - NCP client network definitions data set (NCP generation)
- Documentation for SNALINK LU0 connection failure
  - SNALINK LU0 console log
  - VTAM APPL definitions for SNALINK LU0s
- Documentation for SNMP agent problems
  - SNMP console logs for SNMP agent and client
  - hlq.MIBDESC.DATA data set
  - hlq.PW.SRC data set
More documentation that might be needed is discussed in the analysis section.

**Analysis**

Table 82 shows symptoms of connection problems and refers to the steps needed for initial diagnosis of the error.

*Table 82. NCPROUTE connection problems*

<table>
<thead>
<tr>
<th>Connection Problem</th>
<th>Analysis Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCP client connection failure</td>
<td>1, 2, 7, 8, 10, 13</td>
</tr>
<tr>
<td>NCPROUTE connection failure</td>
<td>1, 3, 5, 6, 7, 8, 10, 11, 13</td>
</tr>
<tr>
<td>SNALINK LU0 connection failure</td>
<td>1, 3, 7, 8, 10, 12</td>
</tr>
<tr>
<td>SNMP Agent connection failure</td>
<td>4, 9, 10, 13</td>
</tr>
</tbody>
</table>

“Steps for NCPROUTE connection problems” gives the diagnostic steps referred to in Table 82.

For TCP/IP configuration-related problems, refer to the *z/OS Communications Server: IP Configuration Reference* for more information.

**Steps for NCPROUTE connection problems:** Perform the following steps to diagnose NCPROUTE connection problems.

1. For an NCP client, make sure that the internet interfaces (token ring and Ethernet) and NCST logical units for communication with the TCP/IP host are defined correctly in an NCP generation. Refer to the *ACF/NCP IP Router Planning and Installation Guide* for detailed information about NCP definitions.
   a. Make sure that the NCPROUTE UDP port (UDPPORT keyword), coded on the IPOWNER statement in an NCP generation, matches the value defined in the .ETC.SERVICES data set. If it is not coded, the value used is the default UDP port 580.
   b. Verify that the assigned port numbers and service names for NCPROUTE and the router are correct. Also make sure that the router service port 520 is defined in the .ETC.SERVICES data set. The NCP clients use this port as a destination port when broadcasting RIP packets to adjacent routers.
   c. Make sure that NCST logical units for the SNALINK LU0s are defined correctly. A partner LU name (INTERFACE keyword) for the SNALINK-NCST interface, coded on the LU statement in an NCST GROUP of an NCP generation, should match the LU name in a SNALINK LU0 DEVICE statement in the hlq.PROFILE.TCPIP data set.
   d. Make sure that the remote LU name (REMLU keyword) for the SNALINK-NCST interface, coded on the LU statement in an NCP generation, matches the VTAM application name in the VTAM APPL definitions for SNALINK LU0s. For more information about SNALINK configuration and VTAM APPL definitions, refer to the *z/OS Communications Server: IP Configuration Guide*.
   e. Make sure that the NCST partner LU name (INTERFACE keyword) for the SNALINK-NCST interface, coded on the IPOWNER and IPLOCAL statements in an NCP generation, matches the partner LU name in Step 1b.
f. Make sure that the IP address for the TCP/IP host (HOSTADDR keyword), coded on the IPOWNER statement in an NCP generation, matches the IP address for the SNALINK LU0 device name coded on the HOME statement in the hlq.PROFILE.TCPIP data set.

g. Make sure that the IP address for the SNALINK-NCST interface (LADDR keyword), coded on a IPLOCAL statement in an NCP generation, matches the IP address for the SNALINK LU0 link name coded on the GATEWAY statement in the hlq.PROFILE.TCPIP data set.

h. Make sure that the destination IP address for the SNALINK-NCST interface (P2PDEST keyword), coded on a IPLOCAL statement in an NCP generation, matches the IP address on the IPOWNER statement in Step 1e.

i. Make sure that IPLOCAL statements are defined for the directly-attached NCP internet interfaces (token ring and Ethernet) in an NCP generation. Verify the correctness of the IP addresses (LADDR keyword), metric values (METRIC keyword), protocol type (PROTOCOL keyword), and subnetwork masks (SNETMASK keyword).

2. Make sure that the appropriate NCP LOADLIB is used and that it contains correct network definitions. The NCP LOADLIB must be in the search list referred to by the //DD STEPLIB statement. Verify that a 374x communications controller to be in the session with NCPROUTE is loaded with the correct NCP load module.

3. Make sure that appropriate cataloged procedures for NCPROUTE (NCPROUT) and SNALINK (SNALPROC) are used, and verify the correctness of the data set references.
   • For the SNALINK cataloged procedure, make sure that the number of SNALINK sessions is large enough to allow multiple NCP sessions with NCPROUTE. This number is referred to by the MAXSESS keyword on the EXEC statement.

4. If using SNMP, make sure that the appropriate cataloged procedure for the SNMP agent (SNMPD) is used and verify the correctness of the data set references. Do likewise for a SNMP client (SNMPQE on MVS host).

5. Make sure that NCPROUTE is configured correctly in the hlq.PROFILE.TCPIP data set. The cataloged procedure name (NCPROUT) is referred to on AUTOLOG (optional), and PORT statements. UDP port 580 must be reserved for NCPROUTE.

6. Make sure that NCPROUTE is configured correctly in the ETC.SERVICES data set. See also Step 1a.

7. Make sure that SNALINK LU0 is configured correctly in the hlq.PROFILE.TCPIP data set. The SNALINK device name, LU name, and VTAM application address space name are referred to on the DEVICE statement. The SNALINK link name is referred to on the LINK, HOME, and GATEWAY statements. See also Steps 1b, 1c, 1e, and 1f.
· If more than one NCP client is to be in session with NCPROUTE, repeat Step 7 to configure SNALINK LU0 for another session. TCP/IP definitions must be defined for each SNALINK LU0 session. If TCP/IP is currently running and another NCP client is to be added, another SNALINK LU0 can be configured using VARY TCPIP,OBEYFILE commands. This allows TCP/IP to be reconfigured without having to shut down TCP/IP.

8. If you are using OMPROUTE, make sure that the routing parameters in the OMPROUTE configuration file (network interface definitions) and TCP/IP configuration (BSDROUTINGPARMS and BEGINROUTES or GATEWAY statements) for the NCP clients are defined correctly. In addition, verify that direct and static routes to the NCP clients are defined correctly in TCP/IP BEGINROUTES or GATEWAY statement.

9. If you are using SNMP, make sure that the SNMP agent is configured correctly in the hlq.PROFILE.TCPIP data set. If the SNMP client is on an MVS host, verify that the SNMP client address space is also configured. The cataloged procedure names, SNMPD, for the SNMP agent and client, are referred to on the AUTOLOG (optional), and PORT statements.
   · For the SNMP agent, make sure that the access authority information is defined correctly in the SEZAINST(EZBNRPRF) data set for the NCPROUTE profile, referenced in the NCPROUTE cataloged procedure.

10. If an NCP client is activated and ready to establish a session with NCPROUTE, make sure that the cataloged procedures for TCPIP, NCPROUTE, and SNALINK are all started. If you are using SNMP, make sure that the SNMP agent and client are started.
   a. Make sure that the SNALINK devices are started by the START statement in the hlq.PROFILE.TCPIP data set. The SNALINK devices can also be started by a VARY TCPIP,OBEYFILE command or a VARY TCPIP,START command.
   b. Make sure that VTAM command prompts at the system operator console are replied to; otherwise, a SNALINK session can be in a pending activation state.
   c. Make sure that the NCP client physical and logical lines for the internet interfaces (token ring and Ethernet) are active.
   d. Make sure that NCST lines are active for the SNALINK LU0 sessions.
   e. Make sure that VTAM cross-domain resource managers (CDRMs) are active in the MVS hosts.

11. For network connectivity problems, see Chapter 4, “Diagnosing network connectivity problems,” on page 29.


13. For OMPROUTE problems, see Chapter 32, “Diagnosing OMPROUTE problems,” on page 749. Ensure that the interface definitions in the BSDROUTINGPARMS statement in the hlq.PROFILE.TCPIP data set match
the definitions in the corresponding interface definitions in the OMPROUTE configuration file. For more information about defining BSDROUTINGPARMS for NCPROUTE, refer to IP Configuration Reference.

Analyzing routing failures
Routing problems are usually the result of outages in a network and there are no alternative routing paths available for recovery. They can also be the result of incorrect configurations in the channel-attached and network-attached routers, as well as incorrect ARP entries, when applicable. PING and Traceroute commands to and from a z/OS host are useful diagnosis aids for problem determination.

In this section, unless otherwise specified, the following command terms are used as described in Table 83.

Table 83. NCPROUTE command terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PING</td>
<td>Refers to z/OS UNIX oping, TSO PING, and the ping commands used on other platforms.</td>
</tr>
<tr>
<td>Traceroute</td>
<td>Refers to z/OS UNIX tracert</td>
</tr>
<tr>
<td>NETSTAT ROUTE</td>
<td>Refers to the z/OS UNIX onetstat -r, TSO NETSTAT ROUTE, and the netstat route commands used on other platforms.</td>
</tr>
<tr>
<td>NETSTAT GATE</td>
<td>Refers to the z/OS UNIX onetstat -g and TSO NETSTAT GATE commands. This command is available only on the z/OS platform.</td>
</tr>
<tr>
<td>NETSTAT ARP</td>
<td>Refers to the z/OS UNIX onetstat -R ALL, TSO NETSTAT ALL, and the netstat arp commands used on other platforms.</td>
</tr>
</tbody>
</table>

NCPROUTE routing failures are reported when a client is unable to get a positive response to a PING or Traceroute command for a remote host where there are NCPs acting as RIP servers along the routing paths.

Documentation
The following documentation should be available for initial diagnosis of routing failures:
- NCPROUTE console log
- TCP/IP console log
- hlq.PROFILE.TCPIP data set
- NCP client network definitions data set (NCP generation)
- Output from MODIFY NCPROUTE,TABLES command for a display of internal tables representing a NCP client.
- Outputs from PING and Traceroute commands.

Analysis
Table 84 on page 788 shows symptoms of PING failures and refers to the steps needed for initial diagnosis of the error.
Steps for analyzing routing failures: This section gives the diagnostic steps referred to in Table 84.

Perform the following steps to analyze routing failures.

Guideline: Because an NCP client cannot respond to Traceroute commands, you can use the PING command to diagnose routing failures. However, a Traceroute command can be used to locate a suspect router along the routing path to a remote host beyond the NCP client. In the steps below, the PING command is used for diagnosis.

1. Make sure the PING command contains a valid destination IP address for the remote host.

2. Make sure that a 374x communications controller acting as a RIP server involved in the PING transaction is active and is running with a correct level of NCP LOADLIB. Verify that correct network definitions are defined in the NCP generation and that the NCP client is in session with NCPROUTE.

3. If the PING command was issued from a remote host, issue the NETSTAT ROUTE, NETSTAT GATE (if host is z/OS), and NETSTAT ARP commands from there for its routing and ARP table information.
   a. If the local host is running with OMPROUTE, verify the routing configuration for routes and networks as defined in the OMPROUTE configuration file (network interface definitions) and TCP/IP configuration (BSDROUTINGPARMS and BEGINROUTES or GATEWAY statements). To ensure NCP connectivity with the NCP clients, verify that direct and static routes to the clients are defined correctly.
   b. If there are any problems with the routes and networks, see “Using the Netstat command” on page 41.

4. If the remote host is running with OMPROUTE, verify its routing configuration for routes and networks as defined in its OMPROUTE configuration file and TCP/IP configuration. See Step 3a for configuration information.
   a. For routers or hosts running on platforms other than z/OS, refer to their documentation for more information on correcting routing problems. Also, refer to these documentations for NETSTAT commands to display the routing and ARP tables for problem determination.

5. If there are no problems with the routes or networks, check for broken or poorly connected cables between the client and the remote host. This includes checking the IP interfaces (token ring and Ethernet) on the 374x communications controller.
6. Make sure there is a channel connection between the 374x communications controller and the MVS host. A channel connection can be interrupted by an Automatic Network Shutdown (ANS) situation. ANS can occur when the system operator puts the MVS console into CP mode. In this case, the system operator needs to return to MVS from CP to recover from ANS.

7. For more information about diagnosing network connectivity problems, refer to Chapter 4, “Diagnosing network connectivity problems,” on page 29.

8. For more information about diagnosing PING problems, refer to “Using the Ping command” on page 37.

9. For more information about diagnosing PING timeouts, refer to “Correcting timeout problems” on page 41.

“Steps for analyzing routing failures” on page 788 gives the diagnostic steps referred to in Table 84 on page 788

Incorrect output

Problems with incorrect output are reported when the data sent to the client is in an unexpected form (for example, incorrect TCP/IP output, incorrect SNALINK LU0 output, invalid RIP commands, incorrect RIP broadcasting information, incorrect routing-table updates, truncated packets, or incorrect SNMP agent or client output).

Documentation

The following documentation should be available for initial diagnosis of incorrect output:

- NCPROUTE cataloged procedure
- Documentation for NCPROUTE incorrect output
  - NCPROUTE console log
  - NCPROUTE.PROFILE data set
  - NCP client network definitions data set (NCP generation)
  - Output from MODIFY NCPROUTE, TABLES command for a display of internal tables (routes, interfaces, and filters) in NCPROUTE used for an NCP client.
- Documentation for TCP/IP incorrect output
  - TCP/IP console log
  - hlq.TCPIP.PROFILE data set
  - TCPIP.DATA data set
- Documentation for SNMP agent incorrect output
  - SNMP console logs for SNMP agent and client
  - hlq.MIBDESC.DATA data set
  - hlq
  - hlq.PW.SRC data set
  - NetView log (if SNMP client is on an MVS host)

Analysis

Table 85 on page 790 shows types of incorrect output and refers to the steps needed for initial diagnosis of the error.
Table 85. NCPROUTE incorrect output

<table>
<thead>
<tr>
<th>Incorrect output</th>
<th>Analysis steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/IP incorrect output</td>
<td>1</td>
</tr>
<tr>
<td>SNALINK LU0 incorrect output</td>
<td>2</td>
</tr>
<tr>
<td>NCPROUTE incorrect output</td>
<td>3</td>
</tr>
<tr>
<td>SNMP agent or client incorrect output</td>
<td>4</td>
</tr>
</tbody>
</table>

Steps for diagnosing incorrect output: This section gives the diagnostic steps referred to in Table 85.

Perform the following steps to diagnose incorrect output.

1. If the TCP/IP console shows a message, refer to z/OS Communications Server: IP Messages Volume 2 (EZB, EZD) and follow the directions for system programmer response for the message.
   a. Information in the TCP/IP console log should contain a detailed description of the error.
   b. In the event of TCP/IP loops or hangs, see Chapter 3, “Diagnosing abends, loops, and hangs,” on page 25.

2. If the SNALINK LU0 console shows a SNALINK error, refer to the explanation of the corresponding error message as described in the z/OS Communications Server: IP Messages Volume 1 (EZA) or z/OS Communications Server: SNA Messages. For more information on diagnosing SNALINK LU0 session outages, see Chapter 19, “Diagnosing SNALINK LU0 problems,” on page 547.

3. If the NCPROUTE console shows a message, refer to z/OS Communications Server: IP Messages Volume 2 (EZB, EZD) and follow the directions for system programmer response for the message.

4. If the SNMP agent or client console shows a message, refer to z/OS Communications Server: IP Messages Volume 2 (EZB, EZD) and follow the directions for system programmer response for the message.

5. For more information about diagnosing SNMP problems, see Chapter 25, “Diagnosing Simple Network Management Protocol (SNMP) problems,” on page 605.

Session outages
Session outages are reported as an unexpected termination of the TCP/IP connection, the SNALINK LU0 task, the NCPROUTE-to-NCP client session, or the NCPROUTE-to-SNMP agent connection. A session that has been disconnected or ended results in NCPROUTE being returned to the initial state of waiting for Hello PDUs and SNMP requests from an NCP client.
**Documentation**

The following documentation should be available for initial diagnosis of session outages:

- Documentation for TCP/IP session outage
  - TCP/IP console log
- Documentation for SNALINK LU0 session outage
  - SNALINK LU0 console log
  - VTAM console log
- Documentation for NCPROUTE-to-NCP client session outage
  - NCPROUTE cataloged procedure
  - NCPROUTE console log
  - NCP client network definitions data set (NCP generation)
- Documentation for NCPROUTE-to-SNMP agent session outage
  - SNMP console log for SNMP agent
  - NetView log (if the SNMP client is on the MVS host)

**Analysis**

Table 86 shows symptoms of session outages and refers to the steps needed for initial diagnosis of the error.

**Table 86. Symptoms of session outages**

<table>
<thead>
<tr>
<th>If this is the outage type</th>
<th>Then perform these steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/IP session outage</td>
<td>If the TCP/IP console shows a TCP/IP error message, refer to <a href="https://www.ibm.com/support/knowledgecenter/SS1STG_2.2.0/com.ibm.zos.doc/zos_ipmsgv2.html">z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</a> and follow the directions for system programmer response for the message.</td>
</tr>
<tr>
<td></td>
<td>If TCP/IP abended, see Chapter 3, “Diagnosing abends, loops, and hangs,” on page 25.</td>
</tr>
<tr>
<td>SNALINK LU0 session outage</td>
<td>If the SNALINK LU0 console shows a SNALINK error, refer to the explanation of the corresponding error message as described in the <a href="https://www.ibm.com/support/knowledgecenter/SS1STG_2.2.0/com.ibm.zos.doc/ipmsgv1.html">z/OS Communications Server: IP Messages Volume 1 (EZA)</a> or <a href="https://www.ibm.com/support/knowledgecenter/SS1STG_2.2.0/com.ibm.zos.doc/snmessages.html">z/OS Communications Server: SNA Messages</a>.</td>
</tr>
<tr>
<td></td>
<td>For more information on diagnosing SNALINK LU0 session outages, see Chapter 19, “Diagnosing SNALINK LU0 problems,” on page 547.</td>
</tr>
<tr>
<td>NCPROUTE-to-NCP client session outage</td>
<td>If the NCPROUTE console shows an NCPROUTE error message, refer to <a href="https://www.ibm.com/support/knowledgecenter/SS1STG_2.2.0/com.ibm.zos.doc/zos_ipmsgv2.html">z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</a> and follow the directions for system programmer response for the message.</td>
</tr>
<tr>
<td>NCPROUTE-to-SNMP agent session outage</td>
<td>If the SNMP agent console shows a SNMP error message, refer to <a href="https://www.ibm.com/support/knowledgecenter/SS1STG_2.2.0/com.ibm.zos.doc/zos_ipmsgv2.html">z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</a> and follow the directions for system programmer response for the message.</td>
</tr>
<tr>
<td></td>
<td>For more information about diagnosing SNMP problems, see Chapter 25, “Diagnosing Simple Network Management Protocol (SNMP) problems,” on page 605.</td>
</tr>
</tbody>
</table>

You can now perform the steps for the decision you have made.
NCPROUTE traces

There are many TCP/IP traces that can be useful in identifying the cause of NCPROUTE problems. This section discusses the NCPROUTE traces.

Guideline: NCPROUTE trace output is sent to the location specified by the SYSPRINT DD statement in the NCPROUTE cataloged procedure.

Activating NCPROUTE global traces

The NCPROUTE global traces are all controlled by parameters on PARMS= in the PROC statement of the NCPROUTE cataloged procedure. (Global tracing means that all NCP clients are traced.)

For example:

```
//NCPROUTE PROC MODULE=NCPROUTE,PARMS='/-t -t'
```

Tip: These parameters are also valid when starting the NCPROUTE server with the START command.

The NCPROUTE parameters that control global tracing are:

- `-t` Activates global tracing of actions for all NCP clients.
- `-t -t` Activates global tracing of packets for all NCP clients. NCPROUTE tracing can be started and stopped using the MODIFY command. For more information, refer to the z/OS Communications Server: IP System Administrator’s Commands.
- `-tq` Deactivates tracing at all levels. This parameter suppresses tracing for all NCP clients and overrides the trace settings on the GATEWAY statements in the NCPROUTE GATEWAYS data set.
- `-dp` Activates global tracing of data packets coming in and out of NCPROUTE. The data is displayed in data format.
- `-dq` Deactivates global tracing of data packets coming in and out of NCPROUTE.

Restrictions:
- A slash (/) must precede the first parameter.
- Each parameter must be separated by a blank.
- Mixed case is allowed for the parameters.
- The parameters for the NCPROUTE procedure are case-sensitive.
- There are no third- or fourth-level global tracing options like those on the GATEWAY statements in the NCPROUTE GATEWAYS data set. The system uses the higher of the two settings for a specific NCP client.
- The data packets trace option is not available for selective tracing.

The parameters described here include only those that activate tracing. Refer to the z/OS Communications Server: IP Configuration Reference for more information about all of the NCPROUTE parameters.

Activating NCPROUTE selective traces

The NCPROUTE selective traces are all activated as trace options specified in the OPTIONS statement for an NCP client in the NCPROUTE GATEWAYS data set.
Selective tracing means a different trace level can be specified for each NCP client. To assist in problem isolation, a particular NCP client can be selected for tracing.

The keyword on the OPTIONS statement that controls selective tracing for an NCP client is **trace.level**. The value that follows this keyword indicates the trace level to be used.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Does not activate any traces.</td>
</tr>
<tr>
<td>1</td>
<td>Activates tracing of actions by the NCPROUTE server.</td>
</tr>
<tr>
<td>2</td>
<td>Activates tracing of all packets sent or received.</td>
</tr>
<tr>
<td>3</td>
<td>Activates tracing of actions, packets sent or received, and packet history. Circular trace buffers are used for each interface to record the history of all packets traced. This history is included in the trace output whenever an interface becomes inactive.</td>
</tr>
<tr>
<td>4</td>
<td>Activates tracing of actions, packets sent or received, packet history, and packet contents. The RIP network routing information is included in the trace output.</td>
</tr>
</tbody>
</table>

**Restriction:** The selective traces must be defined prior to activation of an NCP client or prior to starting the NCPROUTE cataloged procedure.

Refer to the [z/OS Communications Server: IP Configuration Reference](http://www.ibm.com) for more information about the GATEWAYS data set and the GATEWAY and OPTIONS statements.

For example, the following command would activate tracing of actions, packets sent or received, packet history, and packet contents:

```
options trace.level 4
```

**NCPROUTE trace example and explanation**

[Figure 102 on page 794](#) shows an example of an NCPROUTE trace with actions, packets, history, and contents traced. The trace was generated with trace level 4 specified in the OPTIONS statement and `PARMS='-t -t -dp'` in the PROC statement of the NCPROUTE cataloged procedure.

The trace level column does not appear in the actual trace. It was added to the example to indicate the levels of the trace for which the line is generated. For example, including: `trace.level 3` on the options statement NCP client GATEWAYS data set would result in a level 3 trace, and all of the lines indicated as trace level 1, 2, or 3 would be generated in the trace output. Lines indicated as trace level d are generated if the -dp parameter is specified.
Figure 102. NCPROUTE trace (Part 1 of 10)
Figure 102. NCPROUTE trace (Part 2 of 10)
15:30:04 EZB3894I Transport from 9.67.116.65: 44 bytes of RIP data
15:30:04 EZB4045I RESPONSE from 9.67.116.66 -> 520:

15:30:04 EZB4049I destination 9.67.116.65 metric 1

15:30:04 EZB4049I destination 9.67.112.0 metric 1

15:30:04 EZB4029I Tue Jun 28 15:30:04:
15:30:04 EZB3855I NCP_Add out to 9.67.116.65
15:30:04 EZB3855I ADD destination 9.67.112.0, router 9.67.116.66, metric 2
15:30:04 EZB4049I response to 9.67.116.66 -> 0:

15:30:20 EZB4011I client 9.67.116.65: 30 second timer expired (broadcast)

15:30:20 EZB4049I destination 9.67.116.66 metric 1

15:30:20 EZB4049I destination 10.0.0.0 metric 1

15:30:20 EZB4049I destination 9.67.112.0 metric 2

Figure 102. NCPROUTE trace (Part 3 of 10)
Figure 102. NCPROUTE trace (Part 4 of 10)
Figure 102. NCPROUTE trace (Part 5 of 10)
1 15:32:50 EZB4011I client 9.67.116.65: 30 second timer expired (broadcast)
2 15:32:50 EZB4045I RESPONSE to 9.67.116.66 -> 0:
   4 15:32:50 EZB4049I  destination 9.67.116.66 metric 1
   4 15:32:50 EZB4049I  destination 10.0.0.0 metric 1
   4 15:32:50 EZB4049I  destination 9.67.112.0 metric 16
2 15:32:50 EZB3951I client 9.67.116.65: supply 10.68.15.255 -> 0 via TR88
1 15:32:50 EZB3829I Interface TR92 not up
1 15:32:50 EZB3829I Waiting for incoming packets

1 15:32:50 EZB3829I Waiting for incoming packets
1 15:32:50 EZB4009I client 9.67.116.65: 5 minute timer expired for route to 9.67.112.0
1 15:32:50 EZB4029I Tue Jun 28 15:36:39:
2 15:36:39 EZB4030I DELETE destination 9.67.112.0, router 9.67.116.66, metric 16
   flags UP|GATEWAY state SUBNET timer 300
1 15:36:39 EZB4011I client 9.67.116.65: 30 second timer expired (broadcast)
2 15:36:39 EZB4045I RESPONSE to 9.67.116.66 -> 0:
   4 15:36:39 EZB4049I  destination 9.67.116.66 metric 1
   4 15:36:39 EZB4049I  destination 10.0.0.0 metric 1
1 15:36:39 EZB3948I Interface TR92 not up
1 15:36:39 EZB3829I Waiting for incoming packets

1 15:36:39 EZB3829I Waiting for incoming packets
1 15:36:39 EZB4009I client 9.67.116.65: 5 minute timer expired for route to 9.67.112.0
1 15:36:39 EZB4029I Tue Jun 28 15:36:39:
2 15:36:39 EZB4030I DELETE destination 9.67.112.0, router 9.67.116.66, metric 16
   flags UP|GATEWAY state SUBNET timer 300
1 15:36:39 EZB4011I client 9.67.116.65: 30 second timer expired (broadcast)
2 15:36:39 EZB4045I RESPONSE to 9.67.116.66 -> 0:
   4 15:36:39 EZB4049I  destination 9.67.116.66 metric 1
   4 15:36:39 EZB4049I  destination 10.0.0.0 metric 1
1 15:36:39 EZB3895I Transport from 9.67.116.65: 43 bytes of SNMP data
1 15:43:01 EZB3895I Transport from 9.67.116.65: 43 bytes of SNMP data

d =============== Object data (length=13)
d 0000 2b06 0102 0104 1501
d 0008 0709 4374 4207 39f8
da 0010(16)
d =============== prefix + address (length=12)
d 0000 2b06 0104 0102 0611
da 0008 0943 7441 4207 39f8
da 0010(16)

Figure 102. NCPROUTE trace (Part 6 of 10)
Figure 102. NCPROUTE trace (Part 7 of 10)
Figure 102. NCPROUTE trace (Part 8 of 10)
d =============== Sending Transport PDU to NCP client (length=84)
d 0000 0700 0000 0a44 0058
d 0008 4500 004b 0034 0000
d 0010 0411 a1e6 0a44 0058
d 0018 0a44 0001 00a1 040e
d 0020 0037 ec9f 302d 0201
d 0028 0004 0473 666d 70a2
d 0030 2202 0115 0201 0002
d 0038 0100 3017 3015 060d
d 0040 2b06 0102 0104 1901
d 0048 0709 4374 4240 0409
d 0050 4374 4100 0007 3568
d 0058(88)
1 15:43:01 EZB3829I Waiting for incoming packets
...
0 15:44:30 EZB3834I *********************************************************
0 15:44:30 EZB3890I * Recv: status from 9.67.116.65
0 15:44:30 EZB3891I * Interface: 10.68.0.88 is now inactive - TR88
0 15:44:30 EZB3834I *********************************************************
3 15:44:30 EZB4038I *** Packet history for interface TR88 ***
3 15:44:30 EZB4044I Output: trace:
3 15:44:30 EZB4045I RESPONSE to 10.68.15.255 -> 0:
3 15:44:30 EZB4049I   destination 9.67.116.66 metric 1
3 15:44:30 EZB4049I   destination 10.68.0.0 metric 1
3 15:44:30 EZB4049I   destination 9.67.112.0 metric 2
3 15:44:30 EZB4045I RESPONSE to 10.68.15.255 -> 0:
3 15:44:30 EZB4049I   destination 9.67.116.66 metric 1
3 15:44:30 EZB4049I   destination 10.68.0.0 metric 1
3 15:44:30 EZB4049I   destination 9.67.112.0 metric 2
3 15:44:30 EZB4045I RESPONSE to 10.68.15.255 -> 0:
3 15:44:30 EZB4049I   destination 9.67.116.66 metric 1
3 15:44:30 EZB4049I   destination 10.68.0.0 metric 1
3 15:44:30 EZB4044I Input: trace:
3 15:44:30 EZB4045I RESPONSE from 10.68.0.88 -> 520:
3 15:44:30 EZB4049I   destination 9.67.116.66 metric 1
3 15:44:30 EZB4049I   destination 10.68.0.0 metric 1
3 15:44:30 EZB4049I   destination 9.67.112.0 metric 2

Figure 102. NCPROUTE trace (Part 9 of 10)
The following information explains the numbered items in the trace:

1. The port number and the service name are defined as 580 and ncprout in the hlq.ETC.SERVICES data set for this NCPROUTE server.
2. NCPROUTE is processing the NCPROUTE.PROFILE definitions.
3. NCPROUTE is establishing the connection with the SNMP agent defined in NCPROUTE.PROFILE.
4. The NCP client is starting the hand-shaking process with NCPROUTE. NCPROUTE is establishing a session with the NCP client.
5. NCPROUTE received a list of inactive interfaces from the NCP client.
6. NCPROUTE is initializing its interface tables with interface information from the NCP client.
7. NCPROUTE is adding a route to its interface tables.
8. NCPROUTE is processing the NCP client GATEWAYS data set. The trace shows NCPROUTE server options and no additional gateway definitions.
9. NCPROUTE received a transport datagram from the NCP client.
10. The trace shows the contents of the datagram in hexadecimal followed by a division of the datagram into its parts (transport PDU header, IP header, UDP header, and UDP data).
11. The trace shows that the NCP client 9.67.116.65 received the broadcasted routing tables from adjacent router 9.67.116.66.
12. The UDP data in the datagram contains two routing table entries.
NCPROUTE is adding a new route to its tables from the information received in the transport datagram.

NCPROUTE is issuing a request to the NCP client to add the route to its tables.

The NCP client 30-second timer has expired, so NCPROUTE supplies its routing tables to other routers.

NCPROUTE is responding to the request by sending its routing tables to the requesting router for the NCP client.

This line shows an inactive state for interface TR92.

The NCP client 3-minute timer expired. The client was broadcast as a network unreachable route (in the range metric 16—infinte), so NCPROUTE updates its routing tables for the NCP client.

NCPROUTE is deleting the NCP client from its tables.

The NCP client five-minute timer has expired for the route to 9.67.112.0.

NCPROUTE is deleting the route to 9.67.112.0 from its tables for the NCP client.

NCPR received a transport datagram from the SNMP client through NCP client 9.67.116.65.

NCPROUTE is processing the SNMP request.

NCPROUTE has received a status notification from the NCP client. The interface TR88 has become inactive.

The packet history for the interface TR88 is included in the trace because the interface has become inactive.
Chapter 34. Diagnosing X.25 NPSI problems

This topic discusses how to diagnose X.25 NPSI problems and includes the following sections:

- “Operation” on page 806
- “Configuration requirements” on page 807
- “Sources of diagnostic information” on page 808
- “X.25 trace examples” on page 808
- “Steps for diagnosing logon problems” on page 811
- “Session hangs” on page 812

The X.25 NPSI server uses an X.25 network or point-to-point X.25 line to transfer TCP/IP traffic. The X.25 NPSI server is a VTAM application running as a started task. Either the NPSI Generalized Access to X.25 Transport Extension (GATE) or Dedicated Access to X.25 Transport Extension (DATE) can be used. GATE is recommended because it allows NPSI to handle more details of error recovery and allows an X.25 physical link to be shared with other functions.

Details of the GATE and DATE programming interfaces are in *X.25 NPSI Host Programming*, and further diagnostic information is in *X.25 NPSI Diagnosis, Customization, and Tuning*.

Specifications for carriage of IP traffic on X.25 networks can be found in:

**RFC 877**
A Standard for the Transmission of IP Datagrams Over Public Data Networks

**X25.DOC**
Old DDN X.25 specifications from BBN (available by anonymous FTP from nic.ddn.mil in directory netinfo)

**RFC 1236**
IP to X.121 Address Mapping for DDN

**RFC 1356**
Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode

Figure 103 on page 806 shows the X.25 NPSI environment.
The X.25 NPSI server uses NPSI to set up X.25 virtual circuits as needed to carry traffic to and from remote X.25 equipment. The three main functional areas shown in Figure 103 are:

- TCP/IP interface
- NPSI interface
- IP/X.25 address mapping

IP datagrams are transferred between TCP/IP and the X.25 NPSI server on an DLC path established when a TCPIP X25NPSI device is started. The transfer protocol is similar to that used with SNALINK, with the addition of a first-hop IP address passed by TCP/IP from the relevant GATEWAY entry. The X.25 NPSI server uses the first hop IP address to look up an X.25 address in its destination table.

Communication with NPSI is by way of several SNA sessions. One control session is established at initialization for each MCH LU defined in a LINK statement in the
X.25 NPSI server configuration data set. Commands to establish and terminate X.25 virtual circuit connections pass between the X.25 NPSI server and NPSI on the control session. Refer to X.25 NPSI Host Programming for details of the control commands. As new virtual circuits are established, NPSI initiates new SNA sessions with the X.25 NPSI server application by means of VTAM LOGON. IP datagrams are then exchanged with the remote equipment over the VC session until an idle timeout occurs or the VC is taken for another destination.

IP addresses are mapped to X.25 addresses by table lookup, or in the case of the DDN network, by a calculation described in RFC 1236. The X.25 NPSI server performs the lookup with the first-hop IP address on each datagram it receives from TCP/IP. The LINK and DEST entries defined in the X.25 NPSI server configuration data set are scanned in order from top to bottom to find a DEST with a matching IP address. After the DEST is found, the link it applies to is selected to carry the datagram, and the active virtual circuits on that link are scanned to find one with an X.25 address that matches the DEST. If such a VC is found, the datagram is queued for transmission on that VC; if none is found and there is a free VC, a new X.25 call is initiated; if all VCs on the link are in use, the least recently used connection is cleared, as long as it has been open for at least the minimum open time, and a new call is initiated. If no VC matches these conditions, the datagram is discarded.

**Configuration requirements**

The next two sections describe configuration considerations.

**RACF/Security Manager requirement**

The user ID assigned to the X.25 NPSI start procedure needs an OMVS Segment assigned to it.

**VTAM considerations**

- **APPL definition**
  The X.25 NPSI server requires AUTH=(ACQ) and PARSESS=YES in the VTAM APPL definition.

- **SWNET definition for switched circuits**
  - The value specified for MAXDATA for the PU must be at least 10 bytes greater than the value specified for the maximum packet size on the BUFFERS statement in the X.25 NPSI server configuration data set.
  - SSCP=USSNTO and DISCNT=(YES,F) are necessary.

**NPSI considerations**

- **BUILD definition**
  The value specified for X25.MAXPIU must be at least 10 bytes greater than the value specified for the maximum packet size on the BUFFERS statement in the X.25 NPSI server configuration data set.

- **X25.MCH definition**
  - LOGAPPL can be coded for recovery.
  - TRAN=NO is required with GATE=DEDICAT.

- **X25.VC definition**
  - Permanent virtual circuits (PVCs) are not supported.
  - Do not code LOGAPPL except with CONNECT=YES (Fast connect).
  - Do not code MAXDATA except with CONNECT=YES (Fast connect).
X.25 facilities specified with X25.OUFT are not used by the X.25 NPSI server.

**Sources of diagnostic information**

Many problems with the X.25 NPSI server are the result of configuration faults. Check the following configuration files:

- DEVICE, LINK, and GATEWAY entries in PROFILE.TCPIP
- The X.25 NPSI server configuration data set
- VTAM APPL definition for the X.25 NPSI server
- NPSI definitions
- VTAM SWNET definitions for NPSI

The primary diagnostic information source is the activity log produced by the X.25 NPSI server. Messages appear in the MVS system log, and can also be captured into a separate data set by including a SYSPRINT DD statement in the X.25 NPSI cataloged procedure. Normal logging records virtual circuit establishment and termination.

Additional information can be recorded about VC activity by setting the TRACE CONTROL option in the X.25 NPSI server configuration data set. This level is sufficient for almost all problem situations; interpretation of the data requires knowledge of X.25 NPSI packet formats. Tracing of the contents of IP datagrams sent to and received from NPSI is provided by the MVS CTRACE option. For details on using the CTRACE option, see [Chapter 5, “TCP/IP services traces and IPCS support,” on page 47](#).

VTAM buffer traces and NPSI X.25 line traces can also be useful in diagnosing difficult problem situations.

You can perform traces on the X.25 LINKNAME using the TCPIP PKTTRACE command or on the SNA LU name using the VTAM buffer trace command. See [Chapter 5, “TCP/IP services traces and IPCS support,” on page 47](#) for details about how to use the IP packet trace facility.

**X.25 trace examples**

The message severity codes (last position of the message ID) are:

- I: Information (including trace)
- W: Warning
- E: Recoverable error
- S: Recoverable error
- T: Unrecoverable error

The following example shows normal initialization:

```
EZB2111I VTAM ACB X25IP11 opened successfully
EZB2210I MCH XU038 packet level ready
EZB2451I IP AS path accepted for job name TCPIPTES
```

Initialization has four main steps:
1. The configuration file is read and processed.
2. VTAM control blocks are initialized (EZB2111I).
3. NPSI physical links (MCHs) configured by LINK statements are initialized (EZB2210I).
4. TCP/IP establishes an DLC path to the X.25 NPSI server (EZB2451I).
Normal incoming call, TRACE OFF

The following example illustrates a normal incoming call with TRACE OFF:

EZB2301I VC F001XU038 incoming call from 00000039 user data CC
EZB2325I VC F001XU038 facilities: pkt1024.
EZB2320I VC F001XU038 NPSI logon LU VL038001
EZB2330I VC F001XU038 call complete

...some time later...
EZB2350I VC F001XU038 call cleared, cause=00 diagnostic=C5
EZB2351I VC F001XU038 connection terminated for 00000039: sent 1 received
1 dropped 0
EZB2352I VC 010 closed

Notes:
1. The VC identifier F001XU038 ties together the events associated with a single virtual circuit. Messages for one VC are usually intermixed with messages for other VCs.
2. The X.25 address originating the call (00000039) is reported in the EZB2301I message.
3. X.25 calls can optionally request facilities to be applied, such as window size, packet size, throughput class, and reverse charging. These are reported in the EZB2325I message.
4. EZB2330I “call complete” indicates the virtual circuit is ready for transferring TCP/IP data.
5. An X.25 call can be closed by the originator, the acceptor, or the X.25 network. The cause and diagnostic codes in the EZB2350I message indicate the reason. In the example, cause=00 indicates the originator has closed the connection. Lists of cause and diagnostic codes can be found in X.25 NPSI Diagnosis, Customization, and Tuning.
6. EZB2351I reports the number of IP datagrams transferred on the virtual circuit.
7. After the EZB2352I “closed” message is issued, the virtual circuit is ready for reuse by another incoming call or to originate a new call.

Normal incoming call, TRACE DATA

The following example illustrates a normal incoming call with TRACE DATA:

EZB2230I MCH XU038 packet received (length=17)
EZB2000I 0000 .0.h......... 0BF00188 00000038 00000039 03420A0A
EZB2000I 0010 . CC
EZB2301I VC F001XU038 incoming call from 00000039 user data CC
EZB2325I VC F001XU038 facilities: pkt1024.
EZB2330I VC F001XU038 call accept packet sent (length=6)
EZB2000I 0000 0... 0FF0102 0400
EZB2320I VC F001XU038 NPSI logon LU VL038001
EZB2330I VC F001XU038 call complete

EZB2332I VC F001XU038 data received (length=276)
EZB2000I 0000 E.......<.....} 45000114 00100000 3C017F82 820FFD26
EZB2000I 0010 ..k.:wxr-(& 820FFD11 0000AA68 00BF778 72ADA8BE
EZB2000I 0020 0).f9kq...PF:.n 3O7OC66 96BF118 AC085046 3B83DF6E
....data omitted for brevity...
EZB2000I 0110 =_3. BD5F339D
EZB2331I VC F001XU038 data sent (length=277)
EZB2000I 0000 .E.......<.....} 04500014 14001000 03C017F 82820FFD
EZB2000I 0010 ...5;2k.:wxr-( 11820FFD 260000B2 6B00BAF7 782ADA8B
EZB2000I 0020 .).f9kq...PF:. 8E307DC 6B968F1 18AC0850 46383DF6
....data omitted for brevity...
EZB2000I 0110 ^=3. 5FD5F33 9D
EZB2336I VC F001XU038 inactivity timer expired
EZB2351I VC F001XU038 clear request packet sent (length=5)
TRACE DATA can be used to record the full contents of IP datagrams as they pass through the X.25 NPSI server. The IP header begins at byte 45 (X'2D') within the IP packet. A reduced trace given by TRACE CONTROL shows only the X.25 control packets (call request, call accept, clear request, and clear confirm). Refer to X.25 NPSI Host Programming for the detailed packet formats.

**Normal outgoing call, TRACE CONTROL**

The following example illustrates a normal outgoing call with TRACE CONTROL:

```
EZB2310I VC F810XU038 outgoing call to 00000039
EZB2311I VC F810XU038 call request packet sent (length=20)
EZB2000I 0000 ......h.......... 00011300 00
EZB2230I MCH XU038 packet received (length=5)
EZB2000I 0010 .... 420A0ACC
EZB2231I MCH XU038 call accepted by user data
EZB2320I VC 0810XU038 NPSI logon LU VL038001
EZB2330I VC 0810XU038 call complete
EZB2336I VC 0810XU038 inactivity timer expired
EZB2333I VC 0810XU038 packet received (length=1)
EZB2351I VC 0810XU038 clear request packet sent (length=5)
EZB2358I VC 0810XU038 clear sent
EZB2351I VC 0810XU038 packet received (length=1)

The steps involved in outgoing and incoming calls are similar. One important difference is that the virtual circuit identifier changes when the call is accepted (compare the EZB2311I and EZB2314I messages). This is related to the details of the NPSI programming interface.

X.25 experts should note that some X.25 packets do not appear in the trace because they are generated by NPSI without the direct involvement of the host application. Clear confirm is one example. Also, the sequence of events during closing can vary slightly in normal operation, and in some instances, benign VTAM request failures can be reported with message EZB2411E.

**Results of LIST command**

The following example illustrates the results of the LIST command:

```
EZB2020R MCH XU038 state 1050
EZB2021R VC 010 LU VL038001 DTE 00000039 state 4050
EZB2021R VC 00F LU DTE state 1010
EZB2022R IP AS TCPIPTES state 80

The LIST command is useful to get a snapshot of virtual circuit status. This example shows a normal status with one active VC (state 4050). VC state 1010
indicates ready but not in use. With the NPSI fast connect feature, the normal idle state is 1050. Other intermediate states can appear while an X.25 call or clear is in progress. The codes are listed in z/OS Communications Server: IP Messages Volume 1 (EZA).

The status of the path to TCP/IP is shown in the last line:

- 80 is normal
- 00 indicates that the TCPIP X25 NPSI device has not been started

**Termination by TCPIP STOP device**

The following example illustrates termination using the TCPIP STOP device:

```ini
EZB2091I HALT notice accepted, type 0
EZB2250I MCH XU03B terminating
EZB2352I VC 010 closed
EZB2352I VC 00F closed
...
EZB2352I VC 001 closed
EZB2480I IP AS TCPIPTES disconnected: sent 7 received 7 dropped 0
EZB2090I Terminating
EZB2099I Ended
```

EZB2480I reports the number of IP datagrams transferred on the DLC path for TCP/IP.

**Steps for diagnosing logon problems**

Several steps must take place successfully to establish an X.25 virtual circuit for TCP/IP activity:

1. An X.25 call request is received by the X.25 NPSI server from the X.25 network (incoming call) or is sent by the X.25 NPSI server to establish a connection to a new destination (outgoing call).
2. An X.25 call accept confirms the X.25 call request. Call accept is sent by TCPIPX25 for an incoming call, or received from the X.25 network for an outgoing call.
3. NPSI initiates an SNA session with the X.25 NPSI server application by means of a VTAM LOGON.

Each of these steps is reported in the activity log, shown in the “X.25 trace examples” on page 808. Problems fall into two main areas: failure of the X.25 call itself, indicated by either a refusal or an immediate clear, or failure of the NPSI LOGON. Call failures are reported with X.25 cause and diagnostic codes. Standardized cause codes include:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>DTE clearing. The remote system cleared the call.</td>
</tr>
<tr>
<td>01</td>
<td>Number busy. The called number cannot accept another call.</td>
</tr>
<tr>
<td>03</td>
<td>Invalid facility request. A facility requested by the caller is not subscribed or conflicts with a subscribed option.</td>
</tr>
<tr>
<td>05</td>
<td>Network congestion. Congestion conditions or some other problem within the network temporarily prevent the requested virtual circuit from being established.</td>
</tr>
<tr>
<td>09</td>
<td>Out of order. The called number is out of order.</td>
</tr>
</tbody>
</table>
Access barred. The caller is not permitted to obtain a connection to the called number.

Not obtainable. The called number is not assigned or is no longer assigned.

Remote procedure error. An X.25 protocol error at the remote equipment.

Local procedure error. An X.25 protocol error.

Refer to X.25 NPSI Diagnosis, Customization, and Tuning for a list of diagnostic codes. X.25 networks can also have special diagnostic codes in the range 80–FF.

VC LOGON can fail for a variety of reasons. Among the most common reasons are:

- Incorrect VTAM switched circuit definitions. IDNUM entries are error prone; SSCPFM=USSNTO and DISCNT=(YES,F) are necessary.
- A default VTAM USS table ISTINCDT that has been modified to include text in the message 10 entry.
- Coding LOGAPPL on the NPSI X25.VC definitions. LOGAPPL should only be used on the X25.MCH and on the X25.VC with the Fast Connect feature.
- Insufficient number of type 1 LUs configured on the NCP LUDRPOOL statement.

A VTAM buffer trace with ID=VTAM helps diagnose the first problem. Collect the following configuration documentation before contacting the IBM Software Support Center: X.25 NPSI server configuration data set, VTAM APPL definition for the NPSI X.25 server, NPSI definitions, and VTAM SWNET definitions for NPSI.

**Session hangs**

In diagnosing session hang or timeout problems, remember that TCPIPX25 does not track individual TCP sessions; it only transfers IP datagrams. One X.25 virtual circuit can carry datagrams from several TCP sessions. A VC can also be closed and reestablished several times during a TCP session with long periods of inactivity. Failure of an X.25 connection is not directly reflected in TCP sessions it might be carrying, only indirectly by TCP timeouts.

Opening a TCP session, such as a Telnet connection, can fail for reasons not specific to X.25, for example, a TCP/IP routing problem caused by an incorrect GATEWAY definition, or an IP routing problem in the remote device. Symptoms suggesting these problems include:

- No X.25 call is made when a TCP connection is requested.
- No traffic is received from the remote equipment, indicated by a received count of zero in the EZB2351I connection terminated message.

An established TCP connection can hang because the X.25 network or remote device is down. This is indicated by a clear cause and diagnostic, as described in “Steps for diagnosing logon problems” on page 811.

**Helpful hints**

PING fails but Telnet and FTP connect. Setting up a new X.25 connection might take longer than the default PING timeout on a busy system. Use the PING TIMEOUT or COUNT parameters to extend the waiting time. Use the NPSI GATE Fast Connect feature to reduce connection setup time.
PING succeeds but Telnet or FTP data transfer times out. Full-screen Telnet and FTP data transfers create large IP datagrams, while PING uses smaller ones. If the small datagrams go through but large ones do not, there might be a problem with MAXDATA on the VTAM switched circuit definitions; see “Configuration requirements” on page 807 for details. Attempting to pass a datagram larger than MAXDATA on a virtual circuit hangs the VC for all subsequent traffic.

A load-dependent hang can be due to an insufficient number of virtual circuits.

The TRAFFIC command can be used to observe virtual circuit data transfer activity.

**Documentation requirements**

If IBM Support Center help is needed, collect the following configuration documentation before contacting IBM:

- X.25 NSPI server console log showing X.25 connections related to the problem
- X.25 NPSI server configuration data set
- PROFILE.TCPIP data set
- NPSI definitions
- VTAM SWNET definitions for NPSI
Chapter 35. Diagnosing IMS problems

This topic describes how to diagnose IMS problems, and contains the following sections:

- “Steps for setting up the IMS TCP/IP services socket interface system” on page 817
- “Common configuration mistakes” on page 819
- “Quick checklist for common problems” on page 819
- “Documentation references for problem diagnosis” on page 833

The IMS TCP/IP Services socket interface allows TCP/IP clients to access IMS using a TCP/IP network. This access is fully described in the z/OS Communications Server: IP IMS Sockets Guide. A sockets program-to-program connection is established between a client (TCP/IP socket) program and a server (IMS application) program. TCP/IP and the Listener are agents in the connection establishment. The components of the IMS TCP/IP socket interface system are shown in Figure 104 on page 816.
The following list is a brief description of the component interaction and data flow that occurs when a client program requests an IMS transaction.

1. The client program starts and sends the transaction request message (TRM) to the Listener port.

2. The Listener reads the TRM and accepts the socket connection between the client program and the Listener from TCP/IP.

3. The Listener validates the TRM, prepares to give the socket connection to the IMS transaction, builds the transaction initiation message (TIM) containing the socket connection information, and sends the TIM to the IMS transaction manager message queue. For implicit IMS transactions, the Listener also reads the input data from the client program and sends it to the message queue.

4. The IMS transaction manager schedules the requested transaction.

Figure 104. Components of the IMS TCP/IP services socket interface system
IMS Transaction. This can be one of the following:

Implicit
The IMS assist module receives the TIM on behalf of the implicit IMS transaction and takes the socket connection from the Listener. The input data is read and the IMS transaction performs the required database access. The IMS assist module, on behalf of the implicit IMS transaction, writes the output data to the client program, through the socket connection, followed by the commit status message (CSM). The socket connection then closes.

Explicit
The explicit IMS transaction receives the TIM and takes the socket connection from the Listener. Input and output data is read and written as defined by the protocol, and the required database access is performed. The explicit IMS transaction writes the CSM to the client program and closes the socket connection.

The IMS transaction and the client program terminate.

**Steps for setting up the IMS TCP/IP services socket interface system**

Perform the following steps to establish the system described in Figure 104 on page 816.

Steps for setting up the IMS TCP/IP services socket interface system

1. Configure TCP/IP to reserve the Listener port number.
   A TCP/IP port should be reserved for the Listener to connect to when it starts. The following is a sample profile statement to reserve the Listener port.
   ```
   PORT 4096 TCP EZAIMSLN
   ```
   Refer to the [z/OS Communications Server: IP IMS Sockets Guide](#) for details about the PORT statement.

2. Configure the TCP/IP network from the server host to the client host.
   For the client program to issue IMS transaction requests across a socket connection, there must be a TCP/IP network defined between the client and server hosts. Any physical network supported by IBM MVS TCP/IP can be used to establish this socket connection.
   Refer to the appropriate information in the [z/OS Communications Server: IP Configuration Reference](#) for details about how to configure the required network to the server host TCP/IP.

3. Start the TCP/IP address space on the server host.

4. Establish and verify the network connection from the client host to the server host.
Depending on the network connection, start or activate the required device drivers and network nodes required to establish a TCP/IP network connection.

To verify the TCP/IP network connection, use the PING command on the client host, using the server host destination IP address or network name.

---

5. Define the Listener to the IMS transaction manager.
   The IMS transaction manager must be defined to expect message queue input from the Listener. For information about how to define the Listener to IMS, refer to the Listener IMS definitions in the z/OS Communications Server: IP IMS Sockets Guide.

---

6. If the IMS transaction that is requested by the client program is not already written, write it.
   Refer to the z/OS Communications Server: IP IMS Sockets Guide for specific details about writing IMS transactions that can be requested by a TCP/IP client program.

---

7. Define the IMS transaction that is requested by the client program to the IMS transaction manager.
   The IMS transaction must be defined to IMS before the Listener can request it to be scheduled on behalf of the client program. Refer to the z/OS Communications Server: IP IMS Sockets Guide for important restrictions when defining IMS transactions.

---

8. Start the IMS transaction manager and the IMS database manager.

---

9. Complete the Listener configuration data set.
   The Listener configuration data set is read when the Listener is started. The procedure used to start the Listener (usually EZAIMSLN) uses the ddname LSTNCFG to specify the Listener configuration data set. Following is an example statement that specifies TCPIP.LISTENER.DATA as the configuration data set.

   ```plaintext
   LSTNCFG DD DSN=TCPIP.LISTENER.DATA,DISP=SHR
   ```
   This data set must contain a minimum set of required statements to specify the environment the Listener is started in and the list of IMS transactions available to client programs.
   Refer to the z/OS Communications Server: IP IMS Sockets Guide for details about the format and contents of this data set.

---

10. Start the Listener address space.
    The Listener is started as an MVS address space as described in the z/OS Communications Server: IP IMS Sockets Guide. The JCL procedure required for starting the address space is also listed in the z/OS Communications Server: IP IMS Sockets Guide.

---

11. Write the client program, if not already written.
Refer to the **z/OS Communications Server: IP IMS Sockets Guide** for programming details about client programs that can request IMS transactions over a TCP/IP network.

12. Start the client program.

---

### Common configuration mistakes

The following is a list of common configuration mistakes:

- The IMS transaction has not been defined in the Listener configuration data set.
- The Implicit or Explicit parameter in the Listener configuration data set does not match the protocol used by the IMS transaction.
- The program specification block (PSB) for the Listener does not include the ALTPCB label.
- The IMS transaction invoked by the Listener does not specify the MODE=SNGL parameter on the IMS TRANSACT macro in the IMS database manager definition. Refer to the **z/OS Communications Server: IP IMS Sockets Guide** for information about restrictions on application programs.
- The IMS transaction invoked by the Listener was not defined to the IMS transaction manager as a multisegment transaction.
- The IMS transaction invoked by the Listener is an IMS conversational transaction or executes in a remote Multiple Systems Coupling (MSC) environment.

### Quick checklist for common problems

The following list summarizes some initial checks that can be made quickly and are helpful in identifying the problem area.

1. Is the TCP/IP network active?
   
   To verify that the network to the server host is active, use the PING command on the client host, using the same IP address or host name as specified in the client program.

2. Is the Listener started and active on the server host?
   
   Check that the Listener address space is active and running. The MVS SDSF facility can be used to view the active address space list. Also see “Using NETSTAT” on page 834 for details about how to determine if the Listener TCP/IP port is active.

3. Did the Listener program list any configuration errors to the SYSPRINT data set?
   
   Check the JCL DD statement in the Listener start procedure to identify the destination of the SYSPRINT output. See “Where to find error message documentation” on page 836 to determine the reason for any errors. The Listener address space might need to be stopped to flush any error messages to the destination.

4. Have you completed all of the required definitions. See “Steps for setting up the IMS TCP/IP services socket interface system” on page 817 for the list of required definitions and configurations.

5. Is the client program connecting to the same TCP/IP port as the Listener?
   
   See “Using NETSTAT” on page 834 for details about how to use the...
NETSTAT command to identify which port the Listener is connected to and which port the client program is establishing a socket connection on.

**Component problems**

Table 87 lists some of the problems related to starting or stopping one of the components in the IMS TCP/IP Services socket interface system.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
</table>
| The Listener terminates on startup | 1. Incorrect configuration data set.  
2. The prerequisites for starting the Listener have not been completed.  
3. Incorrect method of starting. | 1. Check for configuration error messages written to the SYSPRINT data set and correct the problems (if any).  
2. Complete the required steps listed in “Steps for setting up the IMS TCP/IP services socket interface system” on page 817.  
3. Ensure the Listener is being started as an MVS address space as described in the z/OS Communications Server: IP IMS Sockets Guide. The JCL procedure required for starting the address space is also listed in z/OS Communications Server: IP IMS Sockets Guide. |
| The Listener does not terminate | The Listener waits for all of the currently open socket connections to close before it responds to the user termination request. If any of the socket connections have hung, the Listener needs to be forcibly terminated. | Force the Listener to terminate using the command specified in the section about stopping the IMS Listener in the z/OS Communications Server: IP IMS Sockets Guide. See “Connection problems” on page 821 for a description of how socket connections can hang. |
| As the Listener is starting, messages are written to the system console asking if IMS should be started | The IMS system should be started before the Listener. If the Listener is started first, the operator is prompted to start the IMS system. | Reply to the console messages to start IMS. |
| An implicit IMS transaction written in C is experiencing unexpected problems at startup | If IMS transaction programs written in C are not built correctly, the IMS interface fails on startup. | Build the C program correctly as specified in the section about writing an IMS TCP/IP Services server program in z/OS Communications Server: IP IMS Sockets Guide. |
Table 87. Component problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Listener is abending while accepting the TRM</td>
<td>If a user-defined security exit has been linked into the Listener, it might be causing the problem. The security exit is called when validating the TRM. If the security exit has not been written to accept the required linkage and parameters, the Listener abends because the exit runs in the same address space.</td>
<td>Check that the security exit has been written to accept the linkage and parameters as specified in the section on the IMS security exit in the z/OS Communications Server: IP IMS Sockets Guide.</td>
</tr>
</tbody>
</table>

Connection problems

Table 88 lists some problems related to the TCP/IP socket connection. They include problems with establishing the connection, transferring data over the connection, and unexpected loss of the connection.

Table 88. Connection problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The client program is experiencing intermittent reject connect responses from TCP/IP</td>
<td>The TCP/IP sockets facility has a connection request backlog queue. While this queue is full, further connection attempts are rejected by TCP/IP. Under load, this queue can temporarily fill, causing some client program requests to be silently ignored.</td>
<td>To reduce the frequency of this problem, increase the size of the backlog queue. The size of the queue is controlled by a parameter in the Listener configuration data set and is limited by the SOMAXCONN statement in the TCPIP PROFILE.</td>
</tr>
<tr>
<td>The TCP/IP socket connection to the client program is being broken immediately after the implicit IMS transaction is scheduled</td>
<td>The Listener configuration data set might incorrectly define the implicit IMS transaction as explicit. In this case, the Listener does not pass the input data to the IMS transaction through the message queue as expected. The transaction starts, and upon detecting no data, immediately close the TCP/IP socket connection and terminate.</td>
<td>Verify that the TRANSACTION statements in the Listener configuration data set specify the TYPE parameter correctly.</td>
</tr>
</tbody>
</table>
### Table 88. Connection problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection lockup for an implicit IMS transaction</td>
<td>The Listener might be waiting for the end-of-message (EOM) segment from the client program. The client program must send a valid EOM segment before the Listener instructs the IMS transaction manager to schedule the IMS transaction. If the client program does not send a recognized EOM segment, the Listener waits indefinitely for it, while the client program waits for a response.</td>
<td>Use the IP packet trace facility to determine whether the client program is sending a valid EOM segment. See &quot;Using IP packet trace&quot; on page 833 for details about the IP packet trace facility. Refer to the information about implicit-mode application data in the z/OS Communications Server: IP IMS Sockets Guide for the format of the EOM segment.</td>
</tr>
</tbody>
</table>

Connection lockup for an explicit IMS transaction

A connection lockup occurs when both the explicit IMS transaction and the client program are waiting for data from the other end of the socket connection.

1. Because the explicit IMS transaction protocol is user defined, programming errors can easily lead to connection deadlocks. That is, the server is waiting for more data while the client is waiting for a response, and both wait indefinitely.

2. The Listener configuration data set might incorrectly define the explicit IMS transaction as implicit. In this case the Listener waits for valid implicit data from the client program, or if valid data is received, the explicit IMS transaction waits for data from the client program because the Listener has already read the data and written it to the message queue.

1. Use the IP packet trace facility to identify which part of the protocol is failing. See "Using IP packet trace" on page 833 for details about the IP packet trace facility.

2. Verify that the TRANSACTION statements in the Listener configuration data set specify the TYPE parameter correctly.

Timeouts, especially in the client program, are recommended when issuing socket READs to avoid deadlocks and allow easy diagnosis. Refer to the information about SELECT calls in the z/OS Communications Server: IP IMS Sockets Guide for more information about specifying timeouts for READs.
## Table 88. Connection problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
</table>
| Connection lockup for either an explicit or implicit IMS transaction | 1. If the TRM sent by the client program is incomplete, the Listener waits indefinitely for the rest of the message. 2. If the IMS transaction does not successfully issue the takesocket to gain the connection from the Listener, the Listener waits for this event indefinitely. The takesocket might not be issued successfully due to one of the following reasons:  
- The IMS transaction is defined to run in a message processing region that is not started. In this case, the IMS transaction is never scheduled and, therefore, never issue the takesocket.  
- One of the several TCP/IP socket calls, up to and including the takesocket, might fail and terminate the IMS transaction.  
- An IMS error can stop the transaction from being successfully scheduled, or, especially in the explicit case, can cause the IMS transaction to terminate before the takesocket is issued. | 1. Check the length and format of the TRM by using the IP packet trace facility as described in “Using IP packet trace” on page 833. 2. Check that the IMS transaction is being successfully scheduled by the IMS transaction manager and ensure that any IMS and socket calls issued by the IMS transaction are checked for unsuccessful return codes. |
Table 88. Connection problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The takesocket call issued by the IMS transaction fails</td>
<td>1. IMS can, for recovery reasons, abend a transaction and start it again. If the transaction is abended after it has gained the socket connection (through a takesocket call), the TCP/IP socket connection is lost. Although IMS restores the message queue when it restarts the transaction, the takesocket issued by the transaction fails as the socket connection has already been taken from the Listener.</td>
<td>1. Restart the client program. To reduce the frequency of this problem, determine why IMS is restarting the IMS transaction by using the IMS trace facility. See “IMS traces” on page 834. 2. Make certain the IMS transaction is defined as multisegment.</td>
</tr>
<tr>
<td>Note: For implicit transactions, the IMS assist module routines issue a takesocket for the first get unique (GU) issued by the transaction. If the takesocket fails, the GU returns ZZ.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The client program is always receiving reject connect responses from TCP/IP</td>
<td>The maximum number of active sockets might have been reached, with all the currently active socket connections unable to complete. An increasing number of socket connections eventually reduces the available socket connections to zero when the number of socket connections equals the MaxActiveSockets configured for the Listener. When this happens, TRMs are not processed by the Listener, and they are left on the TCP/IP backlog queue. When the backlog queue fills, TCP/IP silently ignores a client program connection attempt.</td>
<td>Identify the client programs causing the problem using the NETSTAT command as specified in “Using NETSTAT” on page 834. then continue diagnosis to determine why these connections are locking up. The Listener must be restarted to clear the active socket list. Because there are active socket connections, the Listener must be forced to terminate using the command specified in the z/OS Communications Server: IP IMS Sockets Guide.</td>
</tr>
</tbody>
</table>
Table 88. Connection problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection lockup or loss when passing a socket connection from one explicit IMS transaction to another</td>
<td>To pass a socket connection from the first IMS transaction to the second, the first IMS transaction must wait after it issues the givesocket until the second IMS transaction issues a takesocket; otherwise, the connection is lost.</td>
<td>When passing a socket connection between IMS transactions, make sure the first transaction waits for the second to issue the takesocket and that both IMS transactions can be scheduled to run at the same time.</td>
</tr>
<tr>
<td>A connection lockup is when the socket connection reaches a state where it never completes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A connection lockup can occur when the first IMS transaction waits for the takesocket to be issued, but both IMS transactions are defined to run in the same message processing region. In this case, they cannot both be scheduled to run at the same time, and the first IMS transaction waits indefinitely for the takesocket from the second IMS transaction, which is never scheduled.</td>
<td></td>
</tr>
</tbody>
</table>

Error message and return code problems

Table 89 lists problems related to error responses.

Table 89. Error message and return code problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The client program is receiving a request status message (RSM)</td>
<td>The Listener sends this message to the client program when it detects an error condition.</td>
<td>Use the return and reason codes from the message to look up the explanation. See &quot;Where to find return code documentation&quot; on page 835.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The implicit IMS transaction is receiving return codes in the I/O program communication block (PCB) that are not defined in the section on status codes in the IMS Version 8: Diagnosis Guide and Reference</td>
<td>The IMS assist module performs several socket-related functions on behalf of the implicit IMS transaction in response to IMS transaction manager requests. When errors are detected that are not related to the IMS transaction manager request, the IMS assist module sets special return codes in the PCB.</td>
<td>Look up the meaning of the special return codes. See &quot;Where to find return code documentation&quot; on page 835.</td>
</tr>
</tbody>
</table>
Table 89. Error message and return code problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Listener error messages are written to the MVS system console instead of the SYSPRINT data set</td>
<td>If the Listener experiences data set I/O errors, it redirects the error messages to the MVS system console.</td>
<td>Check the MVS system console log for I/O errors on the data set to identify the problem. The SYSPRINT DD statement in the JCL procedure to start the Listener specifies the destination data set for the error messages.</td>
</tr>
</tbody>
</table>

Socket data protocol problems

Table 90 lists problems related to data transfer over the socket connection. They include incorrect data sent, not enough or too much data sent, and data corruption.

Table 90. Socket data protocol problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Listener is not responding to the client program</td>
<td>1. If the TRM sent by the client program is incomplete, the Listener waits indefinitely for the rest of the message.</td>
<td>1. Check the length and format of the TRM by using the IP packet trace facility as described in “Using IP packet trace” on page 833.</td>
</tr>
<tr>
<td></td>
<td>2. If the port specified by the client program is not the port that is attached to the Listener, and the socket connection is established, the other end of the connection does not communicate with the client program as required.</td>
<td>2. Check that the Listener is attached to the port used by the client program to establish the socket connection. Use the command specified in “Using NETSTAT” on page 834.</td>
</tr>
<tr>
<td>All the input data sent from the client program is not being passed to the implicit IMS transaction from the Listener</td>
<td>Any input data written after the first EOM segment is ignored by the Listener.</td>
<td>Check for EOM segments being sent by the client program by using the IP packet trace facility described in “Using IP packet trace” on page 833. Refer to the information about the implicit-mode application data in the z/OS Communications Server: IP IMS Sockets Guide for the format of the EOM segment.</td>
</tr>
</tbody>
</table>
Table 90. Socket data protocol problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit IMS transaction is receiving garbled data from or sending garbled data to the client program</td>
<td>The data might need translation when the client program does not exist on an EBCDIC host. For explicit data transfer, the client program, or the IMS transaction, or both, must provide ASCII to EBCDIC translation and byte-order translation of fixed-point binary integers, if required. The Listener automatically translates the TRM when creating the TIM.</td>
<td>Code the client program or the IMS transaction or both to provide the necessary translation when the client program is not on an EBCDIC host.</td>
</tr>
</tbody>
</table>
### Table 90. Socket data protocol problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implicit IMS transaction is receiving garbled data from or sending</td>
<td>The automatic data translation when the client program does not exist on an EBCDIC host can be causing the problem. For implicit data transfer, the Listener automatically translates input data from ASCII to EBCDIC, based on the TRM contents. The IMS assist module also automatically translates output data from EBCDIC to ASCII when sending to an ASCII client program, as determined by the TRM. If the TRM sent by the client program is not either ASCII or EBCDIC as required, then the automatic translations fail. The client program is also responsible for any required byte-order translation of fixed-point binary integers. Notes: 1. If the data translated between ASCII and EBCDIC contains any nonprintable data, such as integers, flags, or reserved fields, the data is corrupted. In this case, the client program must provide EBCDIC data (including the TRM) for the IMS transaction and expect EBCDIC data from the IMS transaction. 2. If the data is translated between ASCII and EBCDIC and contains characters that are not common to both the ASCII and EBCDIC tables, the nontranslatable characters is translated to spaces.</td>
<td></td>
</tr>
<tr>
<td>receiving garbled data to the client program</td>
<td>Code the client program to provide the necessary translation when the client program is not on an EBCDIC host and the automatic data translation cannot be used.</td>
<td>Code the client program to provide the necessary translation when the client program is not on an EBCDIC host and the automatic data translation cannot be used.</td>
</tr>
<tr>
<td>The security exit does not validate user data from the client program</td>
<td>The security exit might not be successfully linked into the Listener. The exit must be compiled and assembled and then linked into the Listener for it to be called.</td>
<td>Check that the security exit has been coded and built correctly as specified in the z/OS Communications Server: IP IMS Sockets Guide.</td>
</tr>
<tr>
<td>The security exit does not validate user data from the client program</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 90. Socket data protocol problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data is corrupted after an implicit IMS transaction issues a GU</td>
<td>The I/O area declared might be too small. When using the IMS assist module, the I/O area provided for the GU call must be large enough to hold the TIM, even though the data eventually returned in the I/O area can be smaller.</td>
<td>Make certain the implicit IMS transaction has enough storage declared to hold the TIM. The size of this message is specified in the z/OS Communications Server: IP IMS Sockets Guide.</td>
</tr>
</tbody>
</table>

The PL/I IMS transaction is receiving or sending message segments that are not valid | The message segments might be declared incorrectly. The PL/I API interface to the IMS transaction manager defines the message segments with a four-byte length field, but the length value must include only two of those bytes plus the rest of the segment. | Use the following rules to avoid problems:  
• The IMS assist module PL/I API routines mimic the interface used by the PL/I API routines. Code PL/I implicit transaction message segments in exactly the same manner as for this interface.  
• Code the client program in exactly the same manner as for all the IMS transaction API interfaces. The IMS assist module routines automatically converts the message segments from the PL/I API to the standard format.  
• Explicit transactions do not use the IMS assist module. The message segment format, if required, must match on both the client program and the IMS transaction sides. It is recommended that the standard message segment format be used.  
Refer to the information about programming considerations for the implicit-mode server and the explicit-mode server in the z/OS Communications Server: IP IMS Sockets Guide for more details about the PL/I API issues. |

IMS transaction build problems  
Table 91 on page 830 lists some problems related to building a component in the IMS TCP/IP Services socket interface system.
### Table 91. IMS transaction build problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unresolved external reference errors are causing the linker to fail when linking an IMS transaction</td>
<td>1. The implicit IMS transaction link JCL is not including the IMS assist module and the MVS TCP/IP Services sockets library to resolve external references. &lt;br&gt;2. The explicit IMS transaction link JCL is not including the MVS TCP/IP Services sockets library to resolve external references.</td>
<td>1. Compare the link JCL to the sample provided in the section about JCL for linking an implicit-mode server in the z/OS Communications Server: IP IMS Sockets Guide. &lt;br&gt;2. Compare the link JCL to the sample provided in the section about JCL for linking an explicit-mode server in the z/OS Communications Server: IP IMS Sockets Guide.</td>
</tr>
</tbody>
</table>

### IMS database problems

Table 92 lists some problems related to unexpected IMS database actions or failures. They include changes not made or requests for changes that fail.

### Table 92. IMS database problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The IMS transaction is terminating without performing the required function and without issuing any error messages</td>
<td>The IMS transaction might not be checking for interface errors.</td>
<td>It is the responsibility of the IMS transaction programmer to identify and issue error messages if the IMS database manager, IMS transaction manager, or TCP/IP socket interfaces fail.</td>
</tr>
<tr>
<td>The client program is not receiving any data from the implicit IMS transaction, but is receiving a successful CSM</td>
<td>The IMS transaction might be issuing an IMS database rollback (ROLB) call. If the IMS transaction issues a ROLB call, all output accumulated by the IMS assist module is discarded as part of the ROLB function. Depending on how the IMS transaction is coded, it might complete without further output (ISRT calls).</td>
<td>Use caution in issuing ROLB calls in implicit IMS transactions using the IMS assist module. Make certain you understand the details about implicit-mode support for ROLB processing in the z/OS Communications Server: IP IMS Sockets Guide.</td>
</tr>
</tbody>
</table>
Table 92. IMS database problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
</table>
| Local IMS transaction manager ISRT/GU/GN calls are failing when issued in IMS transactions | Local calls assume a terminal has requested the IMS transaction. The input and output of data, however, is actually sent across the socket connection for IMS transactions started by the Listener. The following is a list of specific causes of the problem:  
  • The ISRT call has no terminal associated with the IMS transaction for the output.  
  • There is no data on the message queue for explicit IMS transactions to get with the GU or GN calls.  
  • An implicit IMS transaction receives an unexpected TIM in response to a GU call. | Do not issue local IMS transaction manager calls from transactions started by the Listener. An implicit IMS transaction must use the IMS assist module calls, which accesses either a terminal or socket connection, as required. An explicit IMS transaction must interface directly to the socket connection. |
| The ISRT call fails for an implicit IMS transaction if a large amount of data is output | The IMS assist module restricts the total output for a single IMS transaction execution to 32KB. | Limit the output for an implicit IMS transaction using the IMS assist module to a total of 32KB. |
Table 92. IMS database problems (continued)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The IMS database manager commits the changes made by an IMS transaction, but the client program receives an error</td>
<td>1. The implicit IMS transaction does not issue a second GU. The IMS database commits the changes either when the IMS transaction ends or when another GU is issued. For implicit IMS transactions, the IMS assist module routines send the output data and CSM to the client program and closes the socket connection when the second GU is issued. If the implicit IMS transaction does not issue another GU, the changes are committed when the transaction ends, but the client program assumes failure when the CSM is not received.</td>
<td>1. Implicit IMS transactions that are started by the Listener must issue GU calls to get the next transaction request until the GU call returns with no requests to process. 2. Where possible, the client program should be coded to automatically restart the IMS transaction and handle the condition where the IMS transaction is duplicated. For explicit IMS transactions, a more rigorous protocol can be implemented. <strong>Guideline:</strong> This should be considered as an uncommon case.</td>
</tr>
<tr>
<td></td>
<td>2. The socket connection might have been broken after the changes were committed but before the CSM was sent. In this case, the client program assumes failure, but the changes have been committed.</td>
<td></td>
</tr>
<tr>
<td>The client program does not receive a valid CSM from an implicit IMS transaction</td>
<td>The client program might not have completed the response protocol correctly. The client program must read the response data until it reads an EOM segment. The CSM immediately follows the EOM.</td>
<td>Use the IP packet trace facility to determine whether the IMS transaction is sending a valid EOM segment followed by a valid CSM segment. See “Using IP packet trace” on page 833 for details about the IP packet trace facility. If the correct message segments are being sent, correct the client program to receive the response data. <strong>Refer to the z/OS Communications Server: IP IMS Sockets Guide</strong> for the format of the EOM and CSM segments.</td>
</tr>
</tbody>
</table>
Documentation references for problem diagnosis

This section contains the information and documentation references required to gather and decode diagnostic information about the IMS TCP/IP Services socket interface system.

The two main tools used for problem diagnosis are the IP packet trace facility and the NETSTAT utility. The use of these tools is explained in following sections and example statements and commands are provided. An explanation of how to interpret the output from each of these tools is also provided.

For TCP/IP or IMS-specific tracing, reference is made to the appropriate diagnosis documentation.

Two cross-reference sections, which list all the types of return codes and error messages that can be issued from the IMS TCP/IP Services socket interface system, are provided at the end of this section. For each type of return code and error message, a reference is made to existing documentation that provides a complete description.

Traces

The following traces can be used to gain information about data flows and actions of the IMS TCP/IP Services socket interface system. The IP packet trace facility is the most helpful trace facility when writing and debugging your own client programs and IMS transactions. The TCP/IP internal traces are mainly used to diagnose problems with the TCP/IP network and socket-specific problems. The IMS traces are mainly used to diagnose IMS-specific problems, such as IMS transaction scheduling and database commit and rollback errors. The IMS assist module trace is used to determine problems with the IMS Assist module. This trace can enabled by adding a sysdebug dd card to the IMS region procedure where the IMS transaction using the Assist Module is running.

Using IP packet trace

Use IP packet trace to identify the flow of data between the client program and the Listener and IMS transaction servers. TCP packets can be traced on the socket connections established through the Listener-reserved port. If the IP address of the client program is specified, only packets originating from or destined to the client program are traced. Specifying this parameter is recommended to avoid tracing a large number of unrelated TCP packets.

Restriction: When using X.25 devices to provide the network to the client program, the IP packet trace facility must be activated from the individual device address spaces. The previous example only activates tracing in the TCP/IP address space.

See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47 for details about how to use the IP packet trace facility.

The packets that contain data display the data in hexadecimal digits and, in this case, their EBCDIC characters. The numeric fields in the message segments can be verified from the hexadecimal representation, while any alphabetic data can be verified from the EBCDIC display.

TCP/IP internal traces

The TCP/IP internal traces are sent to CTRACE. This is a key trace used to determine the success or failure of the socket calls made by the IMS Listener and the IMS transactions. These traces provide information about the internals of the
TCP/IP address space. This information can be used to diagnose problems in establishing the network between the client program and the server host or in establishing the socket connections. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 47, for details about how to use the TCP/IP internal tracing facility.

**IMS traces**
The IMS traces provide information about the internals of the IMS database system. This information can be used to diagnose IMS transaction scheduling problems, IMS transaction manager message queue problems, and database change problems that cause rollbacks or commit errors. For an overview of monitoring the IMS system, refer to [IMS Version 8: Administration Guide: System](https://www.ibm.com). For details about tracing and reading the trace reports refer to the [IMS Version 8: Utilities Reference: System](https://www.ibm.com).

**Using NETSTAT**
This section details how to use NETSTAT to query TCP/IP port usage and the state of socket connections. This command can be used to verify that the Listener is active and has opened the correct port and to diagnose problems with the socket connection between the client program and the Listener or IMS transaction.

**Restriction:** The client program must have the socket connection open for NETSTAT to query the connection status.

The NETSTAT SOCKETS command displays which ports are open to which address spaces and displays active socket connections and their status. Following is sample output from this command (the output shown is valid for V2R10 and V1R2):
READY
netstat sockets
MVS TCP/IP NETSTAT CS V2R10 TCPIP Name: TCPCS 12:34:56
Sockets Interface status:

<table>
<thead>
<tr>
<th>Type</th>
<th>Bound to</th>
<th>Connected to</th>
<th>State</th>
<th>Conn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dgram 0.0.0.37</td>
<td><em>..</em></td>
<td>UDP</td>
<td>00000058</td>
<td></td>
</tr>
<tr>
<td>Dgram 0.0.0.13</td>
<td><em>..</em></td>
<td>UDP</td>
<td>00000057</td>
<td></td>
</tr>
<tr>
<td>Dgram 0.0.0.19</td>
<td><em>..</em></td>
<td>UDP</td>
<td>00000056</td>
<td></td>
</tr>
<tr>
<td>Dgram 0.0.0.9</td>
<td><em>..</em></td>
<td>UDP</td>
<td>00000055</td>
<td></td>
</tr>
<tr>
<td>Stream 0.0.0.623</td>
<td>0.0.0.0..0</td>
<td>Listen</td>
<td>0000004B</td>
<td></td>
</tr>
<tr>
<td>Stream 0.0.0.514</td>
<td>0.0.0.0..0</td>
<td>Listen</td>
<td>0000004D</td>
<td></td>
</tr>
<tr>
<td>Stream 0.0.0.513</td>
<td>0.0.0.0..0</td>
<td>Listen</td>
<td>0000004C</td>
<td></td>
</tr>
<tr>
<td>Stream 0.0.0.512</td>
<td>0.0.0.0..0</td>
<td>Listen</td>
<td>0000004E</td>
<td></td>
</tr>
<tr>
<td>Stream 0.0.0.37</td>
<td>0.0.0.0..0</td>
<td>Listen</td>
<td>00000053</td>
<td></td>
</tr>
<tr>
<td>Stream 0.0.0.7</td>
<td>0.0.0.0..0</td>
<td>Listen</td>
<td>0000004F</td>
<td></td>
</tr>
<tr>
<td>Stream 0.0.0.13</td>
<td>0.0.0.0..0</td>
<td>Listen</td>
<td>00000052</td>
<td></td>
</tr>
<tr>
<td>Stream 0.0.0.19</td>
<td>0.0.0.0..0</td>
<td>Listen</td>
<td>00000051</td>
<td></td>
</tr>
<tr>
<td>Stream 0.0.0.9</td>
<td>0.0.0.0..0</td>
<td>Listen</td>
<td>00000050</td>
<td></td>
</tr>
</tbody>
</table>

Name: OSNMPD Subtask: 0060B7A0
Dgram 0.0.0.161 | *..* | UDP | 00000013 |
Stream 0.0.0.1027 | 0.0.0.0..0 | Listen | 00000014 |
Name: TCPCS Subtask: 00600000
Stream 127.0.0.1..23 | 127.0.0.1..1033 | Establish | 00000045 |
Stream 9.67.113.27..23 | 9.37.81.207..1096 | CloseWait | 00000039 |
Name: TCPCS Subtask: 00655780
Stream 0.0.0.23 | 0.0.0.0..0 | Listen | 00000012 |
Name: TCPCS Subtask: 006056F0
Stream 127.0.0.1..1026 | 127.0.0.1..1025 | Establish | 0000000F |
Name: TCPCS Subtask: 00605CF0
Stream 0.0.0.1025 | 0.0.0.0..0 | Listen | 0000000C |
Stream 127.0.0.1..1025 | 127.0.0.1..1026 | Establish | 00000010 |
Name: USER18 Subtask: 006A3400
Stream 127.0.0.1..1033 | 127.0.0.1..23 | Establish | 00000044 |

READY

Refer to [z/OS Communications Server: IP User’s Guide and Commands](#) for more details about the usage, parameters, and output of NETSTAT.

Where to find return code documentation

The following list refers to the appropriate return code documentation for all return codes expected in the IMS TCP/IP Services socket interface system.

- To the client from the Listener (request status message).

Refer to the information about the request status message (RSM) segment in the [z/OS Communications Server: IP IMS Sockets Guide](#) for the format of the RSM and a description of the return codes.

**Guideline:** The RSM with the “IMS transaction unavailable to be started” return code, is returned when the IMS transaction has previously abended or failed and the IMS transaction manager has marked it as not able to be scheduled.

- To the client from an IMS transaction (CSM).

The CSM is received by the client program when the transaction is successful. This message implies a successful return code. If this message is not received, the client program must assume the IMS transaction has not completed successfully.

- To the implicit IMS transaction from the IMS assist module (I/O program communication block).
Refer to the information about the I/O PCB implicit-mode server in the *z/OS Communications Server: IP IMS Sockets Guide* for the format of the I/O PCB and return code explanations.

- To an implicit/explicit IMS transaction from TCP/IP.
  Refer to the information about error messages and return codes for IMS sockets calls in the *z/OS Communications Server: IP IMS Sockets Guide*.

- To an implicit/explicit IMS transaction from the IMS transaction manager.
  Refer to the information about DL/I status codes, return codes, and reason codes in the *IMS Version 8: Diagnosis Guide and Reference*.

- To an implicit/explicit IMS transaction from the IMS database manager.
  Refer to the information about DL/I status codes, return codes, and reason codes in the *IMS Version 8: Diagnosis Guide and Reference*.

**Where to find error message documentation**

The following list refers to the appropriate error message documentation for all error messages expected in the IMS TCP/IP Services socket interface system.

- Error messages from the Listener are written to the SYSPRINT ddname data set. Refer to the information about the IMS Listener error messages in the *z/OS Communications Server: IP IMS Sockets Guide* for descriptions of the error messages in this data set.

- Error messages from TCP/IP are written to the SYSERROR and SYSDEBUG data sets. Refer to the *z/OS Communications Server: IP IMS Sockets Guide* for descriptions of the error messages in these data sets.
Chapter 36. Diagnosing VMCF/TNF/IUCV problems

This topic describes how to diagnose VMCF/IUCV problems and restartable VMCF/TNF problems.

Diagnosing restartable VMCF/TNF problems

This section describes how to diagnose restartable VMCF/TNF problems and contains the following subsections:

- “VMCF or TNF fail to initialize”
- “Abends 0D5 and 0D6”
- “Steps for diagnosing no response to commands”
- “VMCF or TNF does not stop” on page 838

You can configure virtual machine communication facility (VMCF) and termination notification facility (TNF) in two different ways: as restartable subsystems or as nonrestartable subsystems. For details about configuration, refer to the z/OS Communications Server: IP Configuration Reference.

If you choose restartable VMCF and TNF, you might encounter the problems described in this topic.

Note: For information about common VMCF and TNF problems, refer to z/OS Communications Server: IP Configuration Guide.

VMCF or TNF fail to initialize

If VMCF or TNF fail to initialize with an OC4 abend, there is probably an installation problem. Check the PPT entries for errors. Some levels of MVS do not flag PPT syntax errors properly.

Abends 0D5 and 0D6

If, after removing a user, the system crashes with abends 0D5 and 0D6, the application is probably still running and using VMCF. Users should not be removed from VMCF or TNF without first terminating the affected user.

Steps for diagnosing no response to commands

If VMCF and TNF do not respond to commands, one or both of the nonrestartable versions of VMCF or TNF are still active.

Perform the following steps to stop and restart the subsystems.

1. Stop all VMCF and TNF users.

2. Stop the subsystems using the commands FORCE ARM VMCF and FORCE ARM TNF.

3. Restart using EZAZSSI.
VMCF or TNF does not stop

If you are unable to stop VMCF or TNF, users probably still exist in the VMCF and TNF lists. Use the F VMCF,DISPLAY,NAME=* and the F TNF,DISPLAY,NAME=* commands to identify those users who are still active; then either cancel those users or remove them from the lists, using the F VMCF,REMOVE and the F TNF,REMOVE commands.

Diagnosing VMCF/IUCV problems with the TSO MVPXDISP command

The TSO MVPXDISP command is used as a debugging aid to display the state of the connections from some address spaces to the VMCF address space. In addition, the command is used to obtain information about storage utilization for VMCF and IUCV related buffers, as well as routines supporting the underlying PC functions. This information can be used by the IBM Software Support Center to analyze the state of the VMCF address space.

The TSO MVPXDISP command is used to display information about a connection for a single user ID or started task to the VMCF address space, or all connections can be displayed. The command is also used to obtain information about the storage utilization. MVPXDISP must be an Authorized Program Facility (APF) command.

If you have a user application that is hung, issue the TSO MVPXDISP command and keep the output for help in diagnosing the problem.

MVPXDISP userid

userid Specifies the name of a user ID or started task for which you want the information concerning the connection to the VMCF address space.

ISAQ Specifies that you want information pertaining to storage utilization within the VMCF address space.

The parameters are optional. If no parameter is specified, information about all connections to the VMCF address space as well as the storage utilization data is displayed.

Figure 105 on page 839 shows a sample of the output received from issuing the TSO MVPXDISP command with the userid parameter. The messages in this sample are only displayed if the PROFILE MSGID option is in effect for the TSO user ID.
The output from the MVPXDISP command, when it is issued with the `userid` parameter, contains the following information:

**User**
User ID associated with the address space control block (ASCB) owning the connection to the VMCF address space.

**Asid**
Address space ID (ASID) for the user ID.

**Data**
Address of the control block containing extended information about the user ID.

**Sm**
Saved system mask of the user's address space.

**Cr0**
Control register 0 of the user's address space.

**Flags**
Control flags describing the state of the connection. The meaning of the flag bits is as follows:

- `X'80'` SMSG is allowed.
- `X'40'` User ID is a client of VMCF.
- `X'20'` User ID is a client of IUCV.
- `X'10'` User ID is a client of the VMCF address space.
- `X'08'` Reserved.
- `X'04'` User had the TRANSWAP field specified when initially made a client of VMCF.
- `X'02'` Reserved.
- `X'01'` Reserved.

**Client of text string**
Up to 4 lines of text that describe the settings of the bit fields from the Flags variable that concern the client status of the connection. Possible values for text string are:

- VMCF address space
- SMSG
- VMCF
- IUCV

**IUCV mask**
Enable mask used with IUCV communications.

---

**Figure 105. MVPXDISP sample output using the userid parameter**

```plaintext
mvpdisp smtp
EZY2053I MVPXDISP: User SMTP Asid 002C. *****
EZY2054I MVPXDISP: Data @ 15820A00 Sm=FF Cr0=000008E1 Flags=D4.
EZY2055I MVPXDISP: Client of the VMCF address space.
EZY2055I MVPXDISP: Client of SMSG.
EZY2055I MVPXDISP: Client of VMCF.
EZY2056I MVPXDISP: IUCV mask=F8F8, Pending Ctl=0000, Appl=0000.
EZY2057I MVPXDISP: VMCF: Buf=00182BA0, Len=00000118, Fngs=00 User= Key= 80.
EZY2059I MVPXDISP: VMCF: Pending count=0 Flags=00000000.
EZY2059I MVPXDISP: IUCV: Connections=0, Max=255.
EZY2059I MVPXDISP: IUCV: Ctl flags=00000000 Appl flags=00000000.
```

---
Pending Ctl=
  Control pending interrupt mask used with IUCV communications.

Appl=
  Application pending interrupt mask used with IUCV communications.

Buf=
  Address of the VMCF user external interrupt buffer.

Len=
  Length of the VMCF user external interrupt buffer.

Flgs=
  Control flags associated with the VMCF connection. The meaning of the flag bits is as follows:
  X'80'  Specific AUTHORIZE was performed.
  X'40'  Priority messages are allowed.
  X'20'  Connection is in a quiesced state.
  X'1F'  Reserved.

User=
  If a specific AUTHORIZE was performed, the name of the user ID with whom the restricted connection was established; otherwise, a blank field.

Key=
  User key at the time the connection was initialized.

Pending count=
  Count of pending VMCF requests that have been sent.

Flags=
  Control flags associated with pending VMCF requests. Only byte 0 contains defined bit fields. All bit positions in bytes 1 through 3 are reserved. The meaning of the defined flag bits is as follows:
  X'80'  IRB is scheduled or running.
  X'40'  VMCF interrupt might be pending.
  X'3F'  Reserved.

Connections=
  Count of active IUCV connections.

Max=
  Maximum number of IUCV connections allowed.

Ctl flags=
  Control flags associated with pending IUCV requests on the control path. Only byte 0 contains defined bit fields. All bit positions in bytes 1 through 3 are reserved. The meaning of the defined flag bits is as follows:
  X'80'  IRB is scheduled or running.
  X'40'  IUCV interrupt might be pending.
  X'3F'  Reserved.

Appl flags=
  Control flags associated with pending IUCV requests on the application path. Only byte 0 contains defined bit fields. All bit positions in bytes 1 through 3 are reserved. The meaning of the defined flag bits is as follows:
  X'80'  IRB is scheduled or running.
  X'40'  IUCV interrupt might be pending.
  X'3F'  Reserved.

Figure 106 on page 841 shows a sample of the output received from issuing the MVPXDISP command with the ISAQ parameter. The messages in this sample are
only displayed if the PROFILE MSGID option is in effect for the TSO user ID.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVPXDISP</td>
<td>Provides information about dynamic storage areas used by VMCF/TNF/IUCV routines.</td>
</tr>
</tbody>
</table>

The output from the MVPXDISP command, when it is issued with the ISAQ parameter contains the following information:

- **VMCF_Dsa Header at**
  - Address of the anchor block for the dynamic storage areas used by the routines supporting the program call (PC) function.

- **XI_Dsa Header at**
  - Address of the anchor block for the dynamic storage areas used by the VMCF address space while it services cross-memory calls.

- **Subpool**
  - Subpool number from which the storage frames are allocated.

- **1st Getmain count**
  - Count of the number of times a request was made for storage from the pool and none was available. It can be viewed as the maximum number of concurrent requests.

- **2nd Getmain count**
  - Count of the number of times a storage request was made for an area that exceeded the frame size. This value should never be other than zero, since the frame sizes were chosen based on the maximum storage request size that should be made by the various routines.

- **Frame size**
  - Number of bytes (decimal) allocated by a GETMAIN request.

- **Max asked**
  - Largest area in bytes (decimal) that has been obtained from the storage pool to satisfy a request by the routines that exploit the storage pool.

---

Figure 106. MVPXDISP sample output using the ISAQ parameter
Chapter 37. Diagnosing problems with IP CICS sockets

This topic describes how to diagnose IP CICS Sockets problems using the Customer Information Control System (CICS) and contains the following sections:

- "Diagnostic data"
- "Initialization problems" on page 844
- "CICS sockets application problems" on page 846
- "CICS sockets control blocks" on page 847
- "CICS trace" on page 848

CICS is an IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs.

For additional information that might be helpful in solving problems with CICS, refer to the following manuals:

- z/OS Communications Server: IP CICS Sockets Guide
- CICS Diagnosis Reference
- CICS Problem Determination Guide
- CICS Messages and Codes
- z/OS MVS Diagnosis: Tools and Service Aids
- CICS Operations and Utilities Guide

Diagnostic data

To diagnose problems with IP CICS Sockets, some or all of the following data might be required:

- Message logs
  - System log
  - Message log at the transient-data destination specified by the ERRORTD IP CICS Sockets TYPE=CICS configuration option
- CICS external-trace data set (auxtrace)

**Tip:** Using the CICS Trace Control Facility transaction, CETR, ensure the following CICS trace flags are set to obtain the CICS auxiliary trace:

- Set the CICS Master User Trace Flag to the value of ON to generate IP CICS Sockets CICS trace records
- Set the Master System Trace Flag to the value of ON to generate CICS trace records
- Set the AP component trace level to the value of 1

**Rule:** Ensure that CICS tracing is enabled for the IP CICS Socket Interface. If the IP CICS Sockets TYPE=CICS TRACE configuration option is NO then no IP CICS Sockets CICS tracing occurs. Either change the configuration option to enable IP CICS Sockets CICS tracing and then stop and restart the IP CICS Socket Interface or dynamically enable the CICS trace by using the EZAO,START,TRACE command or with the EZAO,SET,CICS transaction specifying TRACE=YES.

- Component trace
  - Engine
- Physical file system (PFS)
- Socket
- Socket (SOCKAPI)
- Transmission control protocol (TCP)

- Dumps
  - CICS address dump, if captured.
  
  **Guideline:** Ensure the following CICS environment before recreating a problem and taking a dump:
  - The CICS internal trace is started
  - The Master System trace flag and Master User trace flag is on
  - Standard trace level 1-2 set for the AP component
  - IP CICS Sockets CICS tracing is enabled
  - Supervisor Call (SVC) dump. SVC dumps are also known as *console dumps* or *system dumps.*
  
  **Guideline:** For hangs and loops, request an SVC dump of CICS, TCP/IP, and the TCPIPDS1 data space.

- NETSTAT SOCKET output
- NETSTAT CONN output

---

**Initialization problems**

This section describes some problems you might encounter when attempting to initialize CICS configured to use IP CICS Sockets.

**Steps for diagnosing CICS socket interface not initialized**

If the CICS socket interface did not initialize, follow the steps below:

1. Issue the EZAO,START,CICS command, and then check that the interface initializes.
   a. If the interface initializes, check that EZACIC20 is in the Program Load Table (DFHPLT).
      
      Putting EZACIC20 into the PLT allows the CICS Socket Interface to initialize on CICS address startup. Refer to the *z/OS Communications Server: IP CICS Sockets Guide* for more information.
   b. If EZACIC20 is defined in the DFHPLT, check the message logs for failures.
   c. If there are no messages, then start CICS with an auxiliary trace active, IP CICS Sockets CICS tracing enabled, and then request an SVC dump of CICS.
   d. Call the Support Center.

2. Verify that the socket Resource Definition Online (RDO) definitions have been properly installed and that the correct data sets are in the STEPLIB and DFHRPL concatenations.

---

**Steps for diagnosing CICS listener not initialized**

If the CICS Listener did not initialize, perform the following steps:

1. Use the EZAC transaction to verify that the listener is defined in the configuration file.
2. In the configuration-file record for that listener, verify that IMMEDIATE is set to YES, and then verify that the correct APPLID and port number are specified.

3. Verify that the listener is properly defined in a CICS RDO group and that the RDO group is in the proper group list.

4. Check the message logs for failures.
   a. If there are no messages, start CICS with auxtrace active IP CICS Sockets CICS tracing enabled, and then request an SVC dump of CICS.
   b. If there are messages, call the Support Center.

5. If an EZY1292E message was issued, investigate why the CICS socket interface did not initialize. (See “Steps for diagnosing CICS socket interface not initialized” on page 844.)

6. If an EZY1369E message was issued, investigate why the TCP/IP stack as specified on the IP CICS Sockets interface TCPADDR configuration option did not initialize. See “Steps for diagnosing problems” on page 844 for steps on diagnosing TCP/IP problems.

No CICS sockets messages issued

If no CICS sockets messages (error or informational) were issued, verify that the correct CICS transient-data queue is specified in the EZACICD TYPE=CICS ERRORTD field in the configuration record for the CICS region. A region is the CICS address space.

Steps for diagnosing TCP/IP clients unable to connect

If TCP/IP clients are unable to connect, perform the following steps.

1. Verify that the listener is active by logging on to CICS, and then issue a CEMT I TASK command. Make sure that the listener name appears in the task list.

2. Verify that the listener is listening on the correct port number by issuing a NETSTAT CONN command, and then check that the listener has the correct port in listen status. Verify that clients are trying to connect to this port and to the correct IP address.

3. Check the ERRORTD log and verify that the EZY1291I message has been issued. If it has not been issued, look for messages indicating a failure.

4. If message EZY1365E is issued then ensure the value specified for the MAXFILEPROC is larger than the listener’s NUMSOCK value. Additionally, ensure the client’s user ID’s FILEPROCMAX setting is appropriately specified.
For more information on how MAXFILEPROC affects tuning applications, refer to [z/OS UNIX System Services Planning](#).

For more information on the FILEPROCMAX specification, refer to the documentation provided for the SAF product in use on your system. If using RACF, this can be found in the [z/OS Security Server RACF Security Administrator's Guide](#).

---

**Steps for diagnosing child-server transactions not starting**

Child-server transactions are transactions started by the listener. If child-server transactions are not starting, perform the following steps.

1. Issue a CEMT I TRANSACTION command to verify that the transaction is installed. If it is not installed, a NOT FND message is displayed.

2. Issue a CEMT I PROGRAM command to verify that the child-server program is installed.

3. If the transaction or program is not installed, define it in the proper RDO group.

4. Check the message logs for failures.

---

**CICS sockets application problems**

This section describes some of the problems you might encounter with CICS sockets applications.

**Steps for diagnosing hung CICS tasks**

If CICS application tasks hang, perform the following steps.

1. While a task is hung, request an SVC dump of CICS, TCP/IP, and the TCPIPDS1 data space.

2. If the problem can be re-created, re-create with CICS auxtrace and component trace turned on.

3. Issue a NETSTAT SOCKET command to determine if the task is waiting on a particular socket call to be posted. If it is waiting, you can issue the NETSTAT DROP command to terminate it.

4. If the application is hung while awaiting completion of a READ command, consider issuing a SELECT or SELECTEX command prior to the READ command. The SELECT command returns either the number of sockets ready to be read or 0 if it times out. The SELECTEX command also returns either the number of sockets ready to be read or 0 if it times out and also returns an ECB or a list of ECBs.
Hung CICS region

If a CICS sockets application program using the Call Instruction API (EZASOKET) is erroneously link-edited without the EZACICAL stub, the entire CICS region might hang while waiting for socket calls to complete. Ensure that EZACICAL is explicitly link-edited with the application.

An EZASOKET call should generate a static call to the EZASOKET entry point within the EZACICAL stub. If the application is not compiled and link edited correctly, the EZASOKET call generates a dynamic call to program EZASOKET, which calls the socket API directly.

Errors on socket calls

If you receive errors on socket calls, note the ERRNO that is received, and then look it up in the section of the z/OS Communications Server: IP CICS Sockets Guide that describes return codes.

A SOCKAPI CTRACE can also help diagnose problems with EZASOKET calls.

CICS shutdown hangs

If an EZY1342I message has been issued, there is a CICS task that has at least one socket open and that is not terminating. You can fix this problem by executing an immediate termination of the CICS socket interface rather than a deferred termination. To execute an immediate termination, issue an EZAO,STOP,CICS command, and then specify YES at the IMMEDIATE prompt.

If you do not add EZACIC20 to the shutdown DFHPLT, CICS cannot terminate because the socket subtasks are still attached to the CICS region. To terminate CICS without EZACIC20, manually shut down the CICS socket interface using the EZAO transaction.

If you have added EZACIC20 to the shutdown DFHPLT then set the IP CICS socket interface PLTSDI configuration option to the value YES to force an immediate shutdown.

CICS sockets control blocks

This section describes some problems you might encounter with the task interface element (TIE) and global work area (GWA). For information about the layout of GWA, TIE, and other control blocks, refer to the section in the z/OS Communications Server: IP CICS Sockets Guide that describes external data structures.

Task interface element

A Task interface element (TIE) represents a CICS task that has issued at least one call to the CICS sockets API. You can locate TIEs in a dump of the CICS region by issuing the IPCS VERBX CICSxxx 'UEH=3' command. CICSxxx is the name of the VERBEXIT used to format a CICS TS dump and is specific to the release of CICS TS that produced the dump. After the CICSxxx VERBEXIT returns, then search for EZACIC01.TIE. The CICSxxx EZACIC01 prefix identifies it as a TIE for CICS sockets. Refer to CICS Problem Determination Guide for more information on the CICS TS VERBEXITS.

The IPCS VERBX CICSxxx 'UEH=3' command output shows a CICS image of the TIE. The TCP/IP TIE is embedded within the CICS image of the TIE and starts at offset +X'80'.
The IPCS VERBX CICSxxx ‘UEH=3’ command output contains TIEs for other interfaces as well.

**Global work area**

The GWA is the main anchor point for the CICS socket interface. It contains general status data, work areas, and pointers to other control-block chains. You can locate the GWA in a dump of the CICS region by issuing the IPCS VERBX CICSxxx ‘UEH=3’ command, and searching for EZACIC01.GWA. The EZACIC01 prefix identifies it as the GWA for CICS sockets.

**CICS trace**

The CICS sockets task related user exit (TRUE), EZACIC01, issues CICS trace entries at the following four points of execution:

- When the TRUE receives a socket call from an application
- When the TRUE is passing the socket call to the subtask
- When the TRUE receives the response from the subtask
- When the TRUE is ready to return its response to the application

The trace point ID is AP 00C7. Trace records are self-explanatory. They show the type of call, the point of execution, the ERRNO, and the RETCODE.

**Steps for displaying the internal trace**

Trace records can be written either to a CICS internal trace table or to its external-trace data set (auxtrace). Perform the following steps to display the internal trace, follow these steps.

1. Request a dump of the CICS region using the RGN SDATA=(option 1,option 2,...option n) parameter on a DUMP command. Examples of options are CSA, PSA, NVC, RGN, TRT, SQA, LSQA, LPA, and so on. For a complete list of options, refer to [z/OS MVS Diagnosis: Tools and Service Aids](#).

2. Display the trace using the IPCS VERBX CICSxxx ‘UEH=3’ command.

   **Tip**: CICS trace can also be directed to the GTF trace data set.

To display the auxtrace, follow the instructions for formatting auxtrace as documented in the [CICS Operations and Utilities Guide](#).
Chapter 38. Diagnosing problems with Express Logon

The Express Logon feature in Communications Server for z/OS allows a user on a workstation, with a TN3270E client and an X.509 certificate, to log on to an SNA application without entering an ID or password.

This topic describes how to diagnose problems using Express Logon for the z/OS Communications Server Express Logon feature, including the Digital Certificate Access Server (DCAS). It contains the following sections:

- “Analyzing start problems with the DCAS” on page 850
- “Analyzing client interface problems” on page 851

For complete information about Express Logon, refer to the following:

- z/OS Communications Server: IP Configuration Guide
- z/OS Security Server RACF Security Administrator’s Guide

For most situations in which the DCAS does not start, a message to the console is displayed. If the explanation in z/OS Communications Server: IP and SNA Codes does not help, you should turn on debugging and logging. You can specify debugging and logging as startup parameters from the z/OS UNIX shell or from the MVS console as a started procedure:

- If the DCAS is started from the z/OS UNIX shell, you can specify the following:
  ```
  dcas -d <debugging_level> -l <logtype>
  ```
- If the DCAS is started from the MVS console, you can specify debugging and logging on the PARM statement after the final slash, as shown in the following example:
  ```
  //DCAS PROC
  //*
  //DCAS EXEC PGM=EZADCDMN,REGION=4096K,TIME=NOLIMIT,
  // PARM='POSIX(ON) ALL31(ON) / -d 1 SYSLOGD'
  ```

The following optional parameters can be used with both DCAS UNIX commands and MVS started procedures:

- **d or -D**
  Indicates debugging. The following levels apply:
  1. Specifies log error and warning messages.
  2. Specifies log error, warning, and informational messages.
  3. Specifies log error, warning, informational, and debug messages.

  The default level is 3.

- **l or -L**
  Indicates logging to SYSLOGD or to a designated log file. If you do not specify this parameter, logging defaults to /tmp/dcas.1og.

  If you specify a debug level, but not logging, the DCAS attempts to open the default log file /tmp/dcas.1og. If this fails, debugging is turned off.

  For SYSLOGD, the DCAS uses the log facility local0.
Tracing can be turned on and off while DCAS is running by sending it a SIGHUP signal. The SIGHUP signal toggles tracing on and off. So, if a SIGHUP is sent while tracing is on, tracing is turned off. And if a SIGHUP is sent while tracing is off, tracing will be turned on. A SIGHUP signal can be sent from the z/OS UNIX shell by issuing:
```
kll -HUP `cat /tmp/dcas.pid`
```
An accent mark (') is used in the definition above, not a single quotation mark.

For further aid in diagnosing errors, refer to the error logs of the TN3270E middle-tier servers. Also, examine the HOD client security message panel.

The following `netstat` commands, issued from the middle-tier server, are useful in determining connectivity problems between z/OS Communications Server and DCAS.

For AIX, the `netstat` command is:
```
netstat -an | grep port#
```
For CS/2, the `netstat` command is:
```
netstat -sn | grep port#
```
For NT, the `netstat` command is:
```
netstat -an | more port#
```
In the `netstat` commands, port# is the listening port of DCAS. The default DCAS port is 8990.

### Analyzing start problems with the DCAS

When analyzing problems that occur when starting the DCAS, consider the following:

- The DCAS must run from an APF Authorized library.
- The DCAS uses z/OS Language Environment C run-time services. Make sure that the Language Environment C run-time library is compatible with the current level of z/OS Communications Server.
- The DCAS uses SSL cryptographic services run-time library. Verify that `hlq.SYSL.SIEALNKE` is accessible at run time. If certificates are authenticated using the X.500 host, SSL uses LDAP services to access the X.500 host. If running from the z/OS UNIX shell, verify that the LIBPATH environment variable includes `/usr/lib`.
- The DCAS attempts to initialize SSL services. If you are using key rings that reside in the z/OS UNIX file system, verify that the KEYRING and STASHFILE keywords in the DCAS configuration file point to valid z/OS UNIX file system file names. Names are case sensitive. If using key rings that reside in RACF, verify that the SAFKEYRING keyword in the DCAS configuration file references a valid RACF key ring.
- The DCAS must be associated with a valid user ID using z/OS UNIX services. It must run with the POSIX(ON) C run-time option. Use the following RACF command:
  ```
  ADDUSER dcasid DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/'))
  ```
- If the DCAS is started as an MVS started procedure, verify that the following RACF commands have been issued:
The DCAS uses the TCP/IP protocol to communicate with clients in the network. Verify that the z/OS Communications Server products VTAM and TCP/IP have been started and are active.

**Analyzing client interface problems**

When analyzing problems with client interfaces, consider the following:

- DCAS uses the TCP/IP protocol to communicate with its clients, the TN3270 middle-tier servers. Verify that the z/OS Communications Server products VTAM and TCP/IP have been started and are active. To verify network connectivity to a client, try pinging that client.

- The DCAS uses RACF services to obtain a user ID given a digital certificate.
  - Verify the certificate has been defined properly to RACF. Use the following commands:
    
    ```
    SETROPTS CLASSACT(DIGTCERT)
    SETROPTS RACLIST(DIGTCERT) REFRESH
    PERMIT IRR.DIGTCERT.function CLASS(FACILITY) ID(dcasid) ACCESS(CONTROL)
    RACDCERT ID(userid) ADD('certificate dataset name') TRUST
    
    - Verify that the user ID associated with the DCAS has permission to access certificates. Use the following RACF commands:
      
      ```
      SETROPTS CLASSACT(DIGTCERT)
      SETROPTS RACLIST(DIGTCERT) REFRESH
      PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(dcasid) ACCESS(CONTROL)
      ```

- The DCAS uses RACF services to obtain a PassTicket for an associated application ID. Verify that the RACF PTKTDATA profile for the application ID has been defined properly. The ID must match the ID specified on the workstation client. For HOD V5, this is the name specified in the Express Logon Application ID pop-up window. It might not be the same name specified on the USSMSG10. For applications such as TSO, specifying the application ID can be difficult since the profile name has special RACF considerations. Refer to the [z/OS Security Server RACF Security Administrator's Guide](#).

  Use these commands to verify the RACF PTKTDATA profile:

  ```
  SETROPTS CLASSACT(PTKTDATA)
  RDEFINE profile PTKTDATA SSIGNON()
  SETROPTS RACLIST(PTKTDATA) REFRESH
  ```
Chapter 39. Diagnosing resolver problems

This topic describes how to diagnose resolver problems and contains the following sections:

- “Steps for resolving the hostname”
- “TRACE RESOLVER” on page 857
- “CTRACE — RESOLVER” on page 876

The resolver provides two kinds of tracing plus an IPCS subcommand to help analyze resolver problems in dumps. The resolver provides TRACE RESOLVER information that can be helpful in debugging problems an application program could have with using resolver facilities (for example, GetHostBy* or GetHostByAddr). Component Trace is used for tracing the RESOLVER component (SYSTCPRE) for diagnosing resolver problems that cannot be isolated to one particular application. Use the IPCS RESOLVER subcommand to format and summarize resolver control blocks (see “RESOLVER” on page 299).

Refer to the z/OS Communications Server: IP Configuration Reference for additional information.

Steps for resolving the hostname

Before you begin: You need to know the exact hostname that failed to resolve and the environment in which the application was running (for example, TSO, UNIX, or batch).

1. Diagnose why the hostname failed to resolve by pinging the hostname. Base your next course of action on the following conditions:

<table>
<thead>
<tr>
<th>If ping for the hostname...</th>
<th>Then...</th>
<th>Solution</th>
</tr>
</thead>
</table>
| Succeeds, but another application fails when resolving the same hostname | One or more of the following may be the problem:  
- The resolver configuration for the application in the users environment.  
- The resolver cache has information about the hostname, and one application is able to access the resolver cache and the other is not.  
- The resolver cache has saved different information about the hostname as provided by different name servers, and the applications are using different cached information. | Use the Trace Resolver to solve the problem. |
If ping for the hostname... Then... Solution

| Fails, but the hostname is converted to an IP address | The resolution is successful but the host is not reachable or active. | See Chapter 4, “Diagnosing network connectivity problems,” on page 29 to continue researching the problem. |
| Fails to convert the name to an IP address | The problem might be with the resolver configuration, querying the resolver cache, searching local host files, or using DNS. | Use Trace Resolver to solve the problem. **Note:** You can use the LOOKUP option in TCPIP.DATA to specify local searching before or instead of asking DNS. |

2. Determine if the name or address being queried is known to DNS if you expect to resolve the hostname using DNS.

The following example looks for the name www.johndoe.com at IP address 1.2.3.4:

dig@1.2.3.4 www.johndoe.com -t any

The command should return all resource records of any type from the DNS at 1.2.3.4 for www.johndoe.com. For more information about dig, see z/OS Communications Server: IP System Administrator’s Commands.

3. If dig does not return all resource records, base your next course of action on the following conditions:

<p>| If dig... Then... | Solution |
| Fails because it cannot contact DNS | You need to check your link to the DNS IP address. | See Chapter 4, “Diagnosing network connectivity problems,” on page 29 to continue researching the problem. |
| Fails because DNS reports that the resource was not found | <a href="http://www.johndoe.com">www.johndoe.com</a> is not a resource record known to DNS. | See the DNS administrator to add the name. As a temporary work around, you might want to add the name to a local host file that the Resolver searches. Refer to z/OS Communications Server: IP Configuration Guide for information about local host files. |</p>
<table>
<thead>
<tr>
<th>If dig...</th>
<th>Then...</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Succeeds</td>
<td>The problem in resolving the hostname using ping or another application might be in configuring the resolver, or might involve the contents of the resolver cache.</td>
<td>The <strong>dig</strong> command bypasses the resolver cache, search orders, local host files, and domain names appended by the Resolver. The best way to check the configuration is to start the Trace Resolver. It is important to use the Trace Resolver in the environment where the application is failing because the application might be using a different TCPIP.DATA file, environment variables, or search order than the environment where the <strong>dig</strong> command was issued. The application may also be accessing the resolver cache, which may have inaccurate or outdated information. See &quot;Steps for resolving caching problems&quot; for more cache information.</td>
</tr>
</tbody>
</table>

You know you are done when the application that previously failed to resolve the hostname can now resolve it.

**Steps for resolving caching problems**

**Before you begin:** You need to know the exact hostname that is suspected to have inaccurate information in the resolver cache, and the environment in which the application was running (for example, TSO, UNIX, or batch).

1. Determine whether the resolver cache contains any information about the host name, using the Netstat RESCache/-q report. The following command can be used to display information about the host name (which in this example is `www.johndoe.com`):
   ```
   netstat -q DETAIL -H www.johndoe.com
   ```
   The command should display all entries that exist in the resolver cache because of hostname-to-IP address resolution requests for `www.johndoe.com`. For more information about the Netstat RESCache/-q report, see [z/OS Communications Server: IP System Administrator's Commands](https://publib.boulder.ibm.com/infocenter/comserver/v7r1/topic/com.ibm.netstat.ref.doc/).

2. Determine, using the **dig** command, if the name or address being queried is known to the DNS name server if you expect to resolve the hostname using DNS.
   The following example looks for the name `www.johndoe.com` at IP address 1.2.3.4:
   ```
   dig@1.2.3.4 www.johndoe.com -t any
   ```
The command should return all resource records of any type from the DNS at 1.2.3.4 for www.johndoe.com. For more information about dig, see z/OS Communications Server: IP System Administrator’s Commands. The DNS name server to use for this search is the first name server listed in the NSINTERADDR list of name servers in the TCPIP.DATA dataset used by the application experiencing the problems. For more information about NSINTERADDR, see z/OS Communications Server: IP Configuration Reference.

3. Base your next course of action on the results of the dig command:

<table>
<thead>
<tr>
<th>If . . .</th>
<th>Then . . .</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dig fails because it cannot contact DNS</td>
<td>You need to check your link to the DNS IP address.</td>
<td>See Chapter 4, “Diagnosing network connectivity problems,” on page 29 to continue researching the problem.</td>
</tr>
<tr>
<td>Dig fails because DNS reports that the resource was not found</td>
<td><a href="http://www.johndoe.com">www.johndoe.com</a> is not a resource record known to DNS.</td>
<td>See the DNS administrator to add the name.</td>
</tr>
<tr>
<td>If dig succeeds, and the resolver cache has different information from what was returned by dig</td>
<td>The cache information might be information that was provided by a different name server, or might represent old information that was assigned a time-to-live (TTL) value that was excessive. The application might be acquiring inaccurate information about the hostname due to the resolver cache data.</td>
<td>Because the dig command bypasses the resolver cache, the dig command is unaffected by the resolver cache information. To remove the cache information, issue the MODIFY RESOLVER,FLUSH,ALL command to delete all entries from the cache. For more information about MODIFY FLUSH processing, see z/OS Communications Server: IP System Administrator’s Commands.</td>
</tr>
<tr>
<td>If dig succeeds, but the resolver cache has the same information as dig, or has no information about the hostname</td>
<td>The problem in resolving the hostname using ping or another application might be in configuring the resolver.</td>
<td>The dig command bypasses the resolver cache, search orders, local host files, and domain names appended by the resolver. The best way to check the configuration is to start the trace resolver. It is important to use the trace resolver in the environment where the application is failing because the application might be using a different TCPIP.DATA file, environment variables, or search order than the environment where the dig command was issued.</td>
</tr>
</tbody>
</table>

You know you are done when the application that previously failed to resolve the hostname can now resolve it.
The Trace Resolver tells what the Resolver looked for (the Questions) and where it
looked (name servers’ IP addresses or local host file names). Check the following
in the trace output:

- Fix or check any problems reported at the top of the trace. These are errors in
  the resolver data sets.
- Are the data sets being used by the resolver the ones you expected? If not, see
  the search orders for data sets in the z/OS Communications Server: IP Configuration
  Guide.
- Check that the expected MVS data sets or UNIX file system files are accessible
  by the user or batch job. Errors detected by a security product (for example,
  RACF) or OPEN services can generate messages that help indicate the problem.
  For example, IEC1411 013-C0 can be generated if a file does not have the correct
  permission bit settings to allow it to be read. RACF message ICH408I can be
  issued if no OMVS segment is defined or if insufficient authorization is granted
  to read a data set. Refer to z/OS Communications Server: IP Configuration Guide
  for more information about security product and file permission bit values.
- Check the TCPIP.DATA parameter values, especially SEARCH, NAMESERVER,
  NSINTERADDR, NOCACHE, and NSPORTADDR. TCPIP.DATA parameters are
  explained in z/OS Communications Server: IP Configuration Reference.
- Check the questions posed by the Resolver to DNS or in searching the local host
  files. Are these the queries you expected?
- Check for the cache query and cache add attempts being attempted by the
  resolver as part of processing a resolver query. Are these the queries you
  expected, and the results that you anticipated for those requests? Are the results
  from the cache-query attempts returning the information that you expected to be
  returned for those target resources? If you suspect that there are problems with
  the operations of the resolver cache, you will need to collect CTRACE records
  for further diagnosis. See “CTRACE — RESOLVER” on page 876 for more
details.
- Look for errors or failures in the trace.
- Did DNS respond (if you expected it to)? If not, see if DNS is active at the IP
  address you specified for NAMESERVER and NSINTERADDR and what port it
  is listening on. Also DNS logs can be helpful. Ask the DNS administrator for
  help.

**Tips:** The resolver supports the Extension Mechanisms for DNS (EDNS0)
standards, which permits DNS messages of greater than 512 bytes to be returned
by DNS to the resolver; however, some network routers are configured to
silently discard DNS messages of greater than 512 bytes. If the trace resolver
suggests that DNS did not respond, verify that no routers between the resolver
and DNS are discarding the messages.

The resolver dynamically determines the EDNS0 capability of each DNS, based
on responses or timeouts to DNS queries. The current view of the DNS
capabilities are included in trace resolver output. A simple way to examine
the trace resolver output is to issue the Netstat HOme/-h command. Use the trace
resolver information to verify that the resolver is using proper EDNS0
processing for a given DNS. Issue the MODIFY REFRESH command to force the
resolver to dynamically relearn the EDNS0 capability of the name servers.
Consider adjusting RESOLVERTIMEOUT values if timeout conditions cause the
resolver to mistakenly avoid EDNS0 processing for a given DNS.
- The following are some common misunderstandings:
- If the queried name server returns NXDOMAIN, the resolver does not continue to the next name server in the list. NXDOMAIN means the domain does not exist according to that name server.

- The resolver only appends the specific names listed in the Search (or Domain) parameter. It does not attempt shorter versions of these. For example, if you look for “johndoe” and your search list has “anywhere.usa.com”, the resolver looks for “johndoe.anywhere.usa.com” and “johndoe” (the order depends on the value of option ndots). The Resolver does not look for “johndoe.anywhere” or “johndoe.anywhere.usa” or “johndoe.usa.com” or “johndoe.com”.

- The contents of any local hosts files are not cached in the system-wide resolver cache, but are saved separately for each task.

- Negative cache entries are created to represent the following responses from a name sever:
  - A response with a return code value of NXDOMAIN. This typically represents a host name that has no records of any type in the specified domain.
  - A response with a return code value of NOERROR when no answer records are returned. This represents a host name that does not have any records of the type that was requested (A, AAAA, etc.) in the specified domain, but does have some records of a different type.
  - A response with a return code value of NOERROR when the answer records returned represent canonical, or alias, names. This represents a resource that is officially known by other names in the specified domain, does not have any records of the type that was requested (A, AAAA, etc.), but does have some records for a different type of resource.

Activate Trace Resolver output in one of the following ways:

- Specify the z/OS UNIX RESOLVER_TRACE environment variable or a SYSTCPT DD allocation. Specifying the RESOLVER_TRACE environment variable or allocating the SYSTCPT DDname dynamically activates Trace Resolver output regardless of the TCPIP:DATA or the _res structure resDebug specification. Dynamic activation of Trace Resolver can be useful when you are not sure where the TCPIP:DATA statements might be found.

- Specify the TCPIP:DATA statement TRACE RESOLVER or OPTIONS DEBUG. When using a TCPIP:DATA statement to activate the trace, have the trace activation statement as your very first statement. This ensures that the trace is in effect for all statements in the TCPIP:DATA specification.

- Set the debug option (resDebug) in an application _res structure.

The resolver uses the following search order to determine if Trace Resolver output is necessary. The Trace Resolver data is contained in the specified output location. If the output location is not available for writing, the next search location is used. The default location for the Trace Resolver output in the z/OS UNIX environment is stdout. In the native MVS environment, it is as specified by the SYSPRINT DD.

1. The RESOLVER_TRACE environment variable (z/OS UNIX environment only).
2. The SYSTCPT DD allocation.
3. The TRACE RESOLVER or OPTIONS DEBUG statements. You must allocate STDOUT or SYSPRINT to generate trace data. The allocations need to exist in all operating environments including TSO, for example, your TSO Logon Procedure.
4. The resDebug bit set to on in the _res structure option field. STDOUT or SYSPRINT must be allocated or no trace data is generated.
Trace Resolver output can be written to any of the following:
- A TSO user terminal screen
- z/OS UNIX STDOUT
- JES SYSOUT
- An MVS Sequential data set (a member of a PDS is not supported). The data set must already exist or be allocated as new with the following DCB characteristics:
  - An LRECL between 80 and 256 with a RECFM of Fixed Block.
  - For an LRECL of 128 or larger, the last six print positions are the storage address of the MVS TCB that issued the resolver call. This can be helpful with multitask applications.
- A z/OS UNIX file system file. The file can either be an existing file or be dynamically allocated by the resolver when needed. The maximum line length used in the file is 255 characters. The last six print positions are the storage address of the MVS TCB that issued the resolver call. This can be helpful with multitask applications.

If the Trace Resolver output uses an MVS data set or z/OS UNIX file system file, the output is for the resolver services invoked by the last command or UNIX process. If possible, use SYSOUT=* or z/OS UNIX STDOUT to trace multiple resolver service invocations (for example, a multitask environment).

**Specifying the Trace Resolver output location**

Your environment determines the method to specify the Trace Resolver output location. This section includes the following environments:
- TSO
- z/OS UNIX
- MVS batch job
- z/OS UNIX batch

**TSO environment**

In the TSO environment, use one of the following to specify the Trace Resolver output location:
- For the user’s terminal, enter the following:
  ```shell
  alloc dd(systcpt) da(*)
  ```
  When directing Trace Resolver output to a TSO terminal, define the screen size to be only 80 columns wide. Otherwise, trace output is difficult to read.
- For an existing MVS data set, enter the following:
  ```shell
  alloc dd(systcpt) da(appl.restrace)
  ```
  The user ID is used as the first qualifier for the data set. For example, if TSO USER1 entered the above command, user1 would be appended to the data set, as shown below:
  ```shell
  alloc dd(systcpt) da('user1.appl.restrace')
  ```
  To disable the Trace Resolver output, enter the following:
  ```shell
  free dd(systcpt)
  ```

**z/OS UNIX shell environment**

In the z/OS UNIX shell environment, use one of the following to specify the Trace Resolver output location:
- For STDOUT, enter the following:
  ```shell
  export RESOLVER_TRACE=STDOUT
  ```
If needed, you can redirect STDOUT when the z/OS UNIX command is issued.

If your application was compiled with the z/OS C/C++ Language Environment Native ASCII support do not use STDOUT. If you use STDOUT with ASCII programs the trace data is not readable. Instead send the trace data to an MVS data set or z/OS UNIX file system file as described below.

- For a new z/OS UNIX file system file or existing MVS data set, enter the following:
  
  ```
  export RESOLVER_TRACE=/tmp/myjob.resolv.trace
  export RESOLVER_TRACE="/appl.restrace"
  ```

  The user ID is used as the first qualifier for the data set. For example, if USER3 entered this command, user3 would be appended to the data set, as follows:

  ```
  export RESOLVER_TRACE="/user3.appl.restrace"
  ```

  To disable the Trace Resolver output, enter the following:

  ```
  set -A RESOLVER_TRACE
  ```

- For a z/OS UNIX file system file or an MVS data set that is already allocated to a ddname:

  ```
  export RESOLVER_TRACE="/dd:ddname"
  ```

  or

  ```
  export RESOLVER_TRACE="dd:ddname"
  ```

**MVS batch job environment**

In the MVS batch job environment, to use the recommended JES SYSOUT, enter the following:

```
//SYSTCPT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
``` 

You must allocate either SYSTCPT or SYSPRINT DD if the TCPIP.DATA, statements TRACE RESOLVER or OPTIONS DEBUG, are specified. If neither are allocated, then no trace output is written.

**z/OS UNIX batch environment**

In the z/OS UNIX batch environment, use one of the following methods to specify the Trace Resolver output location:

- If the application resides in a z/OS UNIX file system file, use BPXBATSL to run the program. In this way, DD allocations is passed to the application. If the application does fork, the DD allocations are not passed to the new process, and the Trace Resolver output cannot be collected.

- To use the recommended JES SYSOUT, enter the following:

  ```
  //SYSTCPT DD SYSOUT=*  
  ```

- Because STDOUT cannot be allocated to SYSOUT=* with BPXBATSL, use one of the following STDOUT DD JCL statements shown below:

  ```
  //STDOUT DD DISP=SHR,DSN=USER3.APPL.RESTRACE  
  ```

  ```
  //STDOUT DD PATH="/tmp/appl.stdout",  
  ```

  ```
  /PATHOPTS=(OWRONLY,OCREATE),  
  ```

  ```
  //PATHMODE=SIRWXU  
  ```

**Note:** In this example, OTRUNC is not specified on the PATHOPTS statement. This means the Trace Resolver output is appended to the z/OS UNIX file system file. To avoid z/OS UNIX file system full conditions, manually delete trace output that is no longer needed to ensure that the file does not fill the specified directory (for example, /tmp/).
You must allocate either SYSTCPT or SYSPIRT DD if the TCPIP.DATA statements, TRACE RESOLVER or OPTIONS DEBUG, are specified. If neither are allocated, then no trace output is written.

To pass the RESOLVER_TRACE environment variable using BPXBATSL or BPXBATCH, enter the following:

```bash
//STDENV DD JCL statement
```

The following shows an example:

```bash
//STDENV DD DISP=SHR,DSN=USER3.APPL..ENVIRON
```

The STDENV data set can be a fixed or variable (nonspanned) record format type. It can contain multiple environment variables, as shown in the following sample:

```bash
RESOLVER_TRACE=/'USER3.APPL.RESTRACE'
_BPXK_SETIBMOPT_TRANSPORT=TCPCS
```

**Notes:**

1. Environment variables must start in column 1, and the data set must not contain any sequence numbers because they would be treated as part of the environment variable.
2. For the RESOLVER_TRACE environment variable, any blanks from a fixed format STDENV data set is removed. Because this might not be true for all variables, a variable record format data set is recommended.
3. For applications that fork, use of an MVS data set is recommended. If you use a z/OS UNIX file system file, a C03 ABEND might occur when the forked process ends.

**Interpreting the Trace Resolver output**

The following is an example showing the setup files used, the command used to invoke the trace, and the trace resolver output:

- **Setup files used for trace resolver:**
  - **Resolver Procedure:**
    ```bash
    //RESOLVER PROC PARMS='CTRACE(CTIRESFL)'
    ```
  - **Setup File TPOUSER.RESOLVER.SETUP.DATA contains:**
    ```bash
    ; DEFAULTTCPIPDATA('TPOUSER.RESOLVER.DEFAULT.DATA')
    ; GLOBALTCPIPDATA('/ETC/TCPIPGLOBAL.DATA')
    # GLOBALTCPIPDATA('SYS1.TCPPARMS(RESGLOBL)')
    ```
  - **Global TCPIP.DATA file SYS1.TCPPARMS(RESGLOBL) contains:**
    ```bash
    # Note that DOMAIN is ignored because SEARCH is mutually exclusive
    # and SEARCH appears after DOMAIN.
    Domain abckxyz
    ; Note that SEARCH can be specified on multiple lines.
    SEARCH tcp.raleigh.ibm.com raleigh.ibm.com
    SEARCH ibm.com com uk
    SEARCH gov
    ```

1a Search m1l
SORTLIST 0.0.19.0/0.0.255.0 0.0.18.99/0.0.255.255 0.42.17.0/0.255.255.0
SORTLIST 129.42.16.0/255.255.255.0
1b Sortlist 9.0.0.0
NSInterAddr 9.67.128.82 Buzz
NameServer 9.67.128.255 not a server
Default TCP/IP.DATA file TPOUSER.RESOLVER.DEFAULT.DATA contains:

; TRACE RESOLVER
DatasetPrefix USER1
TcpipJobname TCPCS3
Hostname VIC097
; trace c sockets
; alwayswto no
; messagecase whoknows
; loaddbcstables tbd

Note: For this example, this file exists but is not used in the procedure for obtaining this example trace resolver output.

Local TCP/IP.DATA file USER55.TCPIP.DATA contains:

; trace resolver
DATASETPREFIX USER55
# If an option is coded multiple times but can only have 1 value,
# the last occurrence is used.
TCPIpJobname TCPCS2
TCPIpJobname TCPCS
Hostname MVS000
DomainOrigin edu ;
NameServer 127.0.0.1 ; loopback
# ResolveVia TCP
ResolverTimeout 22
alwayswto xyz
messagecase mixed
loaddbcstables schinese

- TSO commands issued to obtain the trace (gethostbyname):
  alloc dd(systcpt) dsn(traceres) reuse
  invoke a REXX application which issues gethostbyname for www.ibm.com

- Trace Resolver output in USER55.TRACERES contains (gethostbyname):
  res_init Skipped option(s) on line 8: SYS1.TCPPARMS(RESGLOBL)
  res_init Skipped option(s) on line 11: SYS1.TCPPARMS(RESGLOBL)
  res_init Parse error on line 18: SYS1.TCPPARMS(RESGLOBL)
  res_init Parse error on line 14: USER55.TCPIP.DATA

  res_init Resolver values:
  Global Tcp/Ip Dataset = SYS1.TCPPARMS(RESGLOBL)
  Default Tcp/Ip Dataset = TPOUSER.RESOLVER.DEFAULT.DATA
  Local Tcp/Ip Dataset = USER55.TCPIP.DATA
  Translation Table = Default
  UserId/JobName = USER55
  Caller API = TCP/IP Rexx Sockets
  Caller Mode = EBCDIC
  (L) DataSetPrefix = USER55
  (G) HostPrefix = MvS026
  (G) TcpIpJobName = TCPCS
  (G) Search = tcp.raleigh.ibm.com
  raleigh.ibm.com
  ibm.com
  com
  uk
Gov

(G) SortList = 0.0.19.0/0.0.255.0
0.0.18.99/0.0.255.255
0.42.17.0/0.255.255.0
129.42.16.0/255.255.255.0

(G) NameServer = 127.0.0.1, 9.67.128.255
EDNS0 Support = Unknown

(G) NsPortAddr = 53
(G) ResolverTimeout = 3

(*) ResolveVia = UDP
(*) Options NDots = 1
(*) SockNoTestStor
(*) AlwaysWto = NO
(L) MessageCase = MIXED

(G) LoadDbcsTable = BIG5

res_init Succeeded
res_init Started: 2008/11/08 20:18:29.893218

GetHostByName Started: 2008/11/08 20:18:29.908175
GetHostByName Resolving Name: TESTBEN46.SVT390.COM
GetHostByName Stack Name: TCPCS
res_search(TESTBEN46.SVT390.COM, C_IN, T_A)
res_search Host Alias Search found no alias
res_querydomain(TESTBEN46.SVT390.COM, C_IN, T_A)

res_query(TESTBEN46.SVT390.COM, C_IN, T_A)

Querying resolver cache for TESTBEN46.SVT390.COM.
EZRRECFR: RetVal = 0, RC = 0, Reason = 0x00000000
No cache information was available
res_mkquery(QUER Y, TESTBEN46.SVT390.COM, C_IN, T_A)

res_mkquery created message:

*** Beginning of Message ***

Query Id: 1571
Flags: 00000001 00000000
Flags set: recurDes
OpCode: QUERY
Response Code: NOERROR

Number of Question RRs: 1
Question 1:

TESTBEN46.SVT390.COM
Type (0X0001) T_A Class (0X0001) C_IN

Number of Answer RRs: 0
Number of Authority RRs: 0

Additional 1:

Type (0X0029) T_OPT UDP Payload (0X0C00) 3072
Extended RCODE 0 Version 0 Flags 0000

*** End of Message ***

res_send Name Server Capabilities
Name server 127.0.0.1 EDNS0 Support = unknown
Name server 9.67.128.255 EDNS0 Support = unknown

res_send Sending query to Name Server 127.0.0.1
BPXISOC: RetVal = 0, RC = 0, Reason = 0x00000000

No OPT RR record sent on request to 127.0.0.1
BPXISTO: RetVal = 38, RC = 0, Reason = 0x00000000
BPXIAO Sched: RetVal = 1, RC = 0, Reason = 0x00000000
BPXIAO RECVMSG from 127.0.0.1: RetVal=502, RC=0, Reason=0x00000000
UDP Data Length: 502

res_send received data via UDP. Message received:

*** Beginning of Message ***

Query Id: 1571
Flags: 10000111 10000000
Flags set: auth trunc recurDes recurAvl
OpCode: QUERY
Response Code: NOERROR

Number of Question RRs: 1
Question 1:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN

Trace terminated due to truncation condition
* * * * End of Message * * * *

DNS Communication Ended: 2008/11/08 20:18:29.916671 time used 00:00:00.006655
res_send Sending query to Name Server 127.0.0.1
EDNS0 Probe request sent to 127.0.0.1 id=1572
BPXIO: RetVal = 49, RC = 0, Reason = 0x00000000
BPXIAIO Sched: RetVal = 1, RC = 0, Reason = 0x00000000
BPXIAIO RECVMSG from 127.0.0.1: RetVal=553, RC=0, Reason=0x00000000
UDP Data Length: 553
res_send received data via UDP. Message received:
* * * * = Beginning of Message * * * *
Query Id: 1572
Flags: 10000101 10000000
Flags set: resp auth recurDes recurAvl
OpCode: QUERY
Response Code: NOERROR

Number of Question RRs: 1
Question 1:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN

Number of Answer RRs: 29
Answer 1:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.7
Answer 2:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.8
Answer 3:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.9
Answer 4:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.10
Answer 5:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.11
Answer 6:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.12
Answer 7:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.13
Chapter 39. Diagnosing resolver problems 865
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.27
Answer 22:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.28
Answer 23:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.29
Answer 24:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.30
Answer 25:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.31
Answer 26:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.32
Answer 27:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.33
Answer 28:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.34
Answer 29:
TESTBEN46.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.14.103.35

Number of Authority RRs: 1
Authority 1:
SVT390.COM
Type (0x0002) T_NS Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
sdistcici.SVT390.COM

Number of Additional RRs: 2
Additional 1:
sdistcici.SVT390.COM
Type (0x0001) T_A Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.11.235.51
Additional 2:
Type (0x0029) T_OPT UDP Payload (0x1000) 4096
Extended RCODE 0 Version 0 Flags 0000

* * * * * End of Message * * * * *

866 z/OS V1R11.0 Comm Svr: IP Diagnosis Guide
Cache information was saved
res_query Succeeded
res_querydomain Succeeded
res_search Succeeded
GetHostByName Succeeded: IP Address(es) found:
IP Address(1) is 10.14.103.7
IP Address(2) is 10.14.103.8
IP Address(3) is 10.14.103.9
IP Address(4) is 10.14.103.10
IP Address(5) is 10.14.103.11
IP Address(6) is 10.14.103.12
IP Address(7) is 10.14.103.13
IP Address(8) is 10.14.103.14
IP Address(9) is 10.14.103.15
IP Address(10) is 10.14.103.16
IP Address(11) is 10.14.103.17
IP Address(12) is 10.14.103.18
IP Address(13) is 10.14.103.19
IP Address(14) is 10.14.103.20
IP Address(15) is 10.14.103.21
IP Address(16) is 10.14.103.22
IP Address(17) is 10.14.103.23
IP Address(18) is 10.14.103.24
IP Address(19) is 10.14.103.25
IP Address(20) is 10.14.103.26
IP Address(21) is 10.14.103.27
IP Address(22) is 10.14.103.28
IP Address(23) is 10.14.103.29
IP Address(24) is 10.14.103.81
IP Address(25) is 10.14.103.32
IP Address(26) is 10.14.103.33
IP Address(27) is 10.14.103.34
IP Address(28) is 10.14.103.35
IP Address(29) is 10.14.103.36
GetHostByName Ended: 2008/11/08 20:18:29.930036
***************************************************************************
TSO commands issued to obtain the trace (getaddrinfo):
alloc dd(systcpt) dsn(traceres) reuse
ping cs390-2e
Ping CS V1R5: Pinging host CS390-2E.tcp.raleigh.ibm.com
sendto(): EDC8130I Host cannot be reached.
***************************************************************************
Trace Resolver output in USER55.TRACERES contains (getaddrinfo):
Resolver Trace Initialization Complete -> 2008/11/08 20:30:11.315660
res_init Skipped option(s) on line 8: SYS1.TCPPARMS(RESGLOBL)
res_init Skipped option(s) on line 11: SYS1.TCPPARMS(RESGLOBL)
res_init Parse error on line 18: SYS1.TCPPARMS(RESGLOBL)
res_init Parse error on line 14: USER55.TCPIP.DATA
res_init Resolver values:
Global Tcp/Ip Dataset = SYS1.TCPPARMS(RESGLOBL)
Default Tcp/Ip Dataset = TPOUSER.RESOLVER.DEFAULT.DATA
Local Tcp/Ip Dataset = USER55.TCPIP.DATA
Translation Table = Default
UserlD/JobName = USER55
Caller API = TCP/IP Sockets Extended
Caller Mode = EBCDIC
(L) DatasetPrefix = USER55
(G) HostName = MvS026
(L) TcpIpJobName = TCPCS
(G) Search = tcp.raleigh.ibm.com
raleigh.ibm.com
ibm.com

Chapter 39. Diagnosing resolver problems  867
### z/OS V1R11.0 Comm Svr: IP Diagnosis Guide

<table>
<thead>
<tr>
<th>uk</th>
<th>gov</th>
</tr>
</thead>
</table>

1. **SortList**
   - 0.0.19.0/0.0.255.0
   - 0.0.18.99/0.0.255.255
   - 0.42.17.0/0.255.255.0
   - 129.42.16.0/255.255.255.0

2. **NameServer**
   - 127.0.0.1
   - 9.67.128.255

3. **NsPortAddr**
   - 53

4. **ResolveVia**
   - UDP

5. **Options**
   - N Dots = 1
   - SockNoTestStor = NO

6. **LookUp**
   - DNS LOCAL

7. **ResolverTimeout**
   - 3

8. **ResolverUdpRetries**
   - 1

9. **NDots**
   - 1

10. **MessageCase**
    - MIXED

11. **LoadDbsTable**
    - BIG5

12. **res_init**
    - Started: 2008/11/08 20:30:11.359767
    - Ended: 2008/11/08 20:30:11.359776

13. **GetAddrInfo**
    - Started: 2008/11/08 20:30:11.365043

14. **res_querydomain**
    - res_querydomain(TESTBEN46.SVT390.COM., , C_IN, T_AAAA)
    - res_query(TESTBEN46.SVT390.COM., C_IN, T_AAAA)

15. **res_mkquery**
    - res_mkquery created message:
      - Query Id: 28803
      - Flags: 00000001 00000000
      - Flags set: recurDes
      - OpCode: QUERY
      - Response Code: NOERROR

16. **res_send**
    - Name Server Capabilities
      - Name server 127.0.0.1 EDNS0 Support = Up-level
      - Name server 9.67.128.255 EDNS0 Support = Unknown
res_send received data via UDP. Message received:

* * * * * Beginning of Message * * * * *

Query Id: 28803
Flags: 10000101 10000000
Flags set: resp auth recurDes recurAvl
OpCode: QUERY
Response Code: NOERROR

Number of Question RRs: 1
Question 1:
TESTBEN46.SVT390.COM
Type (0x001C) T_AAAA Class (0x0001) C_IN

Number of Answer RRs: 20
Answer 1:
TESTBEN46.SVT390.COM
Type (0x001C) T_AAAA Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds) 2000:197:14:103::13
Answer 2:
TESTBEN46.SVT390.COM
Type (0x001C) T_AAAA Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds) 2000:197:14:103::14
Answer 3:
TESTBEN46.SVT390.COM
Type (0x001C) T_AAAA Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds) 2000:197:14:103::15
Answer 4:
TESTBEN46.SVT390.COM
Type (0x001C) T_AAAA Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds) 2000:197:14:103::16
Answer 5:
TESTBEN46.SVT390.COM
Type (0x001C) T_AAAA Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds) 2000:197:14:103::17
Answer 6:
TESTBEN46.SVT390.COM
Type (0x001C) T_AAAA Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds) 2000:197:14:103::18
Answer 7:
TESTBEN46.SVT390.COM
Type (0x001C) T_AAAA Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds) 2000:197:14:103::19
Answer 8:
TESTBEN46.SVT390.COM
Type (0x001C) T_AAAA Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds) 2000:197:14:103::20
Answer 9:
TESTBEN46.SVT390.COM
Type (0x001C) T_AAAA Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds) 2000:197:14:103::21
Answer 10:
TESTBEN46.SVT390.COM
Type (0x001C) T_AAAA Class (0x0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
2000:197:14:103::2
Answer 11:
TESTBEN46.SVT390.COM
Type (0X001C) T_AAAA Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
2000:197:14:103::3
Answer 12:
TESTBEN46.SVT390.COM
Type (0X001C) T_AAAA Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
2000:197:14:103::4
Answer 13:
TESTBEN46.SVT390.COM
Type (0X001C) T_AAAA Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
2000:197:14:103::5
Answer 14:
TESTBEN46.SVT390.COM
Type (0X001C) T_AAAA Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
2000:197:14:103::6
Answer 15:
TESTBEN46.SVT390.COM
Type (0X001C) T_AAAA Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
2000:197:14:103::7
Answer 16:
TESTBEN46.SVT390.COM
Type (0X001C) T_AAAA Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
2000:197:14:103::8
Answer 17:
TESTBEN46.SVT390.COM
Type (0X001C) T_AAAA Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
2000:197:14:103::9
Answer 18:
TESTBEN46.SVT390.COM
Type (0X001C) T_AAAA Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
2000:197:14:103::10
Answer 19:
TESTBEN46.SVT390.COM
Type (0X001C) T_AAAA Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
2000:197:14:103::11
Answer 20:
TESTBEN46.SVT390.COM
Type (0X001C) T_AAAA Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
2000:197:14:103::12

Number of Authority RRs: 1
Authority 1:
SVT390.COM
Type (0X0002) T_NS Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
sdistcici.SVT390.COM

Number of Additional RRs: 2
Additional 1:
sdistcici.SVT390.COM
Type (0X0001) T_A Class (0X0001) C_IN
TTL: 0600 (0 days, 0 hours, 10 minutes, 0 seconds)
10.11.235.51
Additional 2:
The following describes highlighted numbered areas of the example setup files and example trace resolver output.
Errors deliberately entered into this example to show action taken.

- Line 8 in the global file specifies seven SEARCH values; the maximum number allowed is 6. The seventh value is ignored.

- Line 11 in the global file specifies five SORTLIST values; the maximum number allowed is 4. The fifth value is ignored.

- Line 18 in the global file has a value for LOADBCSTABLES that is not valid. The value is ignored.

- Line 14 in the local file has a value for ALWAYSWTO that is not valid. The value is ignored and the default is used.

The ResolveVia field specifies UDP even though the local file indicated RESOLVEVIA TCP. UDP is used because GLOBALTCPDATA is being used. If a global file is used, then all resolver-related TCPIP.DATA statements must be specified in it. If the resolver statements are not specified then default values are assigned. In this example, resolver statements are not specified as shown by RESOLVEVIA in the global file being commented out.

A local file cannot override the global file for any value. The global file specifies the hostname, therefore the local file value of MVS000 does not override the global value of MVS026. Likewise, since there is a GLOBALTCPDATA specified all resolver related statements in a local file are ignored (for example, DOMAINORIGIN, NAMESERVER and RESOLVERTIMEOUT).

The list of name servers to be queried also indicates whether the resolver considers the name server to support the Extension Mechanisms for DNS (EDNS0) capability. The resolver may believe that the name server supports EDNS0 (EDNS0 Support = Up-level), that the name server does not support EDNS0 (EDNS0 Support = Down-level), or that the capability of the name server is undetermined (EDNS0 Support = Unknown).

A REXX application calls GetHostByName at the indicated local date and time. The flow through the resolver API calls shows the parameters being passed.

Trace output reports the date and time the TCPIP.DATA statements were processed.

The res_init() resolver initialization values are reported. These are the values actually being used by the resolver, with an indication of the origin of the value. The indicators are:

- * Default value
- D Default file (not used if the local file is found)
- E Environment variable
- G Global file
res_mkquery creates a DNS message (from Beginning of Message to End of Message). The message is interpreted, and flags and codes are spelled out. 

a res_mkquery will append an additional RR record (the OPT RR) to the request being built. The presence of the OPT RR record on the request indicates that the resolver supports the EDNS0 function and that UDP packets of up to 3072 bytes can be sent to the resolver (instead of the normal 512 byte limitation).

res_send sends the query to the name server. The res_send function calls several z/OS UNIX functions; the indentation of the lines following res_send indicate res_send was the caller. The IP address of the DNS that sent the response is also displayed.

res_send receives a message from DNS. This message is truncated because the amount of data that the DNS has available to send regarding the resource is greater than 512 bytes.

a res_send receives a message from DNS. This time, because the DNS query from resolver contained the OPT RR record indicating up to 3072 bytes of UDP data could be returned, the DNS returns all 553 bytes of data using UDP protocols. The total number of Answer records in this response is 29.

b res_send receives a message from DNS. This message is also greater than 512 bytes, but since the OPT RR record was included on the query, the full amount of data can be sent on the first UDP response from the name server. The total number of Answer records in this response is 20.

The GetHostByName function reports success and lists the IP addresses returned. If addresses matched any of the values in the SORTLIST definitions, the order of the addresses would have been modified to match the SORTLIST specification.

LookUp specifies the order in which the DNS and the local host file are to be used for name resolution. There are four possible search orders:

- LOOKUP DNS LOCAL (DNS search first)
- LOOKUP LOCAL DNS (local host file search first)
- LOOKUP DNS (only DNS search)
- LOOKUP LOCAL (only local host file search)

Ping calls GetAddrinfo at the indicated local date and time. The flow through the resolver API calls shows the parameters being passed.

- ai_family = 0 means that AF_UNSPEC is specified
- ai_flags = x'00000062' means that AI_CANNONNAMEOK, AI_ALL, and AI_ADDRCONFIG are specified
- ai_protocol = 0 and ai_socktype = 0 means that protocol and socktype are not specified
Refer to z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference for more information about input values of getaddrinfo.

In order to honor the setting of ai_ADDRCONFIG, the Resolver must query the stacks to determine whether IPv6 or IPv4 interfaces exist (the results of the query are shown in message S7600). A socket, separate from the one used to send DNS queries, is opened for communicating with the stacks.

The socket used for communicating with the stacks is closed prior to finishing Getaddrinfo processing.

The resolver detected that the system can handle both IPv4 and IPv6 addresses.

Because the system can handle both IPv4 and IPv6, and ai.ALL is specified, the resolver sends the IPv6 query (T_AAAA) for IPv6 to DNS first. For an explanation of how resolver decides to send an IPv6 or IPv4 query to DNS, refer to the z/OS Communications Server: IPv6 Network and Application Design Guide.

The resolver prepares to send the IPv4 query (T_A). For an explanation of how resolver decides to send an IPv6 or IPv4 query to DNS, refer to the z/OS Communications Server: IPv6 Network and Application Design Guide.

Because no Service operand was passed as input to Getaddrinfo, there is no service resolution to perform, so any sockaddr returned will have a port number=0.

Prior to returning resolved addresses to the application, the resolver sorts all addresses so that the most preferable is the first in the address chain. Refer to the z/OS Communications Server: IPv6 Network and Application Design Guide for more information.

The caller API value indicates which search order is used by the resolver for any required local table usage. The following caller API values indicate the z/OS UNIX environment search order is used:

1. Language Environment C Sockets
2. Unix System Services

The following caller API values indicate the native MVS environment search order is used:

1. TCP/IP Pascal Sockets
2. TCP/IP C Sockets
3. TCP/IP Rexx Sockets
4. TCP/IP Sockets Extended

The Caller Mode value indicates the representation of any input characters as being either in EBCDIC or ASCII.

Because the resolver’s awareness of the EDNS0 capability of name servers is maintained on a system-wide level, it is possible that the resolver has a
different awareness of the name server capability during res_send processing
than it had during res_init processing. The current awareness level is displayed
prior to sending any queries to the name servers.

Because the EDNS0 capability of the target name server is currently unknown,
the resolver will not send the OPT RR record on the request to the name
server. The DNS query built during res_mkquery processing is manipulated to
remove the additional record.

a  The receipt of the truncated UDP response causes resolver to reissue
the query, to the same DNS, but this time the OPT RR record is not
removed from the request data. This query is treated as an EDNS0
capability probe, and the response we receive from the DNS will
determine whether future queries to the DNS include the OPT RR
record or not. The transaction ID for this EDNS0 probe is different
from the first query, and the new value is included in the trace
information.

b  As just displayed on message line 21, the name server is considered to
be up-level in terms of EDNS0 support, so the resolver sends the OPT
RR record that res_mkquery built on the DNS query.

The receipt of a UDP package greater than 512 bytes, in response to an EDNS0
probe, indicates that this name server is up-level. Future communication with
this name server will always include the OPT RR record to allow the name
server to send UDP responses of greater than 512 bytes without requiring
additional EDNS0 probe requests.

Since the resolver included an OPT RR on the request to indicate that the
resolver supports EDNS0, the name server will also include an OPT RR on the
response. The DNS indicates it can accept up to 4096 bytes of data on a UDP
message, but the resolver does not make use of this information.

The setting of Cache indicates that the system is currently performing resolver
caching and the application is permitted to use the caching function. An
individual application can be prevented from using the resolver caching
function by specifying NOCACHE in the TCPIP.DATA dataset.

Because resolver caching is active and available for use by this application,
prior to sending the query to the DNS, the resolver cache is queried to see if
any information is currently available for this resource.

a  The cache has no information about the resource. The resolver must
query the DNS to obtain the A record information.

b  The cache has no AAAA record information about the resource,
although it does have A record information (from the previous
GetHostByName call). The resolver must query the DNS to obtain the
AAAA information.

c  The cache has A record information about the resource, saved from the
previous GetHostByName call for the same resource, and the
information has not expired, so the saved information is retrieved from
the cache. No communication to the name server is necessary this time
to obtain the A record information about the resource.
A timestamp is displayed when the query is actually sent to the DNS name server.

A corresponding timestamp is displayed when the response from the DNS has been successfully processed. If no response is received, the timestamp represents when the resolver stopped waiting for the response. The amount of time elapsed from the start of communication with the name server is also displayed.

Because resolver caching is active and available for use by this application, the information about this resource is cached for later re-use by this application or other applications on the system.

The resolver will return at most 35 IP addresses for a resource, even if, as in this case, there are more IP addresses that could be returned (20 IPv6 addresses, and 29 IPv4 addresses). Priority is given to IPv6 addresses, so in this case, all the possible IPv6 addresses are returned, but only 15 of the possible IPv4 addresses.

The time to live (TTL) value represents the amount of time that the resolver cache can use the returned information about the resource. The MAXTTL resolver setup statement can be used to define an upper limit on the actual TTL value used for a resource by the resolver.

Notes:
1. If any errors occurred, refer to "z/OS Communications Server: IP and SNA Codes".
2. In a multitasking environment, if the LRECL of the trace resolver output is at least 128 characters, the TCB address appears at the end of each line. The TCB address can be useful in determining the origin of the resolver request.

**CTRACE — RESOLVER**

Component Trace (CTRACE) is used for the RESOLVER component (SYSTCPRE) to collect debug information. The TRACE RESOLVER traces information on a per-application basis and directs the output to a unique file for each application. The CTRACE shows resolver actions for all applications (although it might be filtered).

The CTRACE support allows for JOBNAME, ASID filtering, or both. The trace buffer is located in the Resolver private storage. The trace buffer minimum size is 128K, maximum 128M, default 16M. Trace records can optionally be written to an external writer.

The Resolver CTRACE initialization PARMLIB member can be specified at Resolver start time. Using the sample Resolver procedure shipped with the product, enter the following console command:

```
S RESOLVER,PARMS='CTRACE(CTIRESxx)'
```

where xx is the suffix of the CTIRESxx PARMLIB member to be used. To customize the parameters used to initialize the trace, you can update the SYS1.PARMLIB member CTIRES00.
Note: In addition to specifying the trace options, you can also change the Resolver trace buffer size. The buffer size can be changed only at Resolver initialization.

If the CTIRES00 member is not found when starting the Resolver, the following message is issued:

IEEE538I CTIRES00 MEMBER NOT FOUND in SYS1.PARMLIB

When this occurs, the Resolver component trace is started with a buffer size of 16MB and the MINIMUM tracing option.

After Resolver initialization, you must use the TRACE CT command to change the component trace options (see Chapter 5, “TCP/IP services traces and IPCS support,” on page 47). Each time a new component trace is initialized, all prior trace options are turned off and the new options are put into effect.

Trace options:

ALL
All options.

MINIMUM
The minimum set of options traces exceptions, Resolver initialization and termination, Resolver CTRACE changes, and Resolver operator messages.

Following is the sample PARMLIB member.

```*/
*/   */
*/   */
*/   */
*/   Copyright: */
*/   */
*/   Licensed Materials - Property of IBM */
*/   */
*/   5694-A01 */
*/   */
*/   (C) Copyright IBM Corp. 2001, 2003 */
*/
*/
*/
*/   Status: CSV1R5 */
*/
*/
*/
*/   DESCRIPTION = This parmlib member causes component trace for */
*/   the TCP/IP provided Resolver to be initialized with a trace buffer size of 16M */
*/
*/   This parmlib member only lists those TRACEOPTS values specific to the TCP/IP Resolver. For a */
*/   complete list of TRACEOPTS keywords and their values see: */
*/   z/OS MVS INITIALIZATION AND TUNING REFERENCE. */
*/
*/
*/   $PARMS(CTIRES00),COMP(RES ),PROD(TCPIP ): Resolver Component Trace*/
*/   SYS1.PARMLIB member */
*/
/**/```
When formatting the Resolver trace, use the CTRACE command. See Chapter 5, "TCP/IP services traces and IPCS support," on page 47 for the syntax for formatting a CTRACE. For the Resolver, the following formatting OPTIONS are available:

**ASCII**
Resolver trace data is displayed with ASCII translation only. The default is EBCDIC.

**BOTH**
Resolver trace data is displayed with both EBCDIC and ASCII translations. Each line of formatted data contains the offset, the hexadecimal display, the EBCDIC translation, then the ASCII translation. The default is EBCDIC.

**EBCDIC**
Resolver trace data is displayed with EBCDIC translation only. This is the default.

**HEX**
Resolver trace data is displayed only in hexadecimal (no ASCII or EBCDIC translation). The default is EBCDIC.

**Guideline:** If the formatted CTRACE display wraps on the screen, use the IPCS PROFILE LINESIZE(nnn) command, where nnn is the largest number of characters that displays on one line.
Chapter 40. Diagnosing Simple Network Time Protocol (SNTP) problems

Simple Network Time Protocol (SNTP) is a standard protocol used to synchronize system clocks on routers and computer systems throughout the Internet through a specific formatted message. The Simple Network Time Protocol Daemon (SNTPD) is a TCP/IP daemon that is used to synchronize time between a client and a server.

This topic describes how to diagnose problems with SNTP daemon and contains the following sections:
- "Activating the SNTPD debug trace"
- "Abends"
- "Steps for stopping SNTPD"
- "Sample SNTPD debug output" on page 880

Activating the SNTPD debug trace

To activate the SNTPD debug trace, specify the -d or -df parameter when starting SNTPD via the z/OS UNIX shell or as an MVS started procedure.

<table>
<thead>
<tr>
<th>If this option is used...</th>
<th>Then (phrase) . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>-d parameter</td>
<td>Messages are written to the syslog daemon.</td>
</tr>
<tr>
<td>-df parameter</td>
<td>Messages are written to the file specified on the -df parameter.</td>
</tr>
</tbody>
</table>

Restriction: You must specify a path name and file name.

Abends

An abend during SNTPD processing should result in messages and error related information being sent to the system console. A dump of the error is needed unless the symptoms already match a known problem.

Steps for stopping SNTPD

If SNTPD was started from the z/OS UNIX shell, the kill command must be used to stop SNTPD.

Before you issue the kill command: You must determine the PID (process ID) of SNTPD.

Perform the following steps to stop the process ID of SNTPD.
1. To find the PID, use one of the following methods:
   • Use D OMVS,U=userid. (This is the USERID that started SNTPD from the shell.)
   • Use the ps -ef command from the shell.
   • Write down the PID when you start SNTPD.
2. From a z/OS UNIX shell superuser ID, issue the **kill** command to the process ID (PID) associated with SNTPD.

You know you are finished when the following message appears: EZZ9601I SNTP SERVER ENDED. If SNTPD was started as an MVS started procedure, you must use the **stop** command to stop SNTPD. For example, code:

```
p snntp
```

**Sample SNTPD debug output**

Refer to [z/OS Problem Management](https://www.ibm.com/support/docview.wss?uid=swg21181795) or see Chapter 3, “Diagnosing abends, loops, and hangs,” on page 25, for information about debugging dumps produced during SNTP processing.

The following shows a sample of SNTP debug output.

```
Tue Apr 2 15:26:14 2002 SNTP enabled options: Opening debugging file /tmp/bc6.log
(Multicast: every 120 seconds) (PID FILE: /etc/sntpd.pid) (DEBUG FILE: /tmp/bc6.log
Tue Apr 2 15:26:14 2002 Writing PID to file /etc/sntpd.pid
Tue Apr 2 15:26:14 2002 EZZ9602I SNTP server initializing
Tue Apr 2 15:26:14 2002 Initializing signal handling
Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGINT
Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGTERM
Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGABND
Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGABRT
Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGQUIT
Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGHUP
Tue Apr 2 15:26:14 2002 Set sigaction of signal SIGTTOU
Tue Apr 2 15:26:14 2002 Initializing signal handling
Tue Apr 2 15:26:14 2002 Initializing MVS command handling
Tue Apr 2 15:26:14 2002 Initializing pthread for MVS command
Tue Apr 2 15:26:14 2002 Initializing UDP socket(s)
Tue Apr 2 15:26:15 2002 SNTP port was set to 123
Tue Apr 2 15:26:15 2002 Bound to address: 9.67.2.1
Tue Apr 2 15:26:15 2002 Bound to address: 9.67.115.15
Tue Apr 2 15:26:15 2002 Bound to address: 0.0.0.0
Tue Apr 2 15:26:15 2002 Initializing pthread for multicast/broadcast
Tue Apr 2 15:26:15 2002 Initializing pthread for unicast
Tue Apr 2 15:26:15 2002 EZZ96001 SNTP server ready
Tue Apr 2 15:26:15 2002 Sending NTP message to multicast address 224.0.1.1
Tue Apr 2 15:30:15 2002 Sending NTP message to multicast address 224.0.1.1
```
Chapter 41. Diagnosing Communications Server SMTP application problems

The Communications Server SMTP (CSSMTP) application transfers electronic mail from JES spool datasets to SMTP mail relays for delivery to the final destination. See z/OS Communications Server: IP Configuration Guide for overview and setup of CSSMTP.

The following information describes how to diagnose problems with the CSSMTP application.

- "Gathering diagnostic information"
- "Resolving initialization or logging problems" on page 882
- "Resolving SMTPNOTE CLIST problems" on page 883
- "Diagnosing and resolving Resolver problems" on page 883
- "Resolving problems from the JES spool data set" on page 885
- "Resolving mail problems" on page 888
- "Resolving MODIFY command problems" on page 894
- "Diagnosing checkpoint problems" on page 895
- "Monitoring resources used" on page 896

Gathering diagnostic information

You might need to collect multiple pieces of data in order to accurately diagnose problems.

- Capture the MVS console messages.
- If the problem is caused by configuration issues or target server definitions, then save the configuration file.
- Capture the CSSMTP application log messages. See "Steps for gathering log information" for details.
- If you get an abend during the CSSMTP application processing, messages and error-related information should be sent to the system console. A dump of the error is needed unless the symptoms already match a known problem. If an abend occurs, then save the resulting address space dump. See Chapter 3, "Diagnosing abends, loops, and hangs," on page 25 for details.

Steps for gathering log information

1. Determine where log data is located. If you are logging to a log file, then examine the LOGFILE DD statement in your started procedure to determine the log file or data set and save it. If you have not specified a LOGFILE DD, then the log will go to SYSLOGD.
2. If you are running multiple CSSMTP applications on the same system then you should be logging to SYSLOGD. Look in the SYSLOGD configuration file to determine where log records for CSSMTP are being written.
3. You should at least be executing with the default log level of error, warning, and event. If more logging information is needed, you can specify a loglevel value greater than the default. See the LogLevel statement in z/OS Communications Server: IP Configuration Reference for valid CSSMTP / LogLevel statement information. The log level can also be changed with the MODIFY
LOGLEVEL command, see the MODIFY command: Communications Server
SMTP section in z/OS Communications Server: IP System Administrator's
Commands for details.

**Resolving initialization or logging problems**

The following configuration messages indicate that initialization is done and you have successfully connected to at least one target for receiving mail.

- EZD1802I csproc INITIALIZATION COMPLETE FOR extWrtName
- EZD1821I csproc ABLE TO USE TARGET SERVER ipAddress

**Table 93. Common initialization and logging problems**

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZD1807I</td>
<td>This message indicates that initialization failed.</td>
<td>For details and other messages, see z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</td>
</tr>
<tr>
<td>EZD1824E</td>
<td>This message indicates that a TCP/IP stack is not available.</td>
<td>For details and other messages, see z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</td>
</tr>
<tr>
<td>EZD1815E</td>
<td>This message indicates that there was resolver problem at initialization.</td>
<td>For details and other messages, see z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</td>
</tr>
<tr>
<td>EZD1811I</td>
<td>This message indicates a problem in writing to the log file.</td>
<td>For details and other messages, see z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</td>
</tr>
<tr>
<td>EZD1835I</td>
<td>This message indicates that you did not configure a CHKPOINT DD statement.</td>
<td>For details and other messages, see z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</td>
</tr>
<tr>
<td>EZD1836I</td>
<td>This message indicates that you configured a CHKPOINT DD statement but checkpointing has failed.</td>
<td>For details and other messages, see z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</td>
</tr>
</tbody>
</table>
Resolving SMTPNOTE CLIST problems

Table 94. Problems using SMTPNOTE CLIST to create mail messages on the JES spool data set

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZA5579E</td>
<td>This message indicates that the mail messages generated from the SMTPNOTE CLIST is unable to be transmitted. Ensure that the SMTPNOTE CLIST is customized. See z/OS Communications Server: IP Configuration Guide for details and see the appropriate EZA5579E message in z/OS Communications Server: IP Messages Volume 1 (EZA).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The mail message was queued to the JES spool queue, but it has not been processed by the CSSMTP application. • The specified smtpjob in the SMTPNOTE CLIST may have the wrong name. • The specified smtpjob in the SMTPNOTE CLIST may not be active.</td>
<td>• Ensure that the SMTPNOTE CLIST is customized. See z/OS Communications Server: IP Configuration Guide for details. • Start the smtpjob application.</td>
</tr>
</tbody>
</table>

Diagnosing and resolving Resolver problems

Table 95. Diagnosing and resolving Resolver problems

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>No IP addresses were resolved from the TargetServer configuration statements. EZD1815E (during initialization) is issued or EZD1845I (MODIFY REFRESH or MODIFY REFRESHIPLIST command) is issued. See the appropriate EZD1815E, EZD1845I messages in z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</td>
<td>Possible causes for this error are: • The Resolver is not initialized. • There may be a problem in the DNS or the Resolver setup. • The name specified on the TargetServer statement (TargetName or TargetMx parameter) may be incorrect. • If the resolver is not initialized, start the resolver. See the appropriate EZD1815E, EZD1845I messages in z/OS Communications Server: IP Messages Volume 2 (EZB, EZD) • If the name on the TargetServer statement is incorrect, update the name and issue the MODIFY REFRESH command to reprocess the configuration file. • If the name on the TargetServer statement is correct, investigate your DNS and/or resolver setup.</td>
<td></td>
</tr>
</tbody>
</table>
Table 95. Diagnosing and resolving Resolver problems (continued)

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
</table>
| Updated IP addresses were resolved from the TargetServer configuration statements with warnings. EZD1847I is issued. See the appropriate EZD1847I message in [z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)]. | Possible causes for this warning are:  
- A TargetName or TargetMx name could not resolve to any IP address  
- A TargetName or TargetMx name resolved to more than 4 IP addresses | Verify that the name specified on the TargetName or TargetMx parameter is correct. If the name is in error, update the name and issue the MODIFY REFRESH command to reprocess the configuration file. If the name is correct, investigate your DNS and/or resolver setup.  
If more than four addresses are resolved for a name specified on a TargetServer statement, only the first four are allowed and the rest ignored.  
**Note:** A warning is informational and no action is required. |
| No changed IP addresses were resolved from the TargetServer configuration statements as a result of the MODIFY REFRESH command or MODIFY REFRESHIPLIST command. EZD1843I or EZD1844I is issued. See the appropriate messages in [z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)]. | If changed IP addresses were expected, then it might indicate a problem.  
- The name specified on the TargetServer statement may not be updated with the new name.  
- The name specified on the TargetServer statement was changed, but the updates in the DNS or the Resolver setup may not be refreshed. | Verify that the name specified on the TargetName or TargetMx parameter is correct. If the name is in error, update the name and issue the MODIFY REFRESH command to reprocess the configuration file.  
Refresh the resolver so that the response is not coming from the cache, then issue the MODIFY REFRESHIPLIST command to refresh the TargetName or TargetMx. |
| MODIFY REFRESH or REFRESHIPLIST unsuccessful due to a MODIFY COMMAND in progress. EZD1806I is issued. See the appropriate EZD1806I message in [z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)]. | DNS setup may be in error causing the MODIFY COMMAND to take a long time. | Check if DNS is active at the IP address you specified for the NameServer or NSINTERADDR in the TCPIP.DATASET.  
Check the port that DNS is listening on. The port is specified on the NSPORTADDR in the TCPIP.DATASET. |

The Resolver trace shows requests and responses sent to and received from name servers. It also shows if local hosts tables are used for name resolution. This trace helps you diagnose problems with host name resolution. Please see "TRACE RESOLVER" on page 857 for more information about how to activate trace resolver output.
### Resolving problems from the JES spool data set

**Table 96. JES spool data set problems**

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZD1807I with JES NOT AVAILABLE</td>
<td>The CSSMTP application ends with the error during initialization because JES is not available. This is a permanent JES subsystem error during spool file initialization (for example, authentication errors, The CSSMTP application is not allowed to read it, IEFSSREQ failed).</td>
<td>For details and other messages, see the appropriate EZD1807I message in <a href="https://www.ibm.com">z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</a>.</td>
</tr>
</tbody>
</table>
| EZD1813I or EZD1816I | No mail messages in the spool file have been processed, because the entire JES spool file is permanently inaccessible. Possible causes for the error are:  
  - There is a spool file allocation error  
  - There is a spool file open error  
  - This user ID is not permitted to send output to the external writer that is configured to the CSSMTP application.  
  - The spool file size is larger than the JesJobSize. | For details and other messages, see the appropriate EZD1813I or EZD1816I message in [z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)](https://www.ibm.com). For details on MailAdministrator and JesJobSize configuration options, see [z/OS Communications Server: IP Configuration Reference](https://www.ibm.com). For details on JES issues relating to CSSMTP see [z/OS Communications Server: IP Configuration Guide](https://www.ibm.com). |
<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
</table>
| EZD1813I or EZD1816I | Some of the mail messages in the JES spool file have been processed and sent, but the rest of the JES spool file is permanently inaccessible. Possible causes for the error are:  
- IO error during read  
- Permanent SAPI errors during read  
- Syntax errors from JES spool processing, for example:  
  - No valid HELO or EHLO SMTP command  
  - There is an incorrect beginning SMTP command for HELO, EHLO, MAIL FROM  
  - There is an incorrect ending SMTP command ".", RSET or next HELO or EHLO  
  - There are more than five invalid SMTP commands, or sequence errors  
| EZD1813I or EZD1816I | Some of the individual mail message in a JES spool file are permanently non-deliverable. Possible causes for errors are:  
- There are non-mail boundary errors, for example:  
  - Missing RCPT TO  
  - JmsgSize error  
The following is an error report example to MailAdministrator or to the sysout file.

1. This identifies the report with the jobname and jobid that contained the JES spool file.

2. These further identify the source of the JES spool file with the jobname, the procedure step name, the job step name, the JES jobid, the originator node user ID, and the local time zone.

3. The CSSMTP application jobname and external writer name

4. The line and mail found in the JES spool file cannot be delivered to listed recipients below. The Message-ID is from the header line which identifies this mail to the other SMTP servers. The Error: lines describe the reason the CSSMTP application could not send the mail.

5. The line and mail found in the JES spool file where a JES syntax error was found. The next line identifies the cause of the error. In this case a blank in the mailbox.

---

**Notes:**

1. This identifies the report with the jobname and jobid that contained the JES spool file.

2. These further identify the source of the JES spool file with the jobname, the procedure step name, the job step name, the JES jobid, the originator node user ID, and the local time zone.

3. The CSSMTP application jobname and external writer name.

4. The line and mail found in the JES spool file cannot be delivered to listed recipients below. The Message-ID is from the header line which identifies this mail to the other SMTP servers. The Error: lines describe the reason the CSSMTP application could not send the mail.

5. The line and mail found in the JES spool file where a JES syntax error was found. The next line identifies the cause of the error. In this case a blank in the mailbox.
6. The line and mail found in the JES spool file cannot be delivered to listed recipients below. In this case the target server rejected the mail because a mailbox could not be accepted.
7. Notice that the mail is listed in the order that they were completed and not in the order that they appear in the spool file.
8. The processing of the spool file was completed at this date and time.
9. The number of mail messages found in the spool file.
10. The number of mail messages that contained parsing errors.
11. The number of recipients to whom mail sent successfully.
12. The number of recipients to whom mail could not be delivered.
13. The disposition of the JES spool file that was processed. If the value is HOLD or DELETE it reflects the setting of the BadSpoolDisp configuration statement. If the value is KEEP, it indicates that the CSSMTP application was interrupted during processing of this file. The file will continue to be processed when the application is restarted.

## Resolving mail problems

To resolve common problems with connecting to or communicating with a target server at a certain IP address, begin by reviewing messages sent to the system console. Review the CSSMTP log file for more detail about the specific problem. [Table 97](#) shows the possible messages and their meaning.

### Target server problems

**Table 97. Common messages indicating target server status**

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZD1817I</td>
<td>This message indicates that the application has been unable to successfully connect to the target server at the indicated IP address due to a socket connect() failure or timeout.</td>
<td>For details and other messages, see the appropriate EZD1817I message in <a href="#">z/OS Communications Server: IP Messages Volume 2 (EZB)</a>.</td>
</tr>
</tbody>
</table>
| EZD1818I  | This message indicates that the application has been unable to communicate successfully with the target server at the indicated IP address due to a SMTP protocol problem or timeout after a successful connect. SMTP protocol problems can be the following:  
  - Initial SMTP greeting not sent  
  - 4xx or 5xx reply codes sent on EHLO SMTP command  
  - 4xx or 5xx reply codes sent on HELO SMTP command
| Other problems are socket read() or write() failures on the connection. | For details and other messages, see the appropriate EZD1818I message in [z/OS Communications Server: IP Messages Volume 2 (EZB)](#). If the problem appears to be in communicating with a target server, setting the loglevel to at least 39 will help to capture TCP/IP traces that show SMTP commands and remote SMTP server replies between the CSSMTP application and the TCP/IP network. |
Table 97. Common messages indicating target server status (continued)

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
</table>
| EZD1819I   | This message indicates that the application has been unable to establish a TLS connection with the target server at the indicated IP address due to a SMTP protocol problem, a security certificate problem or a timeout. SMTP protocol problems can have the following causes:  
- STARTTLS option not supported on EHLO SMTP command  
- 4xx or 5xx reply codes sent on STARTTLS SMTP command  
- The stack does not support TLS policies.  
- The policy agent is not started resulting in SIOCTTLSCTL ioctl() failures.  
- Socket read() or write() failures on the connection.                                                                                     | For details and other messages, see the appropriate EZD1819I message in z/OS Communications Server: IP Messages Volume 2 (EZB EZD). For details setting up TLS security on the CSSMTP application, see z/OS Communications Server: IP Configuration Guide. If TLS is enabled and the definitions are configured on the client stacks but EZD1819I and log file ERROR messages indicate this is due to security refer to Chapter 29, “Diagnosing Application Transparent Transport Layer Security (AT-TLS),” on page 701. If TLS is set up correctly on CSSMTP, then verify that TLS is set up correctly on the server. |
| EZD1820E   | This message indicates that no target server is capable of receiving mail. There may be a networking problem affecting all target servers. Processing of mail is suspended when this message is generated.                                                                                                           | For details and other messages, see the appropriate EZD1820E message in z/OS Communications Server: IP Messages Volume 2 (EZB EZD).                                                                                                                                                  |
| EZD1824E   | This message indicates that the application has been unable to successfully communicate to the TCP/IP stack as the result of a socket() failure. Processing of mail is suspended when this message is generated.                                                                                                           | For details and other messages, see the appropriate EZD1824E message in z/OS Communications Server: IP Messages Volume 2 (EZB EZD).                                                                                                                                                  |

Mail problems

Generally, common problems dealing with individual mail messages are handled by creating an undeliverable mail notification when the mail message becomes undeliverable.

- If ReturnToMailFrom is set to YES on the UNDELIVERABLE statement or the default value is used, an undeliverable mail notification is created that contains the original mail text as well as additional information that indicates the reason
why the mail could not be delivered. The undeliverable mail notification is sent
to the originator’s mail address as specified on the MAIL FROM command.

• If requested, a report is generated that contains the error text indicating why the
mail message could not be delivered. The report is handled according to the
setting on the REPORT configuration statement. See z/OS Communications Server:
IP Configuration Reference for details.

The following steps help in debugging mail problems.

1. Start by reviewing the error text captured in the undeliverable mail notification
for the undeliverable mail message. The following list contains reasons why
mail can become undeliverable.

   • Common problems with mail messages can be that there are no target
     servers with the correct capabilities to send the mail, such as the following:
     – The mail requires a TLS connection. This requires the target server to be
       an ESMTP which supports the EHLO SMTP command and the extension
       option STARTTLS. The target server must reply positively to the
       STARTTLS command (2xx reply code). Also the CSSMTP application must
       be able to successfully issue the socket ioctl() to communicate with the
       TCP/IP stack to establish the TLS connection.
     – The mail message size is larger than what the target server indicates it can
       support on the EHLO reply response when it is an ESMTP server.
     – The mail message size is larger than what is configured on the parameter
       MessageSize associated with the TargetServer statement when it is a SMTP
       server.

   • Target server replies to SMTP commands with a 4xx reply code, which
     indicates that the mail message send be retried. However, the number of
     retries exceeded the configured maximum, see the statement RetryLimit in
     z/OS Communications Server: IP Configuration Reference for more information.

   • Target server connection timeouts occur, which indicates that the mail
     message send be retried. However, the number of retries exceeded the
     configured maximum, see the statement RetryLimit in z/OS Communications
     Server: IP Configuration Reference for more information.

   • Target server replies to MAIL FROM, RCPT TO, DATA and sending end of
     mail (EOM) sequence with a 5xx reply code indicating failure.

The following example shows general error text for a single recipient in the
undeliverable mail notification.
Receive line for the undeliverable mail notification.

Date and time the undeliverable mail notification was created.

The email address of the CSSMTP application that created the undeliverable mail notification.

The email address to whom the undeliverable mail notification is sent.

The subject line showing the MSG-ID SMTP2.JOB00051.host2@ibm.com.Nov192008.134147.442406.1 for the original piece of mail. This is the same value as the Message-ID header in the original mail text.

General error text applies to all recipients that mail was not delivered to that do not have a specific error reply text. In this example, the general error text applies to the recipients in line 8 and 11.

Mail was not delivered to the following recipients:

- user2@samehost.ibm.com
- user3@samehost.ibm.com
- Reply text:550 Recipient unknown
- user4@samehost.ibm.com
- user5@badhost.ibm.com
- Reply text:550 "badhost" not found

Original mail text:

```
Received: from host1.ibm.com (host1)
by host2.ibm.com (host2 [x.x.x.x])
for <USER1@host1>
with ESMTP (IBM CSSMTP z/OS V01R11.00)
Id SMTP2.JOB00051.host2@ibm.com.Nov192008.134147.442406.1U ;
Wed, 19 Nov 2008 13:54:31 -0500

Date: Wed, 19 Nov 2008 13:54:31 -0500
From: <CSSMTP@host2.ibm.com>
To: <user1@ibm.com>
Subject: Undeliverable Mail for MSG-ID SMTP2.JOB00051.host2@ibm.com.Nov192008.134147.442406.1
```

This is only a test.
After this line, the list of recipients that did NOT receive the original mail text.

Original mail text was not delivered to this recipient <user3@samehost.ibm.com> because of the specific error described in line reply text. This recipient was unknown to the target server.

Original mail text was not delivered to this recipient <user5@badhost.ibm.com> because of the specific error described in line reply text. The email address of the recipient was not acceptable to the target server.

The original mail message - headers and content.

Using the undeliverable mail notification for problem determination

- If the error text captured in the undeliverable mail notification for the undeliverable mail is not enough to explain the problem, then review the log file generated under the started task using the message-ID value found in the Subject: header and search through the log file to see how the mail message was processed.
- If the problem appears in communicating with a target server, set the loglevel to at least 39 to capture TCP/IP traces. This will show SMTP commands and remote SMTP server replies between the CSSMTP application and the TCP/IP network. Resend the mail message.
- If the mail does not get sent and no undeliverable mail notification is returned to the originator of the mail, wait for CSSMTP to retry to send the mail message based on the configured RetryLimit statement values.

If there is still no undeliverable mail notification do the following:
- Check the configuration file for the undeliverable statement DeadLetterAction. If the statement is set to DELETE, then the undeliverable mail notification was deleted. If DeadLetterAction is set to STORE, then check the z/OS UNIX file system under the configured or default directory path.
- Check the log file that was generated under the started task, using the undeliverable mail notification message ID value to determine how the mail was processed. The following is the message ID for the undeliverable mail notification:

SMTP2.JOB00051.host2@ibm.com.Nov192008.134147.442406.1U
<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>No target server capable of receiving mail: original message size too large.</td>
<td>The mail size is too large. Check size of the mail including headers. Use the MODIFY DISPLAY TARGETS command, see <a href="https://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.zos.zosip.doc/commands.html">z/OS Communications Server IP System Administrator Commands</a> to know what mail size is supported by target servers and whether the target server is SMTP or ESMTP.</td>
<td>If appropriate reduce mail size and resend. Otherwise, increase the mail size supported by the target server. To check target server configuration, see TargetServer statement in <a href="https://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.zos.zosip.doc/ipsysadm/configuration/commands/targetserver.html">z/OS Communications Server IP Configuration Reference</a> if SMTP. Correct the target server on remote host if it is ESMTP.</td>
</tr>
<tr>
<td>No target server capable of receiving mail: retry count or interval is zero for this mail.</td>
<td>Review the log file. The target server is using reply code 4xx or is timing out the connection which requires a retry to occur.</td>
<td>Correct the target server or add a new target to the configuration. Change statement RetryLimit to allow retries to occur.</td>
</tr>
<tr>
<td>No target server capable of receiving mail: retry limit count exceeded for this mail.</td>
<td>Review the log file. The target server is using reply code 4xx or is timing out the connection which requires a retry to occur.</td>
<td>Correct the target server or add a new target to the configuration.</td>
</tr>
<tr>
<td>No target server capable of receiving mail: TLS support not available.</td>
<td>The batch job containing this mail required a TLS connection. The STARTTLS command is part of the batch job. Use the MODIFY DISPLAY TARGETS command. See <a href="https://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.zos.zosip.doc/commands.html">z/OS Communications Server IP System Administrator Commands</a> to determine if TLS is supported by the target servers. Target servers must be ESMTP to support TLS.</td>
<td>If appropriate remove STARTTLS command from batch job. Check the target server configuration. Review the log file for the reply code from the target server or connection timeout which requires a retry to occur. Correct the target server or add a new target server capable of doing TLS to configuration.</td>
</tr>
</tbody>
</table>
### Table 99. Dead letters problems

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZD1826I</td>
<td>This message indicates that the dead letters cannot be written to the configured or default dead letter directory.</td>
<td>For details and other messages, see the appropriate EZD1826I message in [z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)]. No more dead letters are stored. The CSSMTP application internally sets the DeadLetterAction statement as it was set to Delete. The user can automate on this message to monitor storage utilization, and decide whether MODIFY SUSPEND command should be issued to suspend all new spool file processing and should examine and clean up all stored and unneeded dead letters to free up storage. Then MODIFY REFRESH command should be issued with DeadLetterAction set to STORE to tell the CSSMTP application to write dead letters into the configured or default dead letter directory again. MODIFY RESUME command is issued (if MODIFY SUSPEND command was issued previously) to resume new spool file processing.</td>
</tr>
</tbody>
</table>

### Resolving MODIFY command problems

### Table 100. Common MODIFY command

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZD1806I</td>
<td>This message indicates the modify command was unsuccessful.</td>
<td>For details and other messages, see the appropriate EZD1806I message in [z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)].</td>
</tr>
<tr>
<td>EZD1839I</td>
<td>This message indicates MODIFY REFRESH completed with errors.</td>
<td>For details and other messages, see the appropriate EZD1839I message in [z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)].</td>
</tr>
</tbody>
</table>
Table 100. Common MODIFY command (continued)

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZD1845I</td>
<td>This message indicates that there was a resolver problem with the MODIFY REFRESH or MODIFY REFRESHIPLIST commands.</td>
<td>For details and other messages, see the appropriate EZD1845I message in z/OS Communications Server: IP Messages Volume 2 (EZB EZD).</td>
</tr>
</tbody>
</table>

Diagnosing checkpoint problems

Since each message can have several specific problems, reviewing the trace logs captured under the started task will provide more details on common problems dealing with the checkpoint functions. The following table shows the possible messages and their meaning.

Table 101. Common messages indicating checkpoint status

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZD1835I</td>
<td>This message indicates that the CHKPOINT DD statement was missing or the data set name was ‘NULLFILE’. The function is disabled.</td>
<td>For details and other messages, see the appropriate EZD1835I message in z/OS Communications Server: IP Messages Volume 2 (EZB EZD).</td>
</tr>
<tr>
<td>EZD1836I</td>
<td>This message indicates that the checkpoint data set could not be opened.</td>
<td>For details and other messages, see the appropriate EZD1836I message in z/OS Communications Server: IP Messages Volume 2 (EZB EZD).</td>
</tr>
<tr>
<td>EZD1849I</td>
<td>This message indicates that JES spool file could not be allocated by the SAPI interface. The spool file may have been previously purged by an operator command.</td>
<td>For details and other messages, see the appropriate EZD1849I message in z/OS Communications Server: IP Messages Volume 2 (EZB EZD).</td>
</tr>
<tr>
<td>Abend S08B</td>
<td>This abend is issued by the MVS Data-In-Virtual component when accessing the checkpoint data set. Reason code xxxx001D. The checkpoint data set is not a valid VSAM linear data set or an incorrect control interval size was used when the VSAM data set was defined.</td>
<td>For details about the S08B abend, see z/OS MVS System Codes.</td>
</tr>
</tbody>
</table>
**Monitoring resources used**

After the CSSMTP application is started it will periodically check the following resources, and determines if any action needs to be taken:

- JES tasks
- Storage utilization in the CSSMTP application address space
- Storage utilization for z/OS UNIX file system in the configured or default dead letter directory

Table 102. Common messages used for monitoring resources

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZD1856I</td>
<td>This message indicates either that over 75% of DEST JES tasks and 75% of WRITER JES tasks are waiting for long retry processing to complete one or more mail messages in a spool file.</td>
<td>For details and other messages, see the appropriate EZD1856I message in z/OS Communications Server: IP Messages Volume 2 (EZB). The user should automate on this message to monitor the JES tasks that are not available, and decide whether the MODIFY FLUSH Retry command should be issued to remove mail messages from the retry queue to free up the JES tasks, or to issue the MODIFY SUSPEND command to suspend all new spool file processing. Message EZD1857I is issued when the usage of DEST JES tasks and the usage of WRITER JES tasks which are waiting for long retry processing drops below 50%. The user can automate this message to monitor the usage of JES tasks, and to decide whether the MODIFY RESUME command (if MODIFY SUSPEND command was issued previously) should be issued to resume the processing for the new spool files.</td>
</tr>
<tr>
<td>Symptoms</td>
<td>Problem/Cause</td>
<td>Action</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EZD1858I</td>
<td>This message indicates that storage utilization in the CSSMTP application address space exceeds 75%.</td>
<td>For details and other messages, see the appropriate EZD1858I message in <a href="https://www.ibm.com">z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)</a>. The user should automate on this message to monitor storage utilization, and decide whether the MODIFY SUSPEND command should be issued to suspend all new, spool file processing. EZD1859I is issued when storage utilization drops below 50% in the CSSMTP application address space. The user can automate this message to monitor the storage utilization, and decide whether the MODIFY RESUME command (if MODIFY SUSPEND command was issued previously) should be issued to resume the processing for the new spool files.</td>
</tr>
</tbody>
</table>


Table 102. Common messages used for monitoring resources (continued)

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Problem/Cause</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZD1860I</td>
<td>This message indicates that storage utilization for z/OS UNIX file system in the configured or default dead letter directory exceeds 75%.</td>
<td>For details and other messages, see the appropriate EZD1860I message in z/OS Communications Server: IP Messages Volume 2 (EZB, EZD) The user should automate on this message to monitor storage utilization, and decide whether the MODIFY SUSPEND command should be issued to suspend all new spool file processing and clean up all stored and unneeded dead letters to free up storage. EZD1861I is issued when storage utilization for the z/OS UNIX file system in the configured or default dead letter directory drops below 50%. The user can automate this message to monitor the storage utilization, and decide whether the MODIFY RESUME command (if MODIFY SUSPEND command was issued previously) should be issued to resume the processing for the new spool files.</td>
</tr>
</tbody>
</table>
Chapter 42. Diagnosing storage abends and storage growth

The key to the successful resolution of most storage problems is to first determine whether the storage problem you are experiencing is related to common, private or communication storage manager (CSM) storage. This topic outlines steps you can use to determine the type of storage you are having a problem with and the steps to take to diagnose the storage problem.

This topic contains the following sections:

- Storage definitions
- Monitoring storage utilization
- Limiting TCP/IP common and private storage utilization
- Limiting CSM storage utilization
- Storage messages
- Abends
- Problem determination
- Collecting documentation

Storage definitions

TCP/IP uses several types of storage.

**common storage**

Common storage is shared across the whole system and can be accessed from any address space. Common storage is managed by the Virtual Storage Management component of the z/OS operating system. TCP/IP's usage of common storage for the most part is for ECSA (extended common service area).

**private storage**

Private storage is storage that is unique to an address space. Private storage is also referred to as pool storage.

**CSM**

Communication Storage Manager (CSM) enables TCP/IP, VTAM, and other applications to use CSM buffers to reduce data movement. CSM interfaces with the z/OS operating system to provide the storage buffers. Buffers are maintained in both common storage and in the CSM data space. The application (for example TCP/IP) has the option of requesting the buffers from common storage or the CSM dataspace. The storage is managed in pools of the following predefined buffer sizes:

- 4KB
- 16KB
- 32KB
- 60KB
- 180KB

Refer to [z/OS Communications Server: CSM Guide](#) for additional information relating to the CSM component.
Monitoring storage utilization

Storage is a resource that many users monitor very closely to determine their average utilization. You can monitor storage by using storage monitors; by manually issuing TCPIP and CSM display commands; or by using Netview CLISTs that are triggered at specific time intervals. Automatically issuing commands to your system log at periodic intervals is the most efficient way to monitor your storage utilization.

Use this log output to establish a storage utilization history. Knowing how much common, private, or CSM storage you typically use can be helpful when trying to resolve problems where TCP/IP storage utilization is increasing (also referred to as a storage creep) or you receive an abend related to an out of storage condition.

Monitor TCP/IP common and private storage utilization by issuing the Display TCPIP,,STOR console command:

For example:

\verbatim
    d tcpip,,stor
    TCPIP STORAGE
    TCPCS STORAGE CURRENT MAXIMUM LIMIT
    TCPCS ECSA 14M 20M 120M
    TCPCS POOL 52M 62M NOLIMIT
    DISPLAY TCPIP STOR COMPLETED SUCCESSFULLY
\endverbatim

\verbatim
    or
    d tcpip,tcpip2,stor
    TCPIP STORAGE
    TCPIP2 STORAGE CURRENT MAXIMUM LIMIT
    TCPIP2 ECSA 45654K 56823K 204800K
    TCPIP2 POOL 124634K 143743K 524288K
    DISPLAY TCPIP STOR COMPLETED SUCCESSFULLY
\endverbatim

For additional information regarding the Display TCPIP,,STOR command, refer to \textit{z/OS Communications Server: IP System Administrator's Commands}.

TCP/IP's CSM storage utilization is not included in the TCP/IP display. TCP/IP's CSM storage utilization can be monitored by issuing the following console commands.

To display information about storage managed and used by CSM for all owners, issue the following command:

\verbatim
    d net,csm,ownerid=all
\endverbatim

To display information about CSM utilization for TCP/IP:

\verbatim
    d net,csm,ownerid=TCPIP asid
\endverbatim

For example:

\verbatim
    d net,csm,ownerid=01f6
    IVT5508I DISPLAY ACCEPTED
    IVT5549I PROCESSING DISPLAY CSM COMMAND - OWNERID SPECIFIED
    IVT5530I BUFFER BUFFER
    IVT5551I SIZE SOURCE STORAGE ALLOCATED TO OWNER
    IVT5532I -------------------------------------------
    IVT5531I 4K ECSA 256K
\endverbatim
For additional information about the d net, csm command, refer to z/OS Communications Server: SNA Operation.

Limiting TCP/IP common and private storage utilization

You can limit the amount of common and private storage that TCP/IP can use by coding the GLOBALCONFIG parameters ECSALIMIT and POOLLIMIT in the TCP/IP profile.

- The ECSALIMIT parameter specifies the maximum amount of common storage that TCP/IP can use.
- The POOLLIMIT parameter specifies the maximum amount of TCP/IP private storage that TCP/IP can use.

The ECSALIMIT parameter ensures that TCP/IP does not overuse common storage. It can improve system reliability by limiting TCP/IP’s storage usage. The limit must account for peak storage usage during periods of high system activity or TCP/IP storage abends might occur. The limit does not include the CSM storage used by TCP/IP.

Tip: Care should be taken when coding the ECSALIMIT parameter. Setting it too low can cause TCP/IP to terminate prematurely.

Specifying a nonzero ECSALIMIT value enables warning messages EZZ4360I, EZZ4361I, and EZZ4362I to be issued when a storage shortage occurs.

If necessary, the ECSALIMIT and POOLLIMIT parameter values on the GLOBALCONFIG statement in the TCP/IP profile may be increased with a VARY TCPIP,OBEYFILE command. For additional information regarding the VARY TCPIP,OBEYFILE command, refer to z/OS Communications Server: IP System Administrator’s Commands.

Refer to z/OS Communications Server: IP Configuration Reference for more information regarding use of the GLOBALCONFIG statement ECSALIMIT and POOLLIMIT in the TCP/IP profile.

Limiting CSM storage utilization

CSM storage limits are located in the SYS1.PARMLIB member IVTPRMxx. The values you can allocate are:

- ECSA MAX - the maximum amount of ECSA storage that CSM can allocate.
- FIXED MAX - the maximum amount of fixed storage that CSM can allocate. This includes both fixed CSM ECSA and CSM data space storage.

If you do not specify values in the IVTPRMxx parmlib member, the system uses the default values of 100m ECSA and 100m FIXED. You can change these values...
dynamically with the MODIFY CSM command. If the limit specified by these values is reached, results are unpredictable. TCP/IP might not be able to continue. IVTxxxx messages will be issued if CSM is unable to obtain storage. Refer to z/OS MVS Initialization and Tuning Reference for additional information on the IVTPRMxx parmlib member.

To change your CSM settings dynamically, issue the following command:

```
MODIFY net,CSM,ECSA=value,FIXED=value
```

where `value` is in the range 1024KB - 2048MB. Additional information regarding the MODIFY command for CSM can be found in z/OS Communications Server: SNA Operation.

**Storage messages**

The following messages are issued for TCP/IP common or private storage shortage problems. The messages are documented in greater detail in z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM).

Common storage messages are as follows:

- EZZ4360I `jobname` ECSA CONSTRAINED
- EZZ4361I `jobname` ECSA CRITICAL
- EZZ4362I `jobname` ECSA EXHAUSTED
- EZZ4363I `jobname` ECSA SHORTAGE RELIEVED

Specifying a nonzero ECSALIMIT value enables warning messages EZZ4360I, EZZ4361I and EZZ4362I to be issued when a storage shortage occurs. Details regarding the TCP/IP profile GLOBALCONFIG parameter ECSALIMIT can be found in z/OS Communications Server: IP Configuration Reference.

Private storage messages are as follows:

- EZZ4364I `jobname` POOL CONSTAINED
- EZZ4365I `jobname` POOL CRITICAL
- EZZ4366I `jobname` POOL EXHAUSTED
- EZZ4367I `jobname` POOL SHORTAGE RELIEVED

Specifying a nonzero POOLLIMIT enables warning messages EZZ4364I, EZZ4365I, and EZZ4366I to be issued when a storage shortage occurs. Details regarding the TCP/IP profile, GLOBALCONFIG parameter POOLLIMIT, can be found in z/OS Communications Server: IP Configuration Reference. If storage limits were set using the TCP/IP profile GLOBALCONFIG ECSALIMIT statement or the GLOBALCONFIG POOLLIMIT statement look for the TCP/IP warning messages described previously that are issued each time a storage limit boundary is crossed. These messages might indicate a need to raise the limits.

CSM messages always start with the message prefix IVT. For a complete list of messages issued by CSM, refer to z/OS Communications Server: SNA Messages.

CSM messages identify whether the storage problem is related to CSM ECSA or CSM fixed storage. Examine the IVTPRMxx parmlib member to determine whether the limits for the particular type of CSM storage that is depleted should be increased. Issue the Display CSM command to get more details on current CSM.
allocation and limits. As previously described, CSM limits can be increased using the Modify CSM command without reloading the initial program.

For more information about the Display CSM and Modify CSM commands, refer to z/OS Communications Server: SNA Operation.

Sysplex Problem Detection and Recovery (SPDR) storage messages

Critical storage shortages for CSM, ECSA, or PRIVATE are always detected by SPDR. Storage failures (when GLOBALCONFIG LIMITS values are not coded) are detected only when an allocation in the Sysplex code fails.

Sysplex Problem Detection and Recovery (SPDR) issues one of the following messages when a storage request for common, private, or CSM storage cannot be satisfied.

- **EZD1170E**
- **EZD1187E**
- **EZZ9679E**

**EZD1170E**

```
tcpstackname WAS NOT ABLE TO GET TCP/IP storagetype STORAGE
```

When the TCP/IP profile GLOBALCONFIG statement ECSALIMIT or POOLLIMIT parameter is not coded, message EZD1170E is issued. In this situation, the storage request fails because ECSA or private storage is exhausted.

**EZD1187E**

```
tcpstackname WAS NOT ABLE TO GET TCP/IP storagetype STORAGE
```

When TCP/IP profile GLOBALCONFIG statement ECSALIMIT or POOLLIMIT is coded, EZD1187E is issued. In this situation the storage request fails because ECSA or private storage is critical.

**EZZ9679E**

```
tcpstackname DETERMINED THAT CSM WAS CRITICAL FOR AT LEAST timevalue SECONDS
```

SPDR issues EZZ9679E when CSM storage problems are detected. This message is issued when CSM storage has been critical for the configured value specified on the GLOBALCONFIG SYSPLEXMONITOR TIMERSECS parameter or the default value of 60 seconds if the parameter is not specified.

Refer to z/OS Communications Server: IP Messages Volume 2 (EZB, EZD) for debug information when these messages are issued. See z/OS Communications Server: IP Configuration Guide for more information about Sysplex Problem Detection and Recovery.

**Abends**

There are abends for each of the three types of storage problems:

**Common storage**

Common storage shortages typically result in the following abends:

- ABEND878 RC04 or RC08
- ABEND80A RC04 or RC08
For common storage problems, determine which jobs or address spaces are using an excessive amount of storage. To determine the users of common storage, enable common storage tracking (CSA Tracker). For information on how to activate and review data provided by common storage tracking, refer to \textit{z/OS MVS Initialization and Tuning Guide}. The storage totals for TCP/IP in the CSA tracker report does not reflect all the storage in use by TCP/IP. A number of TCP/IP getmains transactions are issued with the owner as SYSTEM. This storage is reported as OWNER = SYSTEM, and not OWNER = TCP/IP.

Contact the IBM Support Center for the owner of the storage that you think is causing the problem. See \textit{“Collecting documentation to submit to the IBM Support Center” on page 905} for more information.

**Private storage**

Private storage shortages typically result in the following abends:
- ABEND878 RC0C or RC10
- ABEND80A RC0C or RC10

If the problem is with TCP/IP private storage, submit a problem record with the IBM TCP/IP support team for dump analysis. See \textit{“Collecting documentation to submit to the IBM Support Center” on page 905} for more information.

**CSM storage**

For CSM storage problems, review the output from any monitoring you have been doing for CSM storage usage. Determine the largest users of the CSM ECSA and dataspace pools (4k, 16k, 32, 60k and 180k). Refer to \textit{z/OS Communications Server: SNA Operation} for additional information.

Contact the IBM Support Center for the owner of the CSM storage that you think is being used in excess. See \textit{“Collecting documentation to submit to the IBM Support Center” on page 905} for more information.

If you have not been tracking CSM storage utilization, refer to \textit{“Monitoring storage utilization” on page 900} to determine how to monitor this storage for use in problem diagnosis.

**Problem determination**

When diagnosing a storage problem, whether it is an out of storage abend condition or a storage growth problem, first determine whether the storage problem is related to common, private, or CSM storage. Use the following steps to identify the root cause of a storage problem.

**Steps to take when reviewing a storage problem**

1. Issue \texttt{D TCPIP,,STOR} and \texttt{D Net,CSM,Ownerid=All} commands to track storage usage.

2. If syslogd and TRMD are active, look in the syslogd output for messages \texttt{EZZ8662I} or \texttt{EZZ8664I}, which indicate that excessive or old data accumulating on the receive or send queue for a TCP connection. If there is not a corresponding message \texttt{EZZ8663I} or \texttt{EZZ8665I} with the same correlator value, indicating that the accumulated data has been processed, then the data might still be accumulating on the queue.
3. If syslogd or TRMD are not active, issue Netstat ALL/-A to determine if there is a lot of application data accumulating on the receive or send queues. Refer to "z/OS Communications Server: IP System Administrator’s Commands" for additional information.

4. Look in the system log for messages related to storage.

5. Run the EREP program against the SYS1.LOGREC log and review software records, looking for any storage related abends.

6. Review the messages and abends you found to determine whether they indicate a common, private, or CSM storage problem.

7. Review your storage settings for any identified problem area (for example common, CSM, or private storage).

8. Compare your current storage usage to your previous usage. If usage has increased, do you know of any situations that would cause increased usage (for example, new applications that use common storage or increased connections)?

9. Review your response to step 6. Can the storage problem be resolved by increasing your storage limits?

10. If you have a dump that was created as a result of the storage problem, proceed to step 10.

11. If no dump was taken, see "Collecting documentation to submit to the IBM Support Center" before proceeding to step 10. If you are unable to obtain a slip or console dump of the problem, a stand-alone dump may be the only method to gather documentation for the IBM Support Center. See "z/OS MVS Diagnosis: Tools and Service Aids" to determine how to take a stand-alone dump.

12. Call the IBM Support Center for assistance in reviewing the documentation.

### Collecting documentation to submit to the IBM Support Center

If your storage problem is caused by TCP/IP common or private storage usage, ensure that you have a dump of TCP/IP; IBM service will need to review it. If you did not get a system dump for the abend, or if you want to obtain a dump of TCP/IP to perform a storage analysis, use the following table for commands you can issue.

#### Table 103. Commands for various types of dumps

<table>
<thead>
<tr>
<th>Commands</th>
<th>Type of dump</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL SET,COMP=xxx,ACTION=SYCD,JOBNAMETcипроцес,SDATA=(NUC, RGN, CSA, SQA, LSQA, TRT), END</td>
<td>SLIP DUMP of the abend</td>
</tr>
<tr>
<td>where xxx equals the abend code you are receiving (for example, 878).</td>
<td></td>
</tr>
<tr>
<td>SL SET,COMP=xxx,ACTION=SYCD,JOBNAMETcипроцес,SDATA=(NUC, RGN, CSA, SQA, LSQA, TRT), END</td>
<td>SLIP dump of the abend if your storage problem is related to TCP/IP’s CSM storage usage (includes the CSM dataspace in your dump).</td>
</tr>
<tr>
<td>where xxx equals the abend code you are receiving (for example, 878).</td>
<td></td>
</tr>
<tr>
<td>DUMP COMM=(tcpip storage growth)</td>
<td>Console dump of TCP/IP</td>
</tr>
<tr>
<td>R XXX,JOBNAMETcипроцес,SDATA=(NUC, RGN, CSA, SQA, LSQA, TRT), CONT R XXX,DSPNAME=('tcипроцес', '*'), END</td>
<td></td>
</tr>
<tr>
<td>SL SET,MSGID=zzzzz,ACTION=SYCD,JOBLIST=(tcипроцес, VTAM address space name),DSPNAME=('tcипроцес', '<em>'),1.CSM=</em>, SDATA=(NUC, RGN, CSA, SQA, LSQA, TRT), END</td>
<td>MSGID slip can be used to take a dump when a particular message is issued.</td>
</tr>
<tr>
<td>where zzzzz causes a dump when the message identified in the slip is issued (i.e. CSM message IYT5562I).</td>
<td></td>
</tr>
</tbody>
</table>
Table 103. Commands for various types of dumps (continued)

<table>
<thead>
<tr>
<th>Commands</th>
<th>Type of dump</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DUMP COMM=(tcpip storage growth)</strong>&lt;br&gt;R xx,JOBNAME=('tcpipprocname'),SDATA=(NUC,RGN,CSA,SQA,LSQA,TRT),CONT&lt;br&gt;R xx,DSPNAME=('tcpipprocname'.<em>,1.CSM</em>),END</td>
<td>Console dump when the storage problem is related to CSM storage usage (includes TCP/IP and CSM dataspaces).</td>
</tr>
<tr>
<td><strong>DUMP COMM=('storage growth')</strong>&lt;br&gt;R xx,JOBNAME=('tcpipprocname',applname),CONT&lt;br&gt;R xx,SDATA=(NUC,RGN,CSA,SQA,LSQA,LPA,TRT),CONT&lt;br&gt;R xx,DSPNAME=('tcpipprocname'.*),END</td>
<td>Dump of TCP/IP and any TCP/IP related applications that may be having storage problems (for example Omproute, FTP)</td>
</tr>
</tbody>
</table>

**Note:** Wildcards (*) allow you to use a single specification to indicate a number of address spaces whose names match the wildcard pattern. This can be useful if you need to dump multiple TCP/IP stacks. You can specify a wildcard on the JOBLIST and DSPNAME parameters of a SLIP. And, the JOBNAME and DSPNAME parameters of the console dump command. For information on how to use wildcards in a SLIP and DUMP command, refer to **z/OS MVS System Commands**.

Submit the console log, and all dumps to the IBM Support Center for review.
Appendix A. First Failure Support Technology (FFST)

This appendix contains the following sections:
- “FFST probe index”
- “FFST probe information”
- “FFST probe naming conventions” on page 908
- “FFST probe descriptions” on page 908

FFST probe index

Table 104 provides an index of FFST probes by probe name and component:

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Component</th>
<th>Reference probes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBIE DST</td>
<td>IOCTL Enablement</td>
<td>IOCTL Enablement Probes</td>
</tr>
<tr>
<td>EZBPADST</td>
<td>Pascal API</td>
<td>Pascal API Probes</td>
</tr>
<tr>
<td>EZBPFDST</td>
<td>PFS IOCTL</td>
<td>PFS IOCTL Probes</td>
</tr>
<tr>
<td>EZBTR DST</td>
<td>TELNET Transform</td>
<td>TELNET Transform Probes</td>
</tr>
<tr>
<td>EZBTTT DST</td>
<td>TELNET SRV</td>
<td>TELNET SRV Probes</td>
</tr>
<tr>
<td>EZBCF DST</td>
<td>Configuration Services</td>
<td>Configuration Services Probes</td>
</tr>
<tr>
<td>EZBIT DST</td>
<td>Infrastructure</td>
<td>Infrastructure</td>
</tr>
<tr>
<td>EZBCABND</td>
<td>TCP/IP Base</td>
<td>TCP/IP Base</td>
</tr>
<tr>
<td>EZBTC DST</td>
<td>Transmission Control Protocol</td>
<td>Transmission Control Protocol Probes</td>
</tr>
<tr>
<td>EZBUDDST</td>
<td>Update Datagram Protocol Layer</td>
<td>Update Datagram Protocol Layer Probes</td>
</tr>
<tr>
<td>EZBSK DST</td>
<td>Streams</td>
<td>Streams Probes</td>
</tr>
<tr>
<td>EZBRWDST</td>
<td>Raw IP Layer</td>
<td>Raw IP Layer Probes</td>
</tr>
<tr>
<td>EZBIP DST</td>
<td>Internet Protocol</td>
<td>Internet Protocol Probes</td>
</tr>
</tbody>
</table>

FFST probe information

When a TCP/IP probe is triggered, an anomaly has occurred in the network. The process that received the condition might not complete normally. The TCP/IP program should attempt to recover from the anomaly and continues processing subsequent requests. Recovery might not be possible for some system anomalies and subsequent requests might fail, terminals might hang, and other abnormal conditions might occur.

Dump data is collected to assist in finding the source of the problem. Contact the appropriate IBM Support Center and give the service representative the console listing that is written at the time of the error, as well as the dump data produced by the probe.
**FFST probe naming conventions**

Table 105 lists the naming conventions for FFST probes used in TCP/IP.

<table>
<thead>
<tr>
<th>Characters</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 3</td>
<td>EZB</td>
<td>These characters represent the product identifier. For TCP/IP, these characters are EZB.</td>
</tr>
<tr>
<td>4, 5</td>
<td>IT</td>
<td>These characters represent the TCP/IP component identifier, IT is the component identifier for Infrastructure Services.</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>For TCP/IP, this character is usually a C.</td>
</tr>
<tr>
<td>7, 8</td>
<td>01</td>
<td>These characters represent the probe number. This number is not duplicated.</td>
</tr>
</tbody>
</table>

**FFST probe descriptions**

This section includes a table for each component that contains FFST probe instructions. The components are in alphabetical order, and the probes for each component are in alphanumeric order by probe name. Table 104 on page 907 provides an index of FFST probes in alphanumerical order by probe name. Each table in this section shows the probe name, the module that issued it, and whether the probe creates a full or minidump when triggered.

Table 106 lists the FFST probes for IOCTL enablement (EZBIECxx).

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBIEC01</td>
<td>EZBIEHOM</td>
<td>Logical interface missing</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBIEC03</td>
<td>EZBIEPRT</td>
<td>Add Portlist Member Failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBIEC04</td>
<td>EZBIECTL</td>
<td>IOCTL Command is Not 99</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBIEC05</td>
<td>EZBIECTL</td>
<td>Null Queue Pointers</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBIEC06</td>
<td>EZBIECTL</td>
<td>Invalid IOCTL Message</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBIEC07</td>
<td>EZBIEINI</td>
<td>m_begin Interval Exceeded</td>
<td>FULL</td>
</tr>
</tbody>
</table>

Guideline: When the EZBIE07 FFST probe is hit, it is recommended that you recycle the TCP/IP stack because it is not stable.

Table 107 lists the FFST probes for Infrastructure Services (EZBITCxx).

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBITC01</td>
<td>EZBITPCI</td>
<td>Connect entry failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC02</td>
<td>EZBITTUB</td>
<td>Timer cancel for BAD TQE</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC05</td>
<td>EZBITTUB</td>
<td>Timer cancel for BAD TQE2</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC07</td>
<td>EZBITDUS</td>
<td>Invalid ASCB</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC08</td>
<td>EZBITPCT</td>
<td>Entry table destroy failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC09</td>
<td>EZBPTDEF</td>
<td>Pat tree key zero</td>
<td>FULL</td>
</tr>
</tbody>
</table>
Table 107. Infrastructure services probes (continued)

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBITC10</td>
<td>EZBPTDEF</td>
<td>Pat tree key too big</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC11</td>
<td>EZBPTADD</td>
<td>Pat tree key exists</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC13</td>
<td>EZBITKRA</td>
<td>Lock release error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC15</td>
<td>EZBITKRA</td>
<td>Lock release error - DUCB</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC16</td>
<td>EZBITKRS</td>
<td>Suspend Lock Failure1</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC17</td>
<td>EZBITKRS</td>
<td>DUCB mismatch</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC18</td>
<td>EZBITKRS</td>
<td>Lock Suspend Failure2</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC19</td>
<td>EZBITSCS</td>
<td>Storage size requested error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC21</td>
<td>EZBITSMT</td>
<td>Message triple release failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC22</td>
<td>EZBITPCI</td>
<td>Create entry table failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBITC23</td>
<td>EZBITPCI</td>
<td>TRESERVE linkage index failure</td>
<td>FULL</td>
</tr>
</tbody>
</table>

Table 108 lists the FFST probes for Pascal API (EZBPACxx).

Table 108. FFST probes for Pascal API

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBPAC01</td>
<td>EZBPAISL</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC02</td>
<td>EZBPAISL</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC03</td>
<td>EZBPAMQY</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC04</td>
<td>EZBPAMQY</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC05</td>
<td>EZBPAPIN</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC06</td>
<td>EZBPAPIN</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC07</td>
<td>EZBPAPIN</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC08</td>
<td>EZBPAPIN</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC09</td>
<td>EZBPARCL</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC10</td>
<td>EZBPAROP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC11</td>
<td>EZBPAROP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC12</td>
<td>EZBPAROP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC13</td>
<td>EZBPAROP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC14</td>
<td>EZBPAROP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
</tbody>
</table>
Table 108. FFST probes for Pascal API (continued)

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBPAC15</td>
<td>EZBPAROP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC16</td>
<td>EZBPAROP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC17</td>
<td>EZBPARRV</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC18</td>
<td>EZBPARRV</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC19</td>
<td>EZBPARRV</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC20</td>
<td>EZBPARSN</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC21</td>
<td>EZBPART2</td>
<td>Function code error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC22</td>
<td>EZBPATAB</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC23</td>
<td>EZBPATAB</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC24</td>
<td>EZBPATAB</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC25</td>
<td>EZBPASTR</td>
<td>Invalid type of M_ERROR</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC26</td>
<td>EZBPASTR</td>
<td>Storage allocation failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC27</td>
<td>EZBPASTR</td>
<td>Unsupported option</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC28</td>
<td>EZBPASTR</td>
<td>Unsupported option</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC29</td>
<td>EZBPASTR</td>
<td>Unrecognized TPI</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC30</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC31</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC32</td>
<td>EZBPASTR</td>
<td>Storage allocation failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC33</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC34</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC35</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC36</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC37</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC38</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC39</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC40</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>Probe name</td>
<td>Module</td>
<td>Description</td>
<td>Dump type</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>EZBPAC41</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC42</td>
<td>EZBPASTR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC43</td>
<td>EZBPASTR</td>
<td>Storage allocation failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC44</td>
<td>EZBPASTR</td>
<td>Storage allocation failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC45</td>
<td>EZBPASTR</td>
<td>Storage allocation failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC46</td>
<td>EZBPAUCL</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC47</td>
<td>EZBPAUNR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC48</td>
<td>EZBPAUNR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC49</td>
<td>EZBPAUNR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC50</td>
<td>EZBPAURV</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC51</td>
<td>EZBPAURV</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC52</td>
<td>EZBPAURV</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC53</td>
<td>EZBPATOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC54</td>
<td>EZBPATOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC55</td>
<td>EZBPATOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC56</td>
<td>EZBPATOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC57</td>
<td>EZBPATOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC58</td>
<td>EZBPATOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC59</td>
<td>EZBPATOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC60</td>
<td>EZBPAUOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC61</td>
<td>EZBPAUOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC62</td>
<td>EZBPAUOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC63</td>
<td>EZBPAUOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC64</td>
<td>EZBPAUOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC65</td>
<td>EZBPAUOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>Probe name</td>
<td>Module</td>
<td>Description</td>
<td>Dump type</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-----------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>EZBPAC66</td>
<td>EZBPAUOP</td>
<td>TPI protocol error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC67</td>
<td>EZBPATFR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC68</td>
<td>EZBPATFR</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC69</td>
<td>EZBPATOA</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC70</td>
<td>EZBPATOA</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC71</td>
<td>EZBPATOA</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC72</td>
<td>EZBPATOA</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC73</td>
<td>EZBPATSN</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC74</td>
<td>EZBPATST</td>
<td>Streams Operation Software Error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC75</td>
<td>EZBPATTN</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC76</td>
<td>EZBPATTN</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC77</td>
<td>EZBPATTN</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC78</td>
<td>EZBPAUSN</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC79</td>
<td>EZBPAUST</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC80</td>
<td>EZBPAUST</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC81</td>
<td>EZBPATCL</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC82</td>
<td>EZBPATCL</td>
<td>Allocate storage failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC83</td>
<td>EZBPATCL</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC84</td>
<td>EZBPATCL</td>
<td>Allocate storage failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC85</td>
<td>EZBPATON</td>
<td>Streams Software Operation Error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC86</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC87</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC88</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC89</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC90</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
</tbody>
</table>
Table 108. FFST probes for Pascal API (continued)

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBPAC91</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC92</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC93</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC94</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC95</td>
<td>EZBPATON</td>
<td>TPI protocol error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC96</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC97</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC98</td>
<td>EZBPATON</td>
<td>TPI protocol error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC99</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC0A</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC0B</td>
<td>EZBPATON</td>
<td>TPI protocol error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC0C</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC0D</td>
<td>EZBPATON</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPAC0E</td>
<td>EZBPATON</td>
<td>TPI protocol error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPACA0</td>
<td>EZBPATOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPACA1</td>
<td>EZBPATOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPACA2</td>
<td>EZBPATOP</td>
<td>Streams operation software failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBPACB0</td>
<td>EZBPASTR</td>
<td>Storage allocate failure</td>
<td>FULL</td>
</tr>
</tbody>
</table>

Table 109 lists the FFST probes for PFS IOCTL (EZBPFCxx).

Table 109. PFS IOCTL probes

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBPFC01</td>
<td>EBPFIOC</td>
<td>SIOCSETTKN mismatch</td>
<td>FULL</td>
</tr>
<tr>
<td>EBPFC02</td>
<td>EBPFIOC</td>
<td>SIOCSETTKN mismatch</td>
<td>FULL</td>
</tr>
</tbody>
</table>
Table 110 lists the FFST probes for Telnet Transform (EZBTRCxx).

**Table 110. Telnet transform probes**

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBTRC01</td>
<td>EZBTRCLT</td>
<td>Unexpected transform request</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTRC03</td>
<td>EZBTRGTT</td>
<td>Terminal ID mismatch</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTRC04</td>
<td>EZBTRMST</td>
<td>Unexpected transform WorkQ request</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTRC05</td>
<td>EZBTRRTI</td>
<td>Negative transform terminal value</td>
<td>FULL</td>
</tr>
</tbody>
</table>

Table 111 lists the FFST probes for Telnet SRV (EZBTTCxx).

**Table 111. FFST probes for Telnet SRV**

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBTTC01</td>
<td>EZBTTCLS</td>
<td>Unlocatable server/vector table</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTTC02</td>
<td>EZBTTCLS</td>
<td>CVB lock failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTTC03</td>
<td>EZBTTTLT</td>
<td>Invalid TCVB token range</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTTC04</td>
<td>EZBTTTLT</td>
<td>Invalid TST entry</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTTC05</td>
<td>EZBTTTLT</td>
<td>Telnet token segment table not found</td>
<td>FULL</td>
</tr>
</tbody>
</table>

Table 112 lists FFST probes for Configuration Services (EZBCFCxx).

**Table 112. Configuration services probes**

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBCFC01</td>
<td>EZACFFST</td>
<td>Unknown configuration error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBCFC02</td>
<td>EZACFTEL</td>
<td>Bad protocol Type 1</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBCFC03</td>
<td>EZACFFST</td>
<td>Configuration bad parameters error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBCFC04</td>
<td>EZACFTEL</td>
<td>Socket closed</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBCFC05</td>
<td>EZACFTEL</td>
<td>Bad protocol Type 2</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBCFC06</td>
<td>EZACFTEL</td>
<td>Bad protocol Type 3</td>
<td>FULL</td>
</tr>
</tbody>
</table>

Table 113 lists the FFST probe for TCP/IP Base (EZBABCxx).

**Table 113. TCP/IP Base probes**

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBABC01</td>
<td>EZBCABND</td>
<td>A C abend recovery failed</td>
<td>FULL</td>
</tr>
</tbody>
</table>

Table 114 on page 915 lists the FFST probes for Transmission Control Protocol (EZBTCCxx).
### Table 114. Transmission Control Protocol probes

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBTC01</td>
<td>EZBTCSTR</td>
<td>Name on Open Does Not Match</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTC02</td>
<td>EZBTCSTR</td>
<td>Could not allocate the SID</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTC03</td>
<td>EZBTCSTR</td>
<td>Cannot Repeat Named Open</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTC04</td>
<td>EZBTCSTR</td>
<td>Hashtable Insert Failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTC05</td>
<td>EZBTCWRT</td>
<td>Not the Controlling Stream</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTC06</td>
<td>EZBTCWRT</td>
<td>Not the Controlling Stream</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBTC07</td>
<td>EZBTCWRT</td>
<td>Not the Controlling Stream</td>
<td>FULL</td>
</tr>
</tbody>
</table>

### Table 115 lists the FFST probes for Update Datagram Protocol Layer (EZBUDCxx).

### Table 115. Update Datagram Protocol Layer probes

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBUD01</td>
<td>EZBUDEXC</td>
<td>DMUX Machine Index Failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD02</td>
<td>EZBUDEXC</td>
<td>DMUX Machine Index Failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD03</td>
<td>EZBUDEXC</td>
<td>SNMP Machine Index Failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD04</td>
<td>EZBUDSTR</td>
<td>Name on Open Does Not Match</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD05</td>
<td>EZBUDSTR</td>
<td>Allocate the MUCB SID failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD06</td>
<td>EZBUDSTR</td>
<td>Stack is Already Active</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD07</td>
<td>EZBUDSTR</td>
<td>Unlock for Machine Index Failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD08</td>
<td>EZBUDSTR</td>
<td>Unlock for Machine Index Failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD09</td>
<td>EZBUDSTR</td>
<td>Unlock for Machine Index Failure</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD10</td>
<td>EZBUDWRT</td>
<td>Unknown Primitive Error Exit</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD11</td>
<td>EZBUDWRT</td>
<td>Unknown Primitive Error Exit</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD12</td>
<td>EZBUDWRE</td>
<td>Matching Prefix Error</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBUD13</td>
<td>EZBUDWRE</td>
<td>Matching Prefix Error</td>
<td>FULL</td>
</tr>
</tbody>
</table>

### Table 116 lists the FFST probes for Streams (EZBSKCxx).

### Table 116. Streams probes

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBSKC01</td>
<td>EZBSKVRB</td>
<td>Streams Are Not Functioning (TSDX_Streams_vcastint)</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBSKC02</td>
<td>EZBSKVRB</td>
<td>Unsupported Message Type</td>
<td>FULL</td>
</tr>
</tbody>
</table>
Table 117 lists the FFST probes for Raw IP Layer (EZBRWCxx).

**Table 117. Raw IP Layer probes**

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBRWC01</td>
<td>EZBRWWRI</td>
<td>WILD TPI Primitive to RAW</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBRWC02</td>
<td>EZBRWWRI</td>
<td>Invalid Messages</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBRWC03</td>
<td>EZBRWSTR</td>
<td>Name on Open Does Not Match</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBRWC04</td>
<td>EZBRWSTR</td>
<td>Could Not Allocate the MRCB SID</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBRWC05</td>
<td>EZBRWSTR</td>
<td>Stack is Already Active</td>
<td>FULL</td>
</tr>
</tbody>
</table>

Table 118 lists the FFST probes for Internet Protocol (EZBIPCxx).

**Table 118. FFST probes for Internet Protocol**

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBIPC01</td>
<td>EZBIPSTR</td>
<td>Not a Clone Open</td>
<td>FULL</td>
</tr>
</tbody>
</table>

Table 119 lists the FFST probes for the Cross-System Coupling Facility (XCF) (EZBXFCxx).

**Table 119. XCF probes**

<table>
<thead>
<tr>
<th>Probe name</th>
<th>Module</th>
<th>Description</th>
<th>Dump type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EZBXFC01</td>
<td>EZBXFINI</td>
<td>Join Failed</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBXFC02</td>
<td>EZBXFINI</td>
<td>Second Query Failed</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBXFC03</td>
<td>EZBXFINI</td>
<td>First Query Failed</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBXFC04</td>
<td>EZBXFMSI</td>
<td>MsgI Failed</td>
<td>FULL</td>
</tr>
<tr>
<td>EZBXFC05</td>
<td>EZBXFMSO</td>
<td>MsgO Failed</td>
<td>FULL</td>
</tr>
</tbody>
</table>

**Guideline:** When partitioning systems out of the sysplex, FFST problem EZBXFC05 might be seen on active systems in the sysplex. This can occur when a response is not given to IXC402D in a timely manner. To avoid this, it is suggested you setup the SFM policy to automatically partition systems from the sysplex without having to respond to IXC402D. Refer to z/OS MVS Setting Up a Sysplex for information on setting up the SFM policy.
Appendix B. Overview of internetworking

This appendix gives an overview of internetworking and contains the following sections:

- “Maximum transmission unit (MTU)” on page 918
- “Fiber Distributed Data Interface (FDDI)” on page 919
- “Token-Ring IEEE 802.5” on page 920
- “IEEE 802.3” on page 921
- “Ethernet — DIX V2” on page 921
- “Subnetwork Access Protocol (SNAP)” on page 922
- “IP routing” on page 922
- “Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6)” on page 923
- “Direct routing” on page 926
- “Indirect routing” on page 926
- “Simplified IP datagram routing algorithm” on page 926
- “IPv4 subnetting” on page 927
- “IPv6 prefixes” on page 929
- “Simplified IP datagram routing algorithm with subnets” on page 929
- “Static routing” on page 930
- “Dynamic routing” on page 931

Networking with TCP/IP connects different networks so that they form one logical interconnected network. This large overall network is called an internetwork, or more commonly, an intranet or internet. Each network uses its own physical layer, and the different networks are connected to each other by means of machines that are called gateways.

Gateways transfer IP datagrams between networks. This function is called routing; therefore, the internet gateways are often called routers. Within this appendix, the terms router and gateway are synonymous; both refer to a machine that transfers IP datagrams between different networks.

If IP datagrams are not passed properly over a bridge, none of the higher TCP/IP protocols or applications work correctly. For a discussion of bridges, refer to TCP/IP Tutorial and Technical Overview.

Linking networks in this way takes place at the network level of the International Organization for Standardization (ISO). It is possible to link networks at a lower level layer using bridges. Bridges link networks at the ISO data link layer. Bridges pass packets or frames between different physical networks regardless of the protocols contained within them. An example of a bridge is the IBM 8209, which can interconnect an Ethernet network and a token-ring network.

A bridge does not connect TCP/IP networks together. It connects physical networks together that still forms the same TCP/IP network. (A bridge does not do IP routing.)
Figure 109 depicts a router and a bridge. The router connects Network 1 to Network 2 to form an intranet.

Maximum transmission unit (MTU)

Different physical networks have different maximum frame sizes. Within the different frames, there is a maximum size for the data field. This value is called the maximum transmission unit (MTU), or maximum packet size in TCP/IP terms.

Figure 109 on page 919 shows the relationship between MTU and frame size.
If an IP datagram is to be sent out onto the network and the size of the datagram is bigger than the MTU, IP fragments the datagram into multiple fragments, so that it fits within the data fields of the frames. If the MTU is larger than the network can support, then the data is lost.

The value of MTU is especially important when bridging is used because of the different network limits. RFC 791—Internet Protocols states that all IP hosts must be prepared to accept datagrams of up to 576 bytes.

The minimum MTU for IPv6 is 1280. Refer to RFC 2460, Internet Protocol, Version 6 (IPv6) Specification for more information.

You can configure an MTU using the max_packet_size value on the GATEWAY statement or the MTU parameter on the BEGINROUTES statement.

**Fiber Distributed Data Interface (FDDI)**

The FDDI specifications define a family of standards for 100 Mbps fiber optic LANs that provide the physical layers and media access control sublayer of the data link layer, as defined by the ISO/OSI Model.

IP-FDDI defines the encapsulating of IP datagrams and ARP requests and replies in FDDI frames.

All frames are transmitted in standard IEEE 802.2 LLC Type 1 Unnumbered Information format, with the DSAP and SSAP fields of the 802.2 header set to the assigned global SAP value for SNAP (decimal 170). The 24-bit Organization Code in the SNAP header is set to zero, and the remaining 16 bits are the EtherType from Assigned Numbers:

- 2048 for IP
- 2054 for ARP

Typically, the MTU is set to 4352.
Mapping of 32-bit internet addresses to 48-bit FDDI addresses is done by the ARP dynamic discovery procedure. The broadcast internet addresses (whose <host address> is set to all ones) are mapped to the broadcast FDDI addresses (all ones).

IP datagrams are transmitted as a series of 8-bit bytes using the usual TCP/IP transmission order called “big-endian” or “network byte order.”

For more information on FDDI architecture, refer to *LAN Concepts and Products*.

**Token-Ring IEEE 802.5**

When a token-ring frame passes through a bridge, the bridge adds information to the routing information field (RIF) of the frame (assuming that the bridge supports source route bridging). The RIF contains information concerning the route taken by the frame and, more importantly, the maximum amount of data that the frame can contain within its data field. This is called the maximum information field (I-field). The value specified for the maximum I-field is sometimes referred to as the largest frame size, but this means the largest frame size, excluding headers. See Figure 111 for details on the relationship of the I-field to the header fields.

Guideline: It is important to be aware that the IBM implementation limits the number of bridges through which a frame can be passed to seven. An attempt to pass a frame through an eighth bridge fails.

The maximum I-field is always decreased by a bridge when it cannot handle the value specified. So, for a given path through a number of token-ring bridges, the maximum I-field is the largest value that all of the bridges support. This value is specified in the Routing Control (RC) field within the RIF as shown in Figure 111.

![Figure 111. Format of an IEEE 802.5 token-ring frame](image)

The size of the MTU is the maximum amount of data that is allowed within a frame. The token-ring architecture specifies the maximum value of the I-field in the data frame, which corresponds to the maximum size of the L-PDU. The maximum I-field value is determined by the bit configuration in the RC field, and is present in all routed frames.

Table 120 on page 921 shows the relationship between the RC field and the maximum I-field values.
Table 120. Relationship between RC field and maximum I-field value

<table>
<thead>
<tr>
<th>Routing control field</th>
<th>Maximum I-field in bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>x000 xxxx xxxx xxxx</td>
<td>516</td>
</tr>
<tr>
<td>x001 xxxx xxxx xxxx</td>
<td>1500</td>
</tr>
<tr>
<td>x010 xxxx xxxx xxxx</td>
<td>2052</td>
</tr>
<tr>
<td>x011 xxxx xxxx xxxx</td>
<td>4472</td>
</tr>
<tr>
<td>x100 xxxx xxxx xxxx</td>
<td>8144</td>
</tr>
<tr>
<td>x101 xxxx xxxx xxxx</td>
<td>11407</td>
</tr>
<tr>
<td>x110 xxxx xxxx xxxx</td>
<td>17800</td>
</tr>
</tbody>
</table>

Figure 111 on page 920 shows that, within the L-PDU, the Logical Link Control (LLC) header uses eight bytes. Thus the MTU value is eight bytes less than the maximum I-field. Note that the L-PDU contains a SNAP header, as described in “Subnetwork Access Protocol (SNAP)” on page 922. Follow this example to calculate the MTU for a token-ring. The token-ring bridges always adjust the value of the maximum I-field to that of the smallest one in the path. Ensure that the MTU value is less than the value specified by the bridge.

Typically, within a 4-Mbps token-ring network, the value of maximum I-field is 2052 bytes. Therefore, the MTU would be set to 2044 bytes (2052 minus eight bytes for the LLC header).

IEEE 802.3

The frame used in IEEE 802.3 Ethernet networks is shown in Figure 112.

![Figure 112. Format of an IEEE 802.3 frame](image)

The maximum size of the L-PDU for a 10Mbps network is 1500 bytes. Because eight bytes are used within the L-PDU for the LLC header, this means that the maximum size of the data field is 1492 bytes. Therefore, the MTU for IEEE 802.3 networks should be set to 1492 bytes.

Ethernet — DIX V2

The frame used in DIX Ethernet networks is shown in Figure 113.

![Figure 113. Format of an Ethernet V2 frame](image)
There is no LLC data in an Ethernet V2 frame. The maximum size for the frame is 1526 bytes. This means that the data field can be 1500 bytes maximum. The MTU for Ethernet V2 can be set to 1500 bytes.

It is possible to bridge Ethernet V2 frames to either IEEE 802.3 or IEEE 802.5 networks; an LLC header is added or removed from the frame, as required, as part of the conversion when bridging.

Subnetwork Access Protocol (SNAP)

The TCP/IP software provides protocol support down to the ISO network layer. Following this layer is the data link layer, which can be separated into two sublayers. These are the Logical Link Control (LLC) and the Media Access Control (MAC) layers.

The IEEE 802.2 standard defines the LLC sublayer, and the MAC sublayer is defined in IEEE 802.3, IEEE 802.4, and IEEE 802.5.

The format of an IEEE 802.2 LLC header with the SNAP header is shown in Figure 114.

The values of the fields in the LLC header when a SNAP header is used are specified in RFC 1042 - Standard for Transmission of IP Datagrams over IEEE 802 Networks. The values specified are:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSAP</td>
<td>X'AA'</td>
</tr>
<tr>
<td>SSAP</td>
<td>X'AA'</td>
</tr>
<tr>
<td>CONT</td>
<td>X'03' Specifies unnumbered information (UI)</td>
</tr>
<tr>
<td>P_id</td>
<td>X'00 00 00'</td>
</tr>
</tbody>
</table>
| Type    | X'8006' — ARP  
|         | X'8035' — RARP  
|         | X'86dd' — IPv6 |

IP routing

IP routing is based on routing tables held within a router or internet host. These tables contain routes which can either be static or dynamic. Typically, static routes are predefined within a configuration file, and dynamic routes are “learned” from the network, using a routing protocol.
Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6)

There are two Internet protocols used to assign addresses to links on a host, Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6). The majority of current internets use IPv4. This protocol is nearly 20 years old and is approaching the limits of the node addresses that its 32 bit addresses allow. IPv6 is the next generation of the Internet Protocol, designed to replace IPv4. Among other advantages, the 128 bit addresses defined by IPv6 provide nearly limitless addresses.

Although IPv6 is expected to eventually replace IPv4, they are likely to coexist for a number of years during the transition.

Internet Protocol Version 4 (IPv4)

A link on a host on an intranet is identified by its IP address. Internet Protocol (IP) is the protocol that is used to deliver datagrams between such hosts. It is assumed the reader is familiar with the TCP/IP protocols. Details of some of the protocols can be found in the TCP/IP Tutorial and Technical Overview. Specific information relating to the Internet Protocol can be found in RFC 791.

An IPv4 address is a 32-bit address that is usually represented in dotted decimal notation, with a decimal value representing each of the four octets (bytes) that make up the address. For example:

```
00001001010000110110000100000010 32-bit address
00001001 01000011 01100001 00000010 4 octets
9 67 97 2 dotted decimal notation (9.67.97.2)
```

The IPv4 address consists of a network address and a host address. Within the Internet, the network addresses are assigned by a central authority, the Network Information Center (NIC). The portion of the IPv4 address that is used for each of these addresses is determined by the class of address. There are three commonly used classes of IPv4 addresses (see Figure 115).

*Figure 115. Classes of IPv4 addresses*

The class of the address is determined by the first octet of the IPv4 address. Figure 116 on page 924 shows how the class of address is determined. The figure also shows Class D addresses. Class D addresses represent multicast groups, not network IP addresses. Multicast group addresses consist of the high-order, four bits of 1110 and the remaining 28 bits, which form a multicast group ID.
As shown in Figure 116, the value of the bits in the first octet determine the class of address, and the class of address determines the range of values for the network and host segment of the IPv4 address. For example, the IPv4 address 9.67.97.2 would be a class A address, since the first two bits in the first octet contain B’00’. The network part of the IPv4 address is “9” and the host part of the IPv4 address is “67.97.2”.

Refer to RFC 1166–Internet Numbers for more information about IPv4 addresses. Refer to RFC 1060–Assigned Numbers for more information about reserved network and host IPv4 addresses, such as a network broadcast address.

**Internet Protocol Version 6 (IPv6)**

As described above, IPv4 addresses are represented in dotted-decimal format. The 32-bit address is divided along 8-bit boundaries. Each set of 8 bits is converted to its decimal equivalent and separated by periods. In contrast, IPv6 addresses are 128-bits divided along 16-bit boundaries. Each 16-bit block is converted to a 4-digit hexadecimal number and separated by colons. The resulting representation is called colon-hexadecimal.

There are three conventional forms for representing IPv6 addresses as text strings:

The preferred form is x:x:x:x:x:x:x:x, where x is the hexadecimal value of the eight 16-bit pieces of the address. For example:

```
FEDC:BA98:7654:3210
```

**Guideline:** It is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field. The following is the only exception.

It is common in some styles of IPv6 addresses to contain long strings of zero bits. To make writing addresses containing zero bits easier, a special syntax is available to compress the zeros. Use two colons (::) to indicate multiple groups of 16 bits of zeros. The two colons (::) can appear only once in an address. The two colons (::) can also be used to compress the leading zeros, the trailing zeros, or both in an address.
For example, the following addresses:

- `1080:0:0:0:8:800:200C:417A`: a unicast address
- `FF01:0:0:0:0:0:0:101`: a multicast address
- `0:0:0:0:0:0:0:1`: the loopback address
- `0:0:0:0:0:0:0:0`: the unspecified addresses

can be represented as:

- `1080::8:800:200C:417A`: a unicast address
- `FF01::101`: a multicast address
- `::1`: the loopback address
- `::`: the unspecified addresses

An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is `x:x:x:x:x:x:d.d.d.d`, where `x` is the hexadecimal value of the six high-order 16-bit pieces of the address, and `d` is the decimal value of the four low-order 8-bit pieces of the address (standard IPv4 representation). For example, `0:0:0:0:0:13.1.68.3` can be expressed in condensed form as `::13.1.68.3`.

Figure 117 shows a simple network with a bridge and a router.
Machine D is acting as an IP router and transfers IP datagrams between the class C, 192.9.200, network and the class A, 9.67.32 network. It is important to note that for Machine B to communicate with Machine C using TCP/IP, both Machine D and the bridge have to be correctly configured and working.

TCP/IP uses the HOME statements, defined in the data set hlq.PROFILE.TCPIP, to assign home addresses and associated link names. HOME statements can be updated using the VARY TCPIP command. Refer to the z/OS Communications Server: IP Configuration Reference for more information about both the HOME statements.

Direct routing

Direct routing can take place when two hosts are directly connected to the same physical network. This can be a bridged token-ring network, a bridged Ethernet, or a bridged token-ring network and Ethernet. The distinction between direct routing and indirect routing is that, with direct routing, an IP datagram can be delivered to the remote host without subsequent interpretation of the IP address, by an intermediate host or router.

In Figure 117 on page 925 a datagram traveling from Machine A to Machine B would be using direct routing, although it would be traveling through a bridge.

Indirect routing

Indirect routing takes place when the destination is not on a directly attached IP network, forcing the sender to forward the datagram to a router for delivery.

In Figure 117 on page 925 a datagram from Machine A being delivered to Machine C would be using indirect routing, with Machine D acting as the router (or gateway).

Simplified IP datagram routing algorithm

To route an IP datagram on the network, the algorithm shown in Figure 118 on page 927 is used.
Using this general routing algorithm, it is very easy to determine where an IP datagram is routed. Following is a simple example based on the configuration shown in Figure 117 on page 925.

Machine A IP Address = 192.9.200.1

Routing Table

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.9.200.1</td>
<td>192.9.200.1 (Machine A's network interface)</td>
</tr>
<tr>
<td>9.0.0.0</td>
<td>192.9.200.2 (Route to the 9.n.n.n address is via Machine D, 192.9.200.2)</td>
</tr>
</tbody>
</table>

Machine A sends a datagram to host 192.9.200.3 (Machine B), using the direct route, 192.9.200.1 (its own network interface). Machine A sends a datagram to host 9.67.32.2 (Machine C), using the indirect route, 192.9.200.2 (Machine D), and Machine D then forwards the datagram to Machine C.

**IPv4 subnetting**

IPv4 allows for a variation of the network and host segments of an IP address, known as subnetting, can be used to physically and logically design a network. For example, an organization can have a single internet network address (NETID) that is known to users outside the organization, yet configure its internal network into different departmental subnets. Subnetwork addresses enhance local routing capabilities, while reducing the number of network addresses required.

To illustrate this, consider a simple example. Assume that we have an assigned class C network address of 192.9.200 for our site. This would mean that we could have host addresses from 192.9.200.1 to 192.9.200.254. If we did not use subnetting, then we could only implement a single IP network with 254 hosts. To split our site into two logical subnetworks, we could implement the network scheme shown in Figure 119 on page 925.
z/OS TCP/IP uses a slightly different scheme for the subnet mask when defining the BEGINROUTES statements in the hlq.PROFILE.TCPIP data set and for displaying the subnet mask within a Netstat GATE/-g command. The subnet mask is applied only to the host segment of the IP address, and Netstat GATE/-g displays the subnet mask for only the host segment of the IP address. The subnet mask in the preceding chart as defined for z/OS TCP/IP would be:

0 0 0 192 0.0.0.192
00000000 00000000 00000000 11000000

Although z/OS TCP/IP defines the subnet mask differently, the application of the subnet mask and subnet value to the IP address is consistent with RFC-architected routing algorithms. A subnet mask of 255 is used for the remainder of this section, to retain symmetry with other routing documents that use 255 as the subnet value for the network segment of an IP address.

Because subnets B'00' and B'11' are both reserved, only two subnets are available. All 0s and all 1s have a special significance in internet addressing and should be used with care. Also notice that the total number of host addresses that we can use is reduced for the same reason. For instance, we cannot have a host address of 16 because this would mean that the subnet/host segment of the address would be B'0001000', which with the subnet mask we are using, would mean a subnet value of B'00', which is reserved.

The same is true for the host segment of the fourth octet. A fourth octet value of B'01111111' is reserved because, although the subnet of B'01' is valid, the host value of B'1' is reserved.

The network segment of the subnet mask is always assumed to be one, so each octet has a decimal value of 255. For example, with a class B address, the first two octets are assumed to be 255,255.
**IPv6 prefixes**

The IPv6 prefix concept is similar to IPv4 subnetting. An IPv6 address with a prefix is written as an IPv6 address followed by a decimal number representing the number of bits in the address that constitute the prefix. It is written as:

```
ipv6-address/prefix-length
```

where:

- **ipv6-address** is an IPv6 address in any notation
- **prefix-length** is a decimal value specifying how many of the leftmost contiguous bits of the address comprise the prefix.

For example, the following are legal representations of the 60-bit prefix 12AB:00000000CD3 (hexadecimal):

- `12AB::00000CD30:0000:0000:0000/60`
- `12AB::CD30:0:0:0/60`
- `12AB::0:CD30::/60`

When writing both a node address and a prefix of that node address (for example, the node subnet prefix), the two can be combined as follows:

The node address

```
12AB:0:0:CD30:123:4567:89AB:CDEF
```

and its subnet number

```
12AB:0:0:CD30::/60
```

can be abbreviated as

```
12AB:0:0:CD30:123:4567:89AB:CDEF/60
```

**Simplified IP datagram routing algorithm with subnets**

When subnetting is used, the algorithm required to find a route for an IP datagram is similar to the one for general routing, with the exception that the addresses being compared are the result of a logical AND of the subnet mask and the IP address.

For example:

```
IP address: 9.67.32.18 00001001 01000011 00100000 00010010
<AND>
Subnet Mask: 255.255.255.240 11111111 11111111 11111111 11110000
```

```
Result of Logical AND: 9.67.32.16 00001001 01000011 00100000 00010000
```

The subnet address is 9.67.32.16, and it is this value that is used to determine the route used.

[Figure 120 on page 930](#) shows the routing algorithm used with subnets.
Figure 120. Routing algorithm with subnets

Figure 121 shows how a subnet route is resolved.

**Static routing**

Static routing, as the name implies, is defined within the local host, and must be manually changed as the network changes. Typically, a configuration file contains the definitions for directly-attached networks, routes for specific hosts, and a possible default route that directs packets to a destination for networks that are not previously defined.

Static routes can be defined using either the z/OS TCP/IP GATEWAY or BEGINROUTES statements to configure the internal routing tables; these
Dynamic routing

Dynamic routing is the opposite of static routing. A TCP/IP protocol is used to dynamically update the internal routing tables when changes to the network occur.

IPv4

For IPv4, there are two dynamic routing protocols available. One routing protocol is the Routing Information Protocol (RIP). It is implemented by the OMPROUTE routing applications. A newer protocol is open shortest path first (OSPF). It is implemented by OMPROUTE only. For more details about OMPROUTE, see Chapter 32, “Diagnosing OMPROUTE problems,” on page 749. For configuration information about both applications, refer to the z/OS Communications Server: IP Configuration Reference.

IPv6

For IPv6, dynamic routing is performed by the Router Discovery protocol and by the IPv6 OSPF and IPv6 RIP dynamic routing protocols of OMPROUTE. For more information about IPv6 dynamic routing, refer to the z/OS Communications Server: IP Configuration Guide.
Appendix C. IKE protocol details

This appendix gives an overview of the IKE daemon and contains the following sections:

- “Negotiating security associations”
- “ISAKMP Main mode limitations” on page 947
- “Commit-bit support in the IKE daemon” on page 948

Negotiating security associations

This section outlines how the ISAKMP and IKE protocols are used to negotiate security associations (SAs) and exchange keys between two systems that want to communicate securely.

Overview of negotiating security associations

The ISAKMP protocol is a framework for dynamically establishing security associations and cryptographic keys in an Internet environment. This framework defines a set of message flows (exchanges) and message formats (payloads). ISAKMP defines a generic payload for key exchange information. This enables the ISAKMP protocol to manage cryptographic keys independent of the key exchange protocol that is used to generate them.

ISAKMP defers the interpretation of the key exchange payload to individual key exchange protocols. Internet Key Exchange (IKE) is such a protocol. IKE augments the ISAKMP protocol to facilitate the creation of authenticated keying material. IKE defines how keying material is generated. The exchanges that are defined by ISAKMP require authentication to take place, but they do not specify how authentication is to be performed. IKE defines how authentication is to be performed.

ISAKMP defines two phases of negotiation. Both of these phases are also applicable to the IKE protocol. The first phase is referred to as phase 1. In phase 1, two ISAKMP servers agree on how to protect traffic between themselves. This agreement results in the creation of an ISAKMP security association. The second phase is referred to as phase2. In phase2, security associations for other security protocols are established; for example, AH or ESP. Negotiations during each phase are accomplished using an ISAKMP-defined exchange or by an exchange that is specific to a key exchange protocol.

Phase 1

IKE supports two types of phase 1 exchanges:

- Main mode
- Aggressive mode

Both of these exchange modes are based on exchanges that are defined by ISAKMP. Main mode is an implementation of ISAKMP’s Identity Protect exchange. Aggressive mode is an implementation of ISAKMP’s Aggressive exchange.

IKE defines four techniques for authentication of phase 1 exchanges:

- Pre-shared key
- Signature-based
- Public key encryption
- Revised public key encryption

**Restriction:** Of these techniques, the z/OS IKE daemon supports only pre-shared key authentication and signature-based authentication using RSA signatures.

**Main mode:** A Main mode exchange is comprised of six messages as shown in [Figure 122](#).

![Figure 122. Main mode exchange](#)

Messages 1 and 2 provide agreement on the negotiable attributes of the ISAKMP security association. These associations are used to protect phase 2 negotiations that are established using this phase 1. The initiator sends a list of acceptable security associations to the responder in message 1. Each security association defines an acceptable combination of attributes for the ISAKMP SA that is being negotiated. The responder picks a security association that is acceptable and returns the choice to the initiator in message 2.

The following attributes can be negotiated in phase 1:
- Authentication method (for example, pre-shared key or RSA signature)
- Hash algorithm (for example, MD5 or SHA1)
- Encryption algorithm (for example, DES, 3DES or AES)
- Diffie-Hellman group information (for example, group 1, group 2, group 5 or group 14)
- Life time and life size of the ISAKMP SA

Messages 3 and 4 are used to exchange information specific to the generation of a shared secret key. This information includes Diffie-Hellman public values and a randomly generated value called a nonce. The initiator sends his Diffie-Hellman public value (for example, $g^{**x \ mod \ n}$) and a nonce in message 3. The responder...
sends a Diffie-Hellman public value (for example, \( g^y \mod n \)) and a nonce in message 4. With this information, both the responder and initiator can independently generate the identical keying information. The calculations that are used to generate keying information vary depending on the authentication method that was agreed upon during messages 1 and 2.

The keying information that is generated by both sides includes the following:

- A key that authenticates messages sent under the protection of this ISAKMP SA (for example, phase 2 messages)
- A key that encrypts messages that are sent under the protection of this ISAKMP SA (for example, phase 2 messages)
- Keying material that derives keys that are established for phase 2 SA

Messages 5 and 6 are used to exchange identity information and authentication information. The authentication information varies depending on the authentication method that was agreed upon during messages 1 and 2. For pre-shared key authentication, public key encryption authentication, and revised public key encryption authentication, the information takes the form of an encrypted hash. For signature based authentication, this information takes the form of a signature. The initiator includes his identity and authentication information in message 5. The responder includes their identity and authentication information in message 6.

Main mode provides a mechanism to exchange certificates when signature-based authentication is used. This mechanism is not shown in Figure 122 on page 934 but works in the following way. In message 5 the initiating ISAKMP server can include the certificate it used to create its signature. In message 6 the responding ISAKMP server might include the certificate it used to create its signature. Inclusion of the certificate is optional unless the ISAKMP server’s peer explicitly requests that the certificate be sent.

**Aggressive mode:** An aggressive mode exchange is comprised of three messages, as shown in Figure 123 on page 936.
Aggressive mode exchanges the same information as Main mode, with the exception of the following:

- In Aggressive mode, the initiator can send only one proposal. In Main mode, the initiator can send a list of proposals.
- In Aggressive mode, only three messages are exchanged instead of six messages as in Main mode.
  - Message 1 of Aggressive mode contains all the information that was contained in messages 1 and 3 of Main mode, plus the identity information sent in message 5 of Main mode.
  - Message 2 of Aggressive mode contains all the information sent in messages 2, 4, and 6 of Main mode.
  - Message 3 of Aggressive mode contains the authentication information that was contained in message 5 of Main mode.
- In Aggressive mode, no messages are required to be encrypted. Message 3 can be sent encrypted, but doing so provides little additional protection. In Main mode, messages 5 and 6 are required to be encrypted. The ISAKMP servers send their identity in messages 5 or 6 of Main mode. The result is that Main mode protects the identity of the ISAKMP servers while Aggressive mode does not. Aggressive mode provides a mechanism to exchange certificates when signature-based authentication is used. This mechanism is not shown in Figure 123, but works in the following way. In message 2 the responding ISAKMP server can include the certificate it used to create its signature. In message 3, the initiating ISAKMP server can include the certificate it used to create its signature. Inclusion of the certificates is optional unless the peer of the ISAKMP server explicitly requests that the certificate be sent.

Interpreting IKE daemon phase 1 SA states: The two IKE modes for negotiating phase 1 SAs (main and aggressive) are not themselves negotiable SA attributes. The initiator determines the mode based on the initiator's local policy. The responder can accept or reject the negotiation mode that is selected by the initiator.
Figure 124 shows how to interpret phase 1 SA states in Main mode.

The following state descriptions apply to the Communications Server IKE daemon when acting as the initiator or responder of a main mode phase 1 SA negotiation. These states are shown in the state field of the `ipsec -k display` command output. See “Main mode” on page 934 for a description of the contents of the messages. The numbers in the following list correspond to the numbered items in Figure 124.

1. The INIT state on the initiator side indicates that message 1 has not yet been sent.
2. The INIT state on the responder side indicates that the responder is processing message 1, which was received from the initiator.
3. This WAIT SA state indicates that the initiator has sent message 1 and is waiting for message 2 from the responder.
4. The WAIT KE state indicates that the responder has processed message 1 and is waiting for message 3 from the initiator.
5. The IN KE state on the initiator side indicates that the initiator has sent message 3.
6. The IN KE state on the responder side indicates that the responder has received message 3.
7. The DONE state on the initiator side indicates that the initiator has received message 6.
8. The DONE state on the responder side indicates that the responder has sent message 6.

Figure 125 shows how to interpret phase 1 SA states in aggressive mode.

---

**Aggressive Mode (Phase 1)**

- **Initiator**
  - **INIT**
  - **WAIT SA**
  - **IN KE**
  - **DONE**

- **Responder**
  - **INIT**
  - **IN KE**
  - **DONE**

**Message Flow:**

1. **INIT**
   - The INIT state on the initiator side indicates that message 1 has not yet been sent.
2. **WAIT SA**
   - The INIT state on the responder side indicates that the responder is processing message 1 received from the initiator.
3. **IN KE**
   - The WAIT SA state on the initiator side indicates that the initiator has sent message 1.
4. **DONE**
   - The IN KE state on the initiator side indicates that the initiator has processed message 1.
5. **IN KE**
   - The IN KE state on the responder side indicates that the responder has received message 2.
6. **DONE**
   - The DONE state on the initiator side indicates that the initiator has sent message 3.

---

*Figure 125. Interpreting phase 1 SA states in Aggressive mode*
7. The DONE state on the responder side indicates that the responder has received message 3.

**Phase 2**
IKE supports one type of phase 2 exchange, Quick mode. Quick mode is an IKE-specific exchange. It is not based on an ISAKMP-defined exchange. Quick mode exchanges are bound to a specific phase 1 exchange. This is accomplished by encrypting a hash of each Quick mode message with a cryptographic key derived during the phase 1 exchange. No explicit authentication of the identities involved in a phase 2 exchange is performed.

**Quick mode:** A Quick mode exchange is comprised of three messages, as shown in Figure 126.

![Figure 126. Quick mode exchange messages](image)

In Quick mode, each message contains an encrypted hash. This hash authenticates the source of the message (for example, verifies that it is bound to an ISAKMP SA), authenticates the integrity of the message, and proves liveliness. In message 1, the initiator sends a list of acceptable proposals to the responder. Each proposal defines an acceptable combination of attributes for the non-ISAKMP SA that is being negotiated (AH or ESP SA). The responder picks a proposal that is acceptable and returns the choice to the initiator in message 2.

The attributes that can be negotiated in Quick mode include the following:
- Protocol (AH, ESP, or both AH and ESP)
- Authentication algorithm (for example, Hmac-Md5 or Hmac-Sha)
- Encapsulation mode (tunnel or transport)
- Encryption algorithm (for example, DES, 3DES or AES)
- Diffie-Hellman group information (for example, group 1, group 2, group 5 or group 14)
Life time and life size of the IPSec SA

Quick mode enables an optional Diffie-Hellman exchange to occur. When the Diffie-Hellman exchange is to take place, the initiator includes a Diffie-Hellman public value (for example, $g^x \mod n$) in message 1, and the responder includes a Diffie-Hellman public value (for example, $g^y \mod n$) in message 2. The key generated from this Diffie-Hellman exchange is used in the calculation that generates the keying material for the non-ISAKMP SA. The Diffie-Hellman exchange provides perfect forward secrecy (PFS).

Quick mode with commit bit: The ISAKMP protocol defines a bit in the ISAKMP message header known as the commit bit. When the commit bit is turned on during a Quick mode exchange, the responder should acknowledge the receipt of message 3. The responder does this by extending the Quick mode exchange to include a fourth message. Figure 127 shows this new message, which includes an encrypted hash along with a notify payload indicating that message 3 was received.

![Diagram](image)

* message must be encrypted

Figure 127. Quick mode exchange with commit-bit support

In a normal Quick mode exchange, the initiator can start using a newly negotiated SA immediately after sending message 3. The responder does not start using the newly negotiated SA until it receives message 3. Message 3 is sent using UDP. Because UDP is not a reliable protocol, it is possible that the initiator sends message 3 and that this message never gets processed by the responder. In this case, the responder retransmits message 2 back to the initiator, causing the initiator to retransmit message 3. Unfortunately, during the period of time between such
retransmissions, the initiator might start using the SA to protect an IP packet. Any such packet would be discarded by the responder until it successfully processed message 3.

In a Quick mode exchange with commit processing, the initiator defers the usage of a newly negotiated SA until one of the following events occur:

- The initiator receives a connected notify message
- The initiator receives an IP packet that was protected with the SA

The responder continues to start using the newly negotiated SA when it receives message 3. This eliminates the window where one side might start using an SA before the other side knows that it is safe to use the SA.

On z/OS, an SA is considered to be in a pending state while the initiator is waiting for a connected notify message (for example, message 4). An SA is placed into a pending state only if another SA that could be used to protect outbound traffic exists. An SA in pending state remains in pending state until one of the following events occur:

- A connected notify is received
- A message protected by the SA is received
- The last usable SA expires

**Interpreting IKE daemon phase 2 SA states:** Commit-bit support is not a negotiable phase 2 SA attribute. The Communications Server IKE daemon always includes the commit bit when initiating a Quick mode negotiation. If the responder does not support commit-bit processing, the Communications Server IKE daemon does not wait for a connected notify message from the responder. If the initiator does not have commit-bit support, then the Communications Server IKE daemon does not send a connected notify message when acting as the responder.

**Quick mode (phase 2) SA states without commit-bit support:** Figure 128 on page 942 shows interpreting Quick mode (phase 2) SA states without commit-bit support
The following state descriptions apply to the Communications Server IKE daemon when acting as the initiator or responder of a Quick mode (phase 2) SA negotiation without commit-bit support (Figure 128). These states are shown in the state field of the **ipsec -y display -b** command output. See “Quick mode” on page 939 for a description of the contents of the messages.

1. The INIT state on the initiator side indicates that message 2 has not yet been received.
2. The INIT state on the responder side indicates that the responder has not yet sent message 2.
3. A phase 1 SA must be established between the initiator and responder before the initiator can send message 1 of Quick mode. The PENDING state indicates that the initiator is waiting for a phase 1 negotiation to complete with the responder. For more information, “Interpreting IKE daemon phase 1 SA states” on page 936. After the phase 1 negotiation completes, message 1 of Quick mode can be sent.
4. The KEP state on the initiator side indicates that message 2 has been received.
5. The KEP state on the initiator side indicates that message 2 has been sent.
6. The DONE state on the initiator side indicates that message 3 has been sent.
7. The DONE state on the responder side indicates that message 3 has been received.

**Figure 128. Quick (phase 2) SA states without commit-bit support**

Quick mode (phase 2) SA states with commit-bit support: [Figure 129 on page 943] shows interpreting Quick mode (phase 2) SA states with commit-bit support.
The following state descriptions apply to the Communications Server IKE daemon when acting as the initiator or responder of a Quick mode (phase 2) SA negotiation with commit-bit support (Figure 129). These states are shown in the state field of the `ipsec -y display -b` command output. See “Quick mode with commit bit” on page 940 for a description of the contents of the messages.

1. The INIT state on the initiator side indicates that message 2 has not yet been received.
2. The INIT state on the responder side indicates that the responder has not yet sent message 2.
3. A phase 1 SA must be established between the initiator and responder before the initiator can send message 1 of Quick mode. The PENDING state indicates that the initiator is waiting for a phase 1 negotiation to complete with the responder. For more information, see “Interpreting IKE daemon phase 1 SA states” on page 936. After the phase 1 negotiation completes, message 1 of Quick mode can be sent.
4. The KEP state on the initiator side indicates that message 2 has been received.
5. The KEP state on the responder side indicates that message 2 has been sent.

Figure 129. Quick (phase 2) SA states with commit-bit support
6. The NOTIFY state indicates that the initiator has sent message 3 and is waiting for message 4.
7. The DONE state on the initiator side indicates that message 4 has been received.
8. The DONE state on the responder side indicates that message 4 has been sent.

**Traversing a NAT**

There are several incompatibility issues that exist between IPSec and Network Address Translation (NAT). These incompatibility issues are described in RFC 3715, "IPsec-Network Address Translation (NAT) Compatibility Requirements." Two RFCs were written to address these incompatibility issues:

- RFC 3947 "Negotiation of NAT-Traversal in the IKE"
- RFC 3948 "UDP Encapsulation of IPsec ESP Packets"

Both of these RFCs have been implemented on z/OS, providing z/OS with the capability to perform IPSec while traversing a NAT in a limited set of environments. RFC 3947 augments IKE's Main mode, Aggressive mode, and Quick mode messages flows to include additional information. It also provides for the negotiation of two new encapsulation modes.

To provide the possibility of interoperability with some pre-RFC implementations, z/OS also provides support for the following pre-RFC "Negotiation of NAT-Traversal in the IKE" drafts:

- draft-ietf-ipsec-nat-t-ike-02
- draft-ietf-ipsec-nat-t-ike-03

**Impacts to phase 1 (Main and Aggressive mode)**

RFC 3947 requires that a vendor ID payload containing a NAT traversal vendor ID be exchanged between two IKE peers. The vendor ID payload is an existing ISAKMP payload. The vendor ID payload is used by an IKE daemon to advertise support for a feature that is an extension to RFC 2408 (ISAKMP) and RFC 2409 (IKE). The vendor ID that is contained in the payload identifies the feature. The NAT traversal vendor ID is defined to be an MD5 hash of the vendor string RFC 3947.

The NAT traversal vendor ID must be received before an IKE daemon can send any of the new payloads and encapsulation modes that are defined in RFC 3947. Likewise, an IKE daemon should not send any of the new payloads and encapsulation modes defined in RFC 3947 without first sending the NAT traversal vendor ID.

If the initiator of a phase 1 negotiation wants to advertise support for RFC 3947, it must send the NAT traversal vendor ID in message 1 of a Main mode exchange or message 1 of an Aggressive mode exchange. If the responder of a phase 1 negotiation wants to advertise support for RFC 3947, it must send the NAT traversal vendor ID in message 2 of a Main mode exchange or message 2 of an Aggressive mode exchange.

z/OS provides limited support for several pre-RFC drafts, as well as additional z/OS-to-z/OS NAT traversal capabilities. Unique vendor IDs are used to identify these various levels of NAT traversal support.

Table 121 on page 945 shows the NAT traversal vendor IDs that are recognized by z/OS. The vendor IDs are listed from least functional to most functional. If z/OS receives multiples of these IDs, it uses the most functional level of support that it received.
Table 121 lists vendor ID strings.

<table>
<thead>
<tr>
<th>Vendor ID string</th>
<th>Vendor ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>draft-ietf-ipsec-nat-t-ike-02\n</td>
<td>90cb8091 3eb6b696 e086381b5 ec427b1f</td>
</tr>
<tr>
<td>draft-ietf-ipsec-nat-t-ike-02</td>
<td>cd604643 35df21f8 7cfd2b2c 68b6a448</td>
</tr>
<tr>
<td>draft-ietf-ipsec-nat-t-ike-03</td>
<td>7d9419a6 5310ca6f 2c179d92 15529d56</td>
</tr>
<tr>
<td>RFC 3947</td>
<td>4a131c81070358455c5728f20e95452f</td>
</tr>
<tr>
<td>z/OS CS-IKE NAT Traversal Level 1</td>
<td>95305bb5 64b82a30b 66968bbc 5326a8d</td>
</tr>
</tbody>
</table>

In z/OS, NAT traversal support can be enabled or disabled with the AllowNat parameter. The AllowNat parameter can be specified on the KeyExchangePolicy statement, the KeyExchangeAction statement of the IPSec Policy file, or both. When AllowNat is set to NO the z/OS IKE daemon does not send NAT traversal vendor IDs. Refer to z/OS Communications Server: IP Configuration Reference for additional details about the AllowNat parameter.

RFC 3947 defines a mechanism for discovering the existence of NAT devices residing between two IKE daemons, as well as the location of the NAT devices. This mechanism is the NAT Discovery (NAT-D) payload. The NAT-D payload is an extension to RFC 2408 and 2409. It contains a hash of several pieces of information including an IP address and port value from the IP packet that is being sent to an IKE peer (for example, the packet containing the NAT-D payload).

Each IKE peer sends two or more NAT-D payloads. The destination IP address and port of the outbound IKE packet are used to construct the hash that is contained within the first NAT-D payload. The source IP address and port of the outbound IKE packet are used to construct the hash that is contained within the second NAT-D payload. Normally, only two NAT-D payloads are exchanged; however, if the sender of the packet has multiple IP addresses and it does not know which IP address is used to send the packet, it can send a NAT-D payload for each IP address it owns.

The initiator of a phase 1 negotiation must send its NAT-D payloads in message 3 of a Main mode exchange or message 3 of an Aggressive mode exchange. The responder of a phase 1 negotiation must send its NAT-D payloads in message 4 of a Main mode exchange or message 2 of an Aggressive mode exchange.

**Impacts to phase 2 (Quick mode)**

RFC 3947 defines two new encapsulation mode values: UDP-Encapsulated-Transport and UDP-Encapsulated-Tunnel. These new encapsulation modes are defined in RFC 3948. Refer to z/OS Communications Server: IP Configuration Guide for a description of these new modes.

When one or more NAT devices are detected between two IKE peers, messages 1 and 2 of a Quick mode exchange should not utilize offers containing tunnel or transport mode of encapsulation. Offers containing UDP-Encapsulated-Transport or UDP-Encapsulated-Tunnel mode of encapsulation should be used instead. Likewise, when no NAT devices are detected between two IKE peers messages 1 and 2 of a Quick mode
exchange should not utilize offers containing UDP-Encapsulated-Transport or UDP-Encapsulated-Tunnel mode of encapsulation.

On z/OS, only the tunnel or transport mode of encapsulation can be specified on the IpDataOffer statement (refer to z/OS Communications Server: IP Configuration Reference). The decision to use UDP-Encapsulated-Transport or UDP-Encapsulated-Tunnel mode is made heuristically by the IKE daemon. When a NAT is detected between two IKE peers, the z/OS IKE daemon converts IpDataOffer statements containing tunnel mode encapsulation to UDP-Encapsulated-Tunnel mode and IpDataOffers containing transport mode encapsulation to UDP-Encapsulated-Transport mode.

In order to facilitate incremental TCP and UDP checksum verification, RFC 3947 requires that IKE peers exchange their view of each others IP addresses when sending SA offers containing UDP-Encapsulated-Transport mode encapsulation. RFC 3947 defines a new payload for this purpose. This new payload is the NAT Original Address (NAT-OA) payload. The NAT-OA payload is an extension of RFC 2408 and 2409. It contains an IP address.

When the initiator of a Quick mode exchange sends a proposal utilizing UDP-Encapsulated-Transport mode, RFC 3947 requires the initiator to send two NAT-OA payload in message 1. The first NAT-OA payload contains the initiator’s view of their IP address. The second NAT-OA payload contains the initiator’s view of the responder’s IP address.

When the responder of a Quick mode exchange accepts a proposal utilizing UDP-Encapsulated-Transport mode, RFC 3947 requires the responder to send two NAT-OA payloads in message 2. The first NAT-OA payload contains the responder’s view of the initiator’s address. The second NAT-OA payload contains the responder’s view of his address.

In pre-RFC 3947 drafts, only one NAT-OA payload can be sent in messages 1 and 2 of a Quick mode exchange. Sending this NAT-OA payload was recommended when sending a proposal utilizing UDP-Encapsulated-Transport encapsulation, but not required. In message 1, it contained the initiator’s view of his IP address. In message 2, it contained the responder’s view of his IP address.

### Utilizing port UDP 4500

In order to avoid any problems that could arise by IPSec-aware NAT devices, RFC 3947 requires the initiator to utilize UDP port 4500 to send and receive IKE traffic after the initiator detects the existence of a NAT device. In Main mode, the initiator detects the existence of a NAT when processing message 4 and switches to a source port of UDP 4500 and a destination port of 4500 when sending message 5. In Aggressive mode, the initiator detects the existence of a NAT when processing message 2 and switches to a source port of UDP 4500 and a destination port of UDP 4500 when sending message 3. When the responder sends the initiator a message it must use the port values from the last message that was received from the initiator.

After the initiator switches to port 4500, which is known as port floating, all subsequent messages must use the floated ports. The initiator always expects to send and receive messages on source port 4500 and destination port 4500. For the responder, if the remote peer is located behind a NAPT, the source port may have been changed to a value other than 4500. If so, the responder receives a message on a random source port Y and
destination port 4500. After receiving this message, the responder sends subsequent messages using a source port of 4500 and destination port of Y. This includes all Quick mode and informational exchange messages, as well as all future Main mode and Aggressive mode messages (including messages sent to refresh an ISAKMP security association).

These ports are also used to send UDP-encapsulated ESP traffic. In order to be able to distinguish UDP encapsulated ESP traffic from IKE traffic, a non-ESP marker is added to each IKE message sent using the UDP encapsulation ports. A non-ESP marker is 4 bytes of 0. Figure 130 shows an IKE packet with and without the non-ESP marker.

**Figure 130. IKE packet with and without the non-ESP marker**

### ISAKMP Main mode limitations

This section contains information about three Main mode scenarios.

#### Main mode scenario 1

Key policy definition is based on the identities of remote ISAKMP servers. Unfortunately, during a Main mode exchange the responding ISAKMP server must accept a key proposal prior to learning the identity of the initiating ISAKMP server. The responder must later verify that the proposal that was agreed to is acceptable with defined policy when the identity becomes known.

The z/OS IKE daemon handles this limitation as follows:

1. Upon receipt of message 1, the IKE daemon uses the IP address of the initiator and responder to find an applicable KeyExchangeRule, which encapsulates the key policy. At this point:
   - If an applicable KeyExchangeRule is found, it is considered tentative until the identity of the initiator becomes known.

2. Upon receipt of message 5, which includes the initiator’s identity, the IKE daemon uses the IP address of the initiator, the IP address of the responder and the identity of the initiator to find an applicable KeyExchangeRule. At this point:
   - If a KeyExchangeRule is not found or is found but is inconsistent with the proposal accepted in message 1, the negotiation fails.
   - If a KeyExchangeRule is found and is consistent with the proposal accepted in message 1, it is considered final, and the negotiation proceeds.
Main mode scenario 2
Pre-shared keys are defined based on the identities of ISAKMP servers. Ideally, pre-shared keys should be unique between ISAKMP server pairs. Unfortunately, during a Main mode exchange the responding ISAKMP server must determine the pre-shared key to use prior to learning the identity of the initiating ISAKMP server.

The z/OS IKE daemon handles this limitation as follows:
1. A key proposal is selected as described in "Main mode scenario 1" on page 947.
2. If the selected key proposal indicates pre-shared key mode authentication, then the IKE daemon must use a pre-shared key in order to generate message 4.
3. Upon receipt of message 5, the IKE daemon must use the same pre-shared key to decrypt the message in order to learn the identity of the initiating ISAKMP server.
4. After message 5 is successfully decrypted, the IKE daemon uses the IP address of the initiator, the IP address of the responder, and the identity of the initiator to find an applicable KeyExchangeRule. At this point:
   • If a KeyExchangeRule is not found or is found but is inconsistent with the proposal accepted in message 1, the negotiation fails.
   • If a KeyExchangeRule is found and is consistent with the proposal accepted in message 1, it is considered final, and the negotiation proceeds.

Main mode scenario 3
Certificate Authority (CA) certificates are associated with the identities of remote ISAKMP servers. When RSA signature mode authentication is being performed, the ISAKMP responder might send one or more certificate requests to the ISAKMP initiator to guide the initiator in selecting a certificate signed by an acceptable CA. Unfortunately, during a Main mode exchange the responding ISAKMP server must send a certificate request prior to learning the identity of the initiating ISAKMP server.

The z/OS IKE daemon handles this limitation as follows:
1. A key proposal is selected as described in Scenario 1.
2. If the selected key proposal indicates RSA signature mode authentication, then the IKE daemon includes one or more certificate requests in message 4.
   • If a tentative KeyExchangeRule is in effect and the KeyExchangeRule’s RemoteSecurityEndpoint includes one or more CaLabels, a certificate request corresponding to each CaLabel is included in message 4.
   • If the RemoteSecurityEndpoint does not include a CaLabel, a certificate request corresponding to each SupportedCertAuth is included in message 4.
   • If there are no applicable CaLabels or SupportedCertAuth statements configured, an empty certificate request is included in message 4, indicating that the initiator can use a certificate signed by any CA.

Commit-bit support in the IKE daemon
During a phase 2 negotiation, the IKE protocol supports the use of the commit-bit of the ISAKMP message header. The IKE daemon uses commit-bit support as defined in the IKE draft dated May 1999. This draft was written after RCF 2409.

No special configuration is required to take advantage of this support. When acting as a responder of a phase 2 negotiation, the IKE daemon always uses
commit-bit logic. When acting as an initiator of a phase 2 negotiation, the IKE daemon always honors the commit-bit preference of the responder.

The major advantage of commit-bit processing is increased interoperability and the elimination of a potential window where IP packets could be dropped during the process of negotiating a new security association. For more information about the specifics of commit-bit processing, see "Quick mode with commit bit" on page 940.
Appendix D. IBM Health Checker for z/OS

IBM Health Checker for z/OS is a z/OS component that installations can use to gather information about their system environment and system parameters to help identify potential configuration problems before they impact availability or cause outages. Individual products, z/OS components, or ISV software can provide checks that take advantage of the IBM Health Checker for z/OS framework.

z/OS Communications Server TCP/IP provides the following checks:

ZOSMIGV1R11_CS_DNSBIND9

Checks whether BIND9 DNS servers are in use on the system. By default this check is inactive. This default can be overridden on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command. If an IBM Health Checker for z/OS exception message is generated then a migration action needs to be put in place.

CSTCP_SYSTCP_PCTRACE_TCPPIPstackname

Checks whether TCP/IP Event Trace (SYSTCP) is active with options other than the default options (MINIMUM, INIT, OPCMDS, or OPMGS). By default, this check will be performed once at stack initialization and then will be repeated once every 24 hours. This default can be overridden on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command. The check name is suffixed by TCPIPstackname, which is the job name of each TCP stack that is started, in order to define a separate check for each stack.

CSTCP_TCPMAXRCVBUFRSIZE_TCPPIPstackname

Checks whether the configured TCP maximum receive buffer size is sufficient to provide optimal support to the z/OS Communications Server FTP Server. By default, this check is performed once at stack initialization and whenever a VARY TCPIP,OBEYFILE command changes the TCPMAXRCVBUFRSIZE parameter. By default, it checks that TCPMAXRCVBUFRSIZE is at least 180K. These defaults can be overridden on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command. The check name is suffixed by TCPIPstackname, which is the job name of each TCP stack that is started, in order to define a separate check for each stack.

CSTCP_SYSPLEXMON_RECOV_TCPPIPstackname

Checks whether the IPCONFIG DYNAMICXCF or IPCONFIG6 DYNAMICXCF parameters have been specified and the GLOBALCONFIG SYSPLEXMONITOR RECOVERY parameter has been specified. This check produces an exception message if the IPCONFIG DYNAMICXCF or IPCONFIG6 DYNAMICXCF parameters were specified, but the GLOBALCONFIG SYSPLEXMONITOR NORECOVERY parameter is in effect. By default, this check is performed once at stack initialization. This default can be overridden on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command. The check name is suffixed by TCPIPstackname, which is the job name of each TCP stack that is started, in order to define a separate check for each stack.
CSTCP_CINET_PORTNG_RSV_ TCPIPstackname

Checks whether the port range specified by INADDRANYPORT and INADDRANYCOUNT in the BPXPRMxx parmlib member is reserved for OMVS on this stack, when operating in a CINET environment. A port range is reserved on a TCP/IP stack using the PORTRANGE TCP/IP profile statement. By default, this check is performed once at stack initialization. This default can be overridden on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command. The check name is suffixed by TCPIPstackname, which is the job name of each TCP/IP stack that is started, in order to define a separate check for each stack.

ZOSMIGV1R11_CS_RFC4301

Checks whether IPSec filter rules that are not compliant with RFC4301 are in use on the system. By default this check is inactive. This default can be overridden on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command. If an IBM Health Checker for z/OS exception message is generated then a migration action needs to be put in place before migrating to a release higher than z/OS V1R11.

For more information about IBM Health Checker for z/OS, refer to IBM Health Checker for z/OS: User’s Guide.
Appendix E. Related protocol specifications

This appendix lists the related protocol specifications (RFCs) for TCP/IP. The Internet Protocol suite is still evolving through requests for comments (RFC). New protocols are being designed and implemented by researchers and are brought to the attention of the Internet community in the form of RFCs. Some of these protocols are so useful that they become recommended protocols. That is, all future implementations for TCP/IP are recommended to implement these particular functions or protocols. These become the de facto standards, on which the TCP/IP protocol suite is built.

You can request RFCs through electronic mail, from the automated Network Information Center (NIC) mail server, by sending a message to service@nic.ddn.mil with a subject line of RFC nnnn for text versions or a subject line of RFC nnnn.PS for PostScript versions. To request a copy of the RFC index, send a message with a subject line of RFC INDEX.

For more information, contact nic@nic.ddn.mil or at:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

Hard copies of all RFCs are available from the NIC, either individually or by subscription. Online copies are available at the following Web address:

http://www.rfc-editor.org/rfc.html

See "Internet drafts” on page 968 for draft RFCs implemented in this and previous Communications Server releases.

Many features of TCP/IP Services are based on the following RFCs:

<table>
<thead>
<tr>
<th>RFC</th>
<th>Title and Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 652</td>
<td>Telnet output carriage-return disposition option D. Crocker</td>
</tr>
<tr>
<td>RFC 653</td>
<td>Telnet output horizontal tabstops option D. Crocker</td>
</tr>
<tr>
<td>RFC 654</td>
<td>Telnet output horizontal tab disposition option D. Crocker</td>
</tr>
<tr>
<td>RFC 655</td>
<td>Telnet output formfeed disposition option D. Crocker</td>
</tr>
<tr>
<td>RFC 657</td>
<td>Telnet output vertical tab disposition option D. Crocker</td>
</tr>
<tr>
<td>RFC 658</td>
<td>Telnet output linefeed disposition D. Crocker</td>
</tr>
<tr>
<td>RFC 698</td>
<td>Telnet extended ASCII option T. Mock</td>
</tr>
<tr>
<td>RFC 726</td>
<td>Remote Controlled Transmission and Echoing Telnet option J. Postel, D. Crocker</td>
</tr>
<tr>
<td>RFC 727</td>
<td>Telnet logout option M.R. Crispin</td>
</tr>
<tr>
<td>RFC 732</td>
<td>Telnet Data Entry Terminal option J.D. Day</td>
</tr>
<tr>
<td>RFC 733</td>
<td>Standard for the format of ARPA network text messages D. Crocker, J. Vital, K.T. Pogran, D.A. Henderson</td>
</tr>
</tbody>
</table>
RFC 734  SLIPDU Protocol M.R. Crispin
RFC 735  Revised Telnet byte macro option D. Crocker, R.H. Gumpertz
RFC 736  Telnet SLIPDU option M.R. Crispin
RFC 749  Telnet SLIPDU—Output option B. Greenberg
RFC 765  File Transfer Protocol specification J. Postel
RFC 768  User Datagram Protocol J. Postel
RFC 779  Telnet send-location option E. Killian
RFC 783  TFTP Protocol (revision 2) K.R. Sollins
RFC 791  Internet Protocol J. Postel
RFC 792  Internet Control Message Protocol J. Postel
RFC 793  Transmission Control Protocol J. Postel
RFC 820  Assigned numbers J. Postel
RFC 821  Simple Mail Transfer Protocol J. Postel
RFC 822  Standard for the format of ARPA Internet text messages D. Crocker
RFC 823  DARPA Internet gateway R. Hinden, A. Sheltzer
RFC 826  Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48-bit Ethernet address for transmission on Ethernet hardware D. Plummer
RFC 855  Telnet Option Specification J. Postel, J. Reynolds
RFC 856  Telnet Binary Transmission J. Postel, J. Reynolds
RFC 857  Telnet Echo Option J. Postel, J. Reynolds
RFC 858  Telnet Suppress Go Ahead Option J. Postel, J. Reynolds
RFC 859  Telnet Status Option J. Postel, J. Reynolds
RFC 860  Telnet Timing Mark Option J. Postel, J. Reynolds
RFC 861  Telnet Extended Options: List Option J. Postel, J. Reynolds
RFC 862  Echo Protocol J. Postel
RFC 863  Discard Protocol J. Postel
RFC 864  Character Generator Protocol J. Postel
RFC 865  Quote of the Day Protocol J. Postel
RFC 868  Time Protocol J. Postel, K. Harrenstien
RFC 877  Standard for the transmission of IP datagrams over public data networks J.T. Korb
RFC 883  Domain names: Implementation specification P.V. Mockapetris
RFC 884  Telnet terminal type option M. Solomon, E. Wimmers
RFC 885  Telnet end of record option J. Postel
RFC 894  Standard for the transmission of IP datagrams over Ethernet networks C. Hornig
RFC 896  Congestion control in IP/TCP internetworks J. Nagle
Appendix E. Related protocol specifications

RFC 955

Appendix E. Related protocol specifications
<table>
<thead>
<tr>
<th>RFC</th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1044</td>
<td>Internet Protocol on Network System’s HYPERchannel: Protocol specification</td>
<td>K. Hardwick, J. Lekashman</td>
</tr>
<tr>
<td>1053</td>
<td>Telnet X.3 PAD option</td>
<td>S. Levy, T. Jacobson</td>
</tr>
<tr>
<td>1055</td>
<td>Nonstandard for transmission of IP datagrams over serial lines: SLIP</td>
<td>J. Romkey</td>
</tr>
<tr>
<td>1058</td>
<td>Routing Information Protocol</td>
<td>C. Hedrick</td>
</tr>
<tr>
<td>1060</td>
<td>Assigned numbers</td>
<td>J. Reynolds, J. Postel</td>
</tr>
<tr>
<td>1071</td>
<td>Computing the Internet checksum</td>
<td>R.T. Braden, D.A. Borman, C. Partridge</td>
</tr>
<tr>
<td>1072</td>
<td>TCP extensions for long-delay paths</td>
<td>V. Jacobson, R.T. Braden</td>
</tr>
<tr>
<td>1073</td>
<td>Telnet window size option</td>
<td>D. Waitzman</td>
</tr>
<tr>
<td>1079</td>
<td>Telnet terminal speed option</td>
<td>C. Hedrick</td>
</tr>
<tr>
<td>1085</td>
<td>ISO presentation services on top of TCP/IP based internets</td>
<td>M.T. Rose</td>
</tr>
<tr>
<td>1091</td>
<td>Telnet terminal-type option</td>
<td>J. VanBokkelen</td>
</tr>
<tr>
<td>1094</td>
<td>NFS: Network File System Protocol specification</td>
<td>Sun Microsystems</td>
</tr>
<tr>
<td>1096</td>
<td>Telnet X display location option</td>
<td>G. Marcy</td>
</tr>
<tr>
<td>1101</td>
<td>DNS encoding of network names and other types</td>
<td>P. Mockapetris</td>
</tr>
<tr>
<td>1112</td>
<td>Host extensions for IP multicasting</td>
<td>S.E. Deering</td>
</tr>
<tr>
<td>1113</td>
<td>Privacy enhancement for Internet electronic mail: Part I — message encipherment and authentication procedures</td>
<td>J. Linn</td>
</tr>
<tr>
<td>1118</td>
<td>Hitchhiker’s Guide to the Internet</td>
<td>E. Krol</td>
</tr>
<tr>
<td>1122</td>
<td>Requirements for Internet Hosts—Communication Layers</td>
<td>R. Braden, Ed.</td>
</tr>
<tr>
<td>1123</td>
<td>Requirements for Internet Hosts—Application and Support</td>
<td>R. Braden, Ed.</td>
</tr>
<tr>
<td>1146</td>
<td>TCP alternate checksum options</td>
<td>J. Zweig, C. Partridge</td>
</tr>
<tr>
<td>1155</td>
<td>Structure and identification of management information for TCP/IP-based internets</td>
<td>M. Rose, K. McCloghrie</td>
</tr>
<tr>
<td>1156</td>
<td>Management Information Base for network management of TCP/IP-based internets</td>
<td>K. McCloghrie, M. Rose</td>
</tr>
<tr>
<td>1158</td>
<td>Management Information Base for network management of TCP/IP-based internets: MIB-II</td>
<td>M. Rose</td>
</tr>
<tr>
<td>1166</td>
<td>Internet numbers</td>
<td>S. Kirkpatrick, M.K. Stahl, M. Recker</td>
</tr>
<tr>
<td>1179</td>
<td>Line printer daemon protocol</td>
<td>L. McLaughlin</td>
</tr>
<tr>
<td>1180</td>
<td>TCP/IP tutorial</td>
<td>T. Socolofsky, C. Kale</td>
</tr>
<tr>
<td>RFC</td>
<td>Title</td>
<td>Authors</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>RFC 1184</td>
<td>Telnet Linemode Option</td>
<td>D. Borman</td>
</tr>
<tr>
<td>RFC 1186</td>
<td>MD4 Message Digest Algorithm</td>
<td>R.L. Rivest</td>
</tr>
<tr>
<td>RFC 1187</td>
<td>Bulk Table Retrieval with the SNMP</td>
<td>M. Rose, K. McCloghrie, J. Davin</td>
</tr>
<tr>
<td>RFC 1188</td>
<td>Proposed Standard for the Transmission of IP Datagrams over FDDI Networks</td>
<td>D. Katz</td>
</tr>
<tr>
<td>RFC 1190</td>
<td>Experimental Internet Stream Protocol: Version 2 (ST-II)</td>
<td>C. Topolcic</td>
</tr>
<tr>
<td>RFC 1191</td>
<td>Path MTU discovery</td>
<td>J. Mogul, S. Deering</td>
</tr>
<tr>
<td>RFC 1198</td>
<td>FYI on the X window system</td>
<td>R. Scheifler</td>
</tr>
<tr>
<td>RFC 1207</td>
<td>FYI on Questions and Answers: Answers to commonly asked &quot;experienced Internet user&quot; questions</td>
<td>G. Malkin, A. Marine, J. Reynolds</td>
</tr>
<tr>
<td>RFC 1208</td>
<td>Glossary of networking terms</td>
<td>O. Jacobsen, D. Lynch</td>
</tr>
<tr>
<td>RFC 1215</td>
<td>Convention for defining traps for use with the SNMP MIB</td>
<td>M. Rose</td>
</tr>
<tr>
<td>RFC 1227</td>
<td>SNMP MUX protocol and MIB</td>
<td>M.T. Rose</td>
</tr>
<tr>
<td>RFC 1228</td>
<td>SNMP-DPI: Simple Network Management Protocol Distributed Program Interface</td>
<td>G. Carpenter, B. Wijnen</td>
</tr>
<tr>
<td>RFC 1229</td>
<td>Extensions to the generic-interface MIB</td>
<td>K. McCloghrie</td>
</tr>
<tr>
<td>RFC 1230</td>
<td>IEEE 802.4 Token Bus MIB</td>
<td>K. McCloghrie, R. Fox</td>
</tr>
<tr>
<td>RFC 1231</td>
<td>IEEE 802.5 Token Ring MIB</td>
<td>K. McCloghrie, R. Fox, E. Decker</td>
</tr>
<tr>
<td>RFC 1236</td>
<td>IP to X.121 address mapping for DDN</td>
<td>L. Morales, P. Hasse</td>
</tr>
<tr>
<td>RFC 1256</td>
<td>ICMP Router Discovery Messages</td>
<td>S. Deering, Ed.</td>
</tr>
<tr>
<td>RFC 1267</td>
<td>Border Gateway Protocol 3 (BGP-3)</td>
<td>K. Lougheed, Y. Rekhter</td>
</tr>
<tr>
<td>RFC 1268</td>
<td>Application of the Border Gateway Protocol in the Internet</td>
<td>Y. Rekhter, P. Gross</td>
</tr>
<tr>
<td>RFC 1269</td>
<td>Definitions of Managed Objects for the Border Gateway Protocol: Version 3</td>
<td>S. Willis, J. Burruss</td>
</tr>
<tr>
<td>RFC 1270</td>
<td>SNMP Communications Services</td>
<td>F. Kastenholz, ed.</td>
</tr>
<tr>
<td>RFC 1285</td>
<td>FDDI Management Information Base</td>
<td>J. Case</td>
</tr>
<tr>
<td>RFC 1315</td>
<td>Management Information Base for Frame Relay DTEs</td>
<td>C. Brown, F. Baker, C. Carvalho</td>
</tr>
<tr>
<td>RFC 1321</td>
<td>The MD5 Message-Digest Algorithm</td>
<td>R. Rivest</td>
</tr>
<tr>
<td>RFC 1323</td>
<td>TCP Extensions for High Performance</td>
<td>V. Jacobson, R. Braden, D. Borman</td>
</tr>
<tr>
<td>RFC 1325</td>
<td>FYI on Questions and Answers: Answers to Commonly Asked &quot;New Internet User&quot; Questions</td>
<td>G. Malkin, A. Marine</td>
</tr>
</tbody>
</table>
| RFC 1327 | Mapping between X.400 (1988)/ISO 10021 and RFC 822                                           | S. Hardcastle-Kille
<table>
<thead>
<tr>
<th>RFC</th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1340</td>
<td>Assigned Numbers</td>
<td>J. Reynolds, J. Postel</td>
</tr>
<tr>
<td>1344</td>
<td>Implications of MIME for Internet Mail Gateways</td>
<td>N. Bornstein</td>
</tr>
<tr>
<td>1349</td>
<td>Type of Service in the Internet Protocol Suite</td>
<td>P. Almquist</td>
</tr>
<tr>
<td>1350</td>
<td>The TFTP Protocol (Revision 2)</td>
<td>K.R. Sollins</td>
</tr>
<tr>
<td>1351</td>
<td>SNMP Administrative Model</td>
<td>J. Davin, J. Galvin, K. McCloghrie</td>
</tr>
<tr>
<td>1352</td>
<td>SNMP Security Protocols</td>
<td>J. Galvin, K. McCloghrie, J. Davin</td>
</tr>
<tr>
<td>1353</td>
<td>Definitions of Managed Objects for Administration of SNMP Parties</td>
<td>K. McCloghrie, J. Davin, J. Galvin</td>
</tr>
<tr>
<td>1354</td>
<td>IP Forwarding Table MIB</td>
<td>F. Baker</td>
</tr>
<tr>
<td>1356</td>
<td>Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode</td>
<td>A. Malis, D. Robinson, R. Ullmann</td>
</tr>
<tr>
<td>1358</td>
<td>Charter of the Internet Architecture Board (IAB)</td>
<td>L. Chapin</td>
</tr>
<tr>
<td>1363</td>
<td>A Proposed Flow Specification</td>
<td>C. Partridge</td>
</tr>
<tr>
<td>1368</td>
<td>Definition of Managed Objects for IEEE 802.3 Repeater Devices</td>
<td>D. McMaster, K. McCloghrie</td>
</tr>
<tr>
<td>1372</td>
<td>Telnet Remote Flow Control Option</td>
<td>C. L. Hedrick, D. Borman</td>
</tr>
<tr>
<td>1374</td>
<td>IP and ARP on HIPPI</td>
<td>J. Renwick, A. Nicholson</td>
</tr>
<tr>
<td>1381</td>
<td>SNMP MIB Extension for X.25 LAPB</td>
<td>D. Throop, F. Baker</td>
</tr>
<tr>
<td>1382</td>
<td>SNMP MIB Extension for the X.25 Packet Layer</td>
<td>D. Throop</td>
</tr>
<tr>
<td>1387</td>
<td>RIP Version 2 Protocol Analysis</td>
<td>G. Malkin</td>
</tr>
<tr>
<td>1388</td>
<td>RIP Version 2 Carrying Additional Information</td>
<td>G. Malkin</td>
</tr>
<tr>
<td>1389</td>
<td>RIP Version 2 MIB Extensions</td>
<td>G. Malkin, F. Baker</td>
</tr>
<tr>
<td>1390</td>
<td>Transmission of IP and ARP over FDDI Networks</td>
<td>D. Katz</td>
</tr>
<tr>
<td>1393</td>
<td>Traceroute Using an IP Option</td>
<td>G. Malkin</td>
</tr>
<tr>
<td>1398</td>
<td>Definitions of Managed Objects for the Ethernet-Like Interface Types</td>
<td>F. Kastenholz</td>
</tr>
<tr>
<td>1408</td>
<td>Telnet Environment Option</td>
<td>D. Borman, Ed.</td>
</tr>
<tr>
<td>1413</td>
<td>Identification Protocol</td>
<td>M. St. Johns</td>
</tr>
<tr>
<td>1416</td>
<td>Telnet Authentication Option</td>
<td>D. Borman, ed.</td>
</tr>
<tr>
<td>1420</td>
<td>SNMP over IPX</td>
<td>S. Bostock</td>
</tr>
<tr>
<td>1428</td>
<td>Transition of Internet Mail from Just-Send-8 to 8bit-SMTP/MIME</td>
<td>G. Vaudreuil</td>
</tr>
<tr>
<td>1445</td>
<td>Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)</td>
<td>J. Galvin, K. McCloghrie</td>
</tr>
<tr>
<td>1447</td>
<td>Party MIB for version 2 of the Simple Network Management Protocol</td>
<td>K. McCloghrie, J. Galvin</td>
</tr>
<tr>
<td>RFC 1464</td>
<td>Using the Domain Name System to Store Arbitrary String Attributes</td>
<td>R. Rosenbaum</td>
</tr>
<tr>
<td>RFC 1469</td>
<td>IP Multicast over Token-Ring Local Area Networks</td>
<td>T. Pusateri</td>
</tr>
<tr>
<td>RFC 1483</td>
<td>Multiprotocol Encapsulation over ATM Adaptation Layer 5</td>
<td>Juha Heinanen</td>
</tr>
<tr>
<td>RFC 1514</td>
<td>Host Resources MIB</td>
<td>P. Grillo, S. Waldbusser</td>
</tr>
<tr>
<td>RFC 1516</td>
<td>Definitions of Managed Objects for IEEE 802.3 Repeater Devices</td>
<td>D. McMaster, K. McCloghrie</td>
</tr>
<tr>
<td>RFC 1521</td>
<td>MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies</td>
<td>N. Borenstein, N. Freed</td>
</tr>
<tr>
<td>RFC 1535</td>
<td>A Security Problem and Proposed Correction With Widely Deployed DNS Software</td>
<td>E. Gavron</td>
</tr>
<tr>
<td>RFC 1536</td>
<td>Common DNS Implementation Errors and Suggested Fixes</td>
<td>A. Kumar, J. Postel, C. Neuman, P. Danzig, S. Miller</td>
</tr>
<tr>
<td>RFC 1537</td>
<td>Common DNS Data File Configuration Errors</td>
<td>P. Beertema</td>
</tr>
<tr>
<td>RFC 1540</td>
<td>Internet Official Protocol Standards</td>
<td>J. Postel</td>
</tr>
<tr>
<td>RFC 1571</td>
<td>Telnet Environment Option Interoperability Issues</td>
<td>D. Borman</td>
</tr>
<tr>
<td>RFC 1572</td>
<td>Telnet Environment Option</td>
<td>S. Alexander</td>
</tr>
<tr>
<td>RFC 1573</td>
<td>Evolution of the Interfaces Group of MIB-II</td>
<td>K. McCloghrie, F. Kastenholz</td>
</tr>
<tr>
<td>RFC 1577</td>
<td>Classical IP and ARP over ATM</td>
<td>M. Laubach</td>
</tr>
<tr>
<td>RFC 1583</td>
<td>OSPF Version 2</td>
<td>J. Moy</td>
</tr>
<tr>
<td>RFC 1591</td>
<td>Domain Name System Structure and Delegation</td>
<td>J. Postel</td>
</tr>
<tr>
<td>RFC 1594</td>
<td>FYI on Questions and Answers—Answers to Commonly Asked &quot;New Internet User&quot; Questions</td>
<td>A. Marine, J. Reynolds, G. Malkin</td>
</tr>
<tr>
<td>RFC 1644</td>
<td>T/TCP — TCP Extensions for Transactions Functional Specification</td>
<td>R. Braden</td>
</tr>
<tr>
<td>RFC 1646</td>
<td>TN3270 Extensions for LUname and Printer Selection</td>
<td>C. Graves, T. Butts, M. Angel</td>
</tr>
<tr>
<td>RFC 1647</td>
<td>TN3270 Enhancements</td>
<td>B. Kelly</td>
</tr>
<tr>
<td>RFC 1652</td>
<td>SMTP Service Extension for 8bit-MIMEtransport</td>
<td>J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker</td>
</tr>
<tr>
<td>RFC 1664</td>
<td>Using the Internet DNS to Distribute RFC1327 Mail Address Mapping Tables</td>
<td>C. Allochio, A. Bonito, B. Cole, S. Giordano, R. Hagens</td>
</tr>
<tr>
<td>RFC 1693</td>
<td>An Extension to TCP: Partial Order Service</td>
<td>T. Connolly, P. Amer, P. Conrad</td>
</tr>
<tr>
<td>RFC 1695</td>
<td>Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2</td>
<td>M. Ahmed, K. Tesink</td>
</tr>
<tr>
<td>RFC</td>
<td>Title</td>
<td>Authors</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RFC 1701</td>
<td>Generic Routing Encapsulation (GRE) S. Hanks, T. Li, D. Farinacci, P. Traina</td>
<td></td>
</tr>
<tr>
<td>RFC 1702</td>
<td>Generic Routing Encapsulation over IPv4 networks S. Hanks, T. Li, D. Farinacci, P. Traina</td>
<td></td>
</tr>
<tr>
<td>RFC 1706</td>
<td>DNS NSAP Resource Records B. Manning, R. Colella</td>
<td></td>
</tr>
<tr>
<td>RFC 1712</td>
<td>DNS Encoding of Geographical Location C. Farrell, M. Schulze, S. Pleitner D. Baldoni</td>
<td></td>
</tr>
<tr>
<td>RFC 1713</td>
<td>Tools for DNS debugging A. Romao</td>
<td></td>
</tr>
<tr>
<td>RFC 1723</td>
<td>RIP Version 2—Carrying Additional Information G. Malkin</td>
<td></td>
</tr>
<tr>
<td>RFC 1752</td>
<td>The Recommendation for the IP Next Generation Protocol S. Bradner, A. Mankin</td>
<td></td>
</tr>
<tr>
<td>RFC 1766</td>
<td>Tags for the Identification of Languages H. Alvestrand</td>
<td></td>
</tr>
<tr>
<td>RFC 1771</td>
<td>A Border Gateway Protocol 4 (BGP-4) Y. Rekhter, T. Li</td>
<td></td>
</tr>
<tr>
<td>RFC 1794</td>
<td>DNS Support for Load Balancing T. Brisco</td>
<td></td>
</tr>
<tr>
<td>RFC 1826</td>
<td>IP Authentication Header R. Atkinson</td>
<td></td>
</tr>
<tr>
<td>RFC 1828</td>
<td>IP Authentication using Keyed MD5 P. Metzger, W. Simpson</td>
<td></td>
</tr>
<tr>
<td>RFC 1829</td>
<td>The ESP DES-CBC Transform P. Karn, P. Metzger, W. Simpson</td>
<td></td>
</tr>
<tr>
<td>RFC 1830</td>
<td>SMTP Service Extensions for Transmission of Large and Binary MIME Messages G. Vaudreuil</td>
<td></td>
</tr>
<tr>
<td>RFC 1832</td>
<td>XDR: External Data Representation Standard R. Srinivasan</td>
<td></td>
</tr>
<tr>
<td>RFC 1833</td>
<td>Binding Protocols for ONC RPC Version 2 R. Srinivasan</td>
<td></td>
</tr>
<tr>
<td>RFC 1850</td>
<td>OSPF Version 2 Management Information Base F. Baker, R. Coltun</td>
<td></td>
</tr>
<tr>
<td>RFC 1854</td>
<td>SMTP Service Extension for Command Pipelining N. Freed</td>
<td></td>
</tr>
<tr>
<td>RFC 1869</td>
<td>SMTP Service Extensions J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker</td>
<td></td>
</tr>
<tr>
<td>RFC 1870</td>
<td>SMTP Service Extension for Message Size Declaration J. Klensin, N. Freed, K. Moore</td>
<td></td>
</tr>
<tr>
<td>RFC 1876</td>
<td>A Means for Expressing Location Information in the Domain Name System C. Davis, P. Vixie, T. Goodwin, I. Dickinson</td>
<td></td>
</tr>
<tr>
<td>RFC 1883</td>
<td>Internet Protocol, Version 6 (IPv6) Specification S. Deering, R. Hinden</td>
<td></td>
</tr>
<tr>
<td>RFC 1886</td>
<td>DNS Extensions to support IP version 6 S. Thomson, C. Huitema</td>
<td></td>
</tr>
<tr>
<td>RFC 1888</td>
<td>OSI NSAPs and IPv6 J. Bound, B. Carpenter, D. Harrington, J. Houldsworth, A. Lloyd</td>
<td></td>
</tr>
<tr>
<td>RFC 1891</td>
<td>SMTP Service Extension for Delivery Status Notifications K. Moore</td>
<td></td>
</tr>
<tr>
<td>RFC 1892</td>
<td>The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages G. Vaudreuil</td>
<td></td>
</tr>
</tbody>
</table>
Appendix E. Related protocol specifications
<table>
<thead>
<tr>
<th>RFC</th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018</td>
<td>TCP Selective Acknowledgement Options</td>
<td>M. Mathis, J. Mahdavi, S. Floyd, A. Romanow</td>
</tr>
<tr>
<td>2026</td>
<td>The Internet Standards Process — Revision 3</td>
<td>S. Bradner</td>
</tr>
<tr>
<td>2030</td>
<td>Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI</td>
<td>D. Mills</td>
</tr>
<tr>
<td>2033</td>
<td>Local Mail Transfer Protocol</td>
<td>J. Myers</td>
</tr>
<tr>
<td>2034</td>
<td>SMTP Service Extension for Returning Enhanced Error Codes</td>
<td>N. Freed</td>
</tr>
<tr>
<td>2045</td>
<td>Multipurpose Internet Mail Extensions (MIME) Part One: Format of</td>
<td>N. Freed, N. Borenstein</td>
</tr>
<tr>
<td></td>
<td>Internet Message Bodies</td>
<td></td>
</tr>
<tr>
<td>2052</td>
<td>A DNS RR for specifying the location of services (DNS SRV)</td>
<td>A. Gulbrandsen, P. Vixie</td>
</tr>
<tr>
<td>2065</td>
<td>Domain Name System Security Extensions</td>
<td>D. Eastlake 3rd, C. Kaufman</td>
</tr>
<tr>
<td>2066</td>
<td>TELNET CHARSET Option</td>
<td>R. Gellens</td>
</tr>
<tr>
<td>2080</td>
<td>RIPng for IPv6</td>
<td>G. Malkin, R. Minnear</td>
</tr>
<tr>
<td>2096</td>
<td>IP Forwarding Table MIB</td>
<td>F. Baker</td>
</tr>
<tr>
<td>2104</td>
<td>HMAC: Keyed-Hashing for Message Authentication</td>
<td>H. Krawczyk, M. Bellare, R. Canetti</td>
</tr>
<tr>
<td>2119</td>
<td>Keywords for use in RFCs to Indicate Requirement Levels</td>
<td>S. Bradner</td>
</tr>
<tr>
<td>2133</td>
<td>Basic Socket Interface Extensions for IPv6</td>
<td>R. Gilligan, S. Thomson, J. Bound, W. Stevens</td>
</tr>
<tr>
<td>2136</td>
<td>Dynamic Updates in the Domain Name System (DNS UPDATE)</td>
<td>P. Vixie, Ed., S. Thomson, Y. Rekhter, J. Bound</td>
</tr>
<tr>
<td>2137</td>
<td>Secure Domain Name System Dynamic Update</td>
<td>D. Eastlake 3rd</td>
</tr>
<tr>
<td>2163</td>
<td>Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM)</td>
<td>C. Allocchio</td>
</tr>
<tr>
<td>2168</td>
<td>Resolution of Uniform Resource Identifiers using the Domain Name System</td>
<td>R. Daniel, M. Mealling</td>
</tr>
<tr>
<td>2178</td>
<td>OSPF Version 2</td>
<td>J. Moy</td>
</tr>
<tr>
<td>2181</td>
<td>Clarifications to the DNS Specification</td>
<td>R. Elz, R. Bush</td>
</tr>
<tr>
<td>2210</td>
<td>The Use of RSVP with IETF Integrated Services</td>
<td>J. Wroclawski</td>
</tr>
<tr>
<td>2211</td>
<td>Specification of the Controlled-Load Network Element Service</td>
<td>J. Wroclawski</td>
</tr>
<tr>
<td>2212</td>
<td>Specification of Guaranteed Quality of Service</td>
<td>S. Shenker, C. Partridge, R. Guerin</td>
</tr>
<tr>
<td>2215</td>
<td>General Characterization Parameters for Integrated Service Network Elements</td>
<td>S. Shenker, J. Wroclawski</td>
</tr>
<tr>
<td>2217</td>
<td>Telnet Com Port Control Option</td>
<td>G. Clarke</td>
</tr>
<tr>
<td>RFC 2219</td>
<td>Use of DNS Aliases for Network Services M. Hamilton, R. Wright</td>
<td></td>
</tr>
<tr>
<td>RFC 2228</td>
<td>FTP Security Extensions M. Horowitz, S. Lunt</td>
<td></td>
</tr>
<tr>
<td>RFC 2230</td>
<td>Key Exchange Delegation Record for the DNS R. Atkinson</td>
<td></td>
</tr>
<tr>
<td>RFC 2233</td>
<td>The Interfaces Group MIB using SMIv2 K. McCloghrie, F. Kastenholz</td>
<td></td>
</tr>
<tr>
<td>RFC 2240</td>
<td>A Legal Basis for Domain Name Allocation O. Vaughn</td>
<td></td>
</tr>
<tr>
<td>RFC 2246</td>
<td>The TLS Protocol Version 1.0 T. Dierks, C. Allen</td>
<td></td>
</tr>
<tr>
<td>RFC 2254</td>
<td>The String Representation of LDAP Search Filters T. Howes</td>
<td></td>
</tr>
<tr>
<td>RFC 2261</td>
<td>An Architecture for Describing SNMP Management Frameworks D. Harrington, R. Presuhn, B. Wijnen</td>
<td></td>
</tr>
<tr>
<td>RFC 2271</td>
<td>An Architecture for Describing SNMP Management Frameworks D. Harrington, R. Presuhn, B. Wijnen</td>
<td></td>
</tr>
<tr>
<td>RFC 2273</td>
<td>SNMPv3 Applications D. Levi, P. Meyer, B. Stewartz</td>
<td></td>
</tr>
<tr>
<td>RFC 2275</td>
<td>View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP) B. Wijnen, R. Presuhn, K. McCloghrie</td>
<td></td>
</tr>
<tr>
<td>RFC 2279</td>
<td>UTF-8, a transformation format of ISO 10646 F. Yergeau</td>
<td></td>
</tr>
<tr>
<td>RFC 2292</td>
<td>Advanced Sockets API for IPv6 W. Stevens, M. Thomas</td>
<td></td>
</tr>
<tr>
<td>RFC 2308</td>
<td>Negative Caching of DNS Queries (DNS NCACHE) M. Andrews</td>
<td></td>
</tr>
<tr>
<td>RFC 2317</td>
<td>Classless IN-ADDR.ARPA delegation H. Eidnes, G. de Groot, P. Vixie</td>
<td></td>
</tr>
<tr>
<td>RFC 2320</td>
<td>Definitions of Managed Objects for Classical IP and ARP Over ATM Using SMIv2 (IPOA-MIB) M. Greene, J. Luciani, K. White, T. Kuo</td>
<td></td>
</tr>
<tr>
<td>RFC 2328</td>
<td>OSPF Version 2 J. Moy</td>
<td></td>
</tr>
<tr>
<td>RFC 2345</td>
<td>Domain Names and Company Name Retrieval J. Klensin, T. Wolf, G. Oglesby</td>
<td></td>
</tr>
<tr>
<td>RFC 2352</td>
<td>A Convention for Using Legal Names as Domain Names O. Vaughn</td>
<td></td>
</tr>
<tr>
<td>RFC 2355</td>
<td>TN3270 Enhancements B. Kelly</td>
<td></td>
</tr>
<tr>
<td>RFC 2358</td>
<td>Definitions of Managed Objects for the Ethernet-like Interface Types J. Flick, J. Johnson</td>
<td></td>
</tr>
<tr>
<td>RFC 2373</td>
<td>IP Version 6 Addressing Architecture R. Hinden, S. Deering</td>
<td></td>
</tr>
<tr>
<td>RFC 2374</td>
<td>An IPv6 Aggregatable Global Unicast Address Format R. Hinden, M. O’Dell, S. Deering</td>
<td></td>
</tr>
<tr>
<td>RFC 2375</td>
<td>IPv6 Multicast Address Assignments R. Hinden, S. Deering</td>
<td></td>
</tr>
<tr>
<td>RFC 2385</td>
<td>Protection of BGP Sessions via the TCP MD5 Signature Option A. Hefferman</td>
<td></td>
</tr>
</tbody>
</table>
RFC 2389   Feature negotiation mechanism for the File Transfer Protocol P. Hethmon, R. Elz
RFC 2401   Security Architecture for Internet Protocol S. Kent, R. Atkinson
RFC 2402   IP Authentication Header S. Kent, R. Atkinson
RFC 2403   The Use of HMAC-MD5–96 within ESP and AH C. Madson, R. Glenn
RFC 2404   The Use of HMAC-SHA–1–96 within ESP and AH C. Madson, R. Glenn
RFC 2405   The ESP DES-CBC Cipher Algorithm With Explicit IV C. Madson, N. Doraswamy
RFC 2406   IP Encapsulating Security Payload (ESP) S. Kent, R. Atkinson
RFC 2407   The Internet IP Security Domain of Interpretation for ISAKMPD. Piper
RFC 2409   The Internet Key Exchange (IKE) D. Harkins, D. Carrel
RFC 2410   The NULL Encryption Algorithm and Its Use With IPsec R. Glenn, S. Kent,
RFC 2428   FTP Extensions for IPv6 and NATs M. Allman, S. Ostermann, C. Metz
RFC 2445   Internet Calendaring and Scheduling Core Object Specification (iCalendar) F. Dawson, D. Stenerson
RFC 2459   Internet X.509 Public Key Infrastructure Certificate and CRL Profile R. Housley, W. Ford, W. Polk, D. Solo
RFC 2460   Internet Protocol, Version 6 (IPv6) Specification S. Deering, R. Hinden
RFC 2462   IPv6 Stateless Address Autoconfiguration S. Thomson, T. Narten
RFC 2463   Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification A. Conta, S. Deering
RFC 2464   Transmission of IPv6 Packets over Ethernet Networks M. Crawford
RFC 2476   Message Submission R. Gellens, J. Klensin
RFC 2487   SMTP Service Extension for Secure SMTP over TLS P. Hoffman
RFC 2505   Anti-Spam Recommendations for SMTP MTAs G. Lindberg
RFC 2523   Phaturs: Extended Schemes and Attributes P. Karn, W. Simpson
RFC 2535   Domain Name System Security Extensions D. Eastlake 3rd
RFC 2538   Storing Certificates in the Domain Name System (DNS) D. Eastlake 3rd, O. Gudmundsson
RFC 2539   Storage of Diffie-Hellman Keys in the Domain Name System (DNS) D. Eastlake 3rd
RFC 2540   Detached Domain Name System (DNS) Information D. Eastlake 3rd
RFC 2554   SMTP Service Extension for Authentication J. Myers
Appendix E. Related protocol specifications

RFC 2570  Introduction to Version 3 of the Internet-standard Network Management Framework  J. Case, R. Mundy, D. Partain, B. Stewart
RFC 2573  SNMP Applications  D. Levi, P. Meyer, B. Stewart
RFC 2575  View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)  B. Wijnen, R. Presuhn, K. McCloghrie
RFC 2579  Textual Conventions for SMIv2  K. McCloghrie, D. Perkins, J. Schoenwaelder
RFC 2580  Conformance Statements for SMIv2  K. McCloghrie, D. Perkins, J. Schoenwaelder
RFC 2581  TCP Congestion Control  M. Allman, V. Paxson, W. Stevens
RFC 2583  Guidelines for Next Hop Client (NHC) Developers  R. Carlson, L. Winkler
RFC 2591  Definitions of Managed Objects for Scheduling Management Operations  D. Levi, J. Schoenwaelder
RFC 2625  IP and ARP over Fibre Channel  M. Rajagopal, R. Bhagwat, W. Rickard
RFC 2635  Don't SPEW A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam)  S. Hambridge, A. Lunde
RFC 2640  Internationalization of the File Transfer Protocol  B. Curtin
RFC 2665  Definitions of Managed Objects for the Ethernet-like Interface Types  J. Flick, J. Johnson
RFC 2671  Extension Mechanisms for DNS (EDNS0)  P. Vixie
RFC 2672  Non-Terminal DNS Name Redirection  M. Crawford
RFC 2675  IPv6 Jumbograms  D. Borman, S. Deering, R. Hinden
RFC 2710  Multicast Listener Discovery (MLD) for IPv6  S. Deering, W. Fenner, B. Haberman
RFC 2711  IPv6 Router Alert Option  C. Partridge, A. Jackson
RFC 2740  OSPF for IPv6  R. Coltun, D. Ferguson, J. Moy
RFC 2753  A Framework for Policy-based Admission Control  R. Yavatkar, D. Pendarakis, R. Guerin
| RFC 2782 | A DNS RR for specifying the location of services (DNS SRV) | A. Gubrandsen, P. Vixie, L. Esibov |
| RFC 2821 | Simple Mail Transfer Protocol | J. Klensin, Ed. |
| RFC 2822 | Internet Message Format | P. Resnick, Ed. |
| RFC 2840 | TELNET KERMIT OPTION | J. Altman, F. da Cruz |
| RFC 2845 | Secret Key Transaction Authentication for DNS (TSIG) | P. Vixie, O. Gudmundsson, D. Eastlake 3rd, B. Wellington |
| RFC 2851 | Textual Conventions for Internet Network Addresses | M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder |
| RFC 2852 | Deliver By SMTP Service Extension | D. Newman |
| RFC 2874 | DNS Extensions to Support IPv6 Address Aggregation and Renumbering | M. Crawford, C. Huitema |
| RFC 2915 | The Naming Authority Pointer (NAPTR) DNS Resource Record | M. Mealling, R. Daniel |
| RFC 2920 | SMTP Service Extension for Command Pipelining | N. Freed |
| RFC 2930 | Secret Key Establishment for DNS (TKEY RR) | D. Eastlake, 3rd |
| RFC 2941 | Telnet Authentication Option | T. Ts’o, ed., J. Altman |
| RFC 2942 | Telnet Authentication: Kerberos Version 5 | T. Ts’o |
| RFC 2946 | Telnet Data Encryption Option | T. Ts’o |
| RFC 2952 | Telnet Encryption: DES 64 bit Cipher Feedback | T. Ts’o |
| RFC 2953 | Telnet Encryption: DES 64 bit Output Feedback | T. Ts’o |
| RFC 2992 | Analysis of an Equal-Cost Multi-Path Algorithm | C. Hopps |
| RFC 3019 | IP Version 6 Management Information Base for The Multicast Listener Discovery Protocol | B. Haberman, R. Wozzella |
| RFC 3060 | Policy Core Information Model—Version 1 Specification | B. Moore, E. Ellesson, J. Strassner, A. Westerinen |
| RFC 3152 | Delegation of IP6.ARPA | R. Bush |
| RFC 3164 | The BSD Syslog Protocol | C. Lonvick |
| RFC 3207 | SMTP Service Extension for Secure SMTP over Transport Layer Security | P. Hoffman |
| RFC 3226 | DNSSEC and IPv6 A6 aware server/resolver message size requirements | O. Gudmundsson |
| RFC 3291 | Textual Conventions for Internet Network Addresses | M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder |
| RFC 3376 | Internet Group Management Protocol, Version 3 | B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan |
| RFC 3390 | Increasing TCP’s Initial Window | M. Allman, S. Floyd, C. Partridge |
| RFC 3410 | Introduction and Applicability Statements for Internet-Standard Management Framework | J. Case, R. Mundy, D. Partain, B. Stewart |
Appendix E. Related protocol specifications

RFC 3415  View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)  B. Wijnen, R. Presuhn, K. McCloghrie
RFC 3419  Textual Conventions for Transport Addresses  M. Daniele, J. Schoenwaelder
RFC 3484  Default Address Selection for Internet Protocol version 6 (IPv6)  R. Draves
RFC 3513  Internet Protocol Version 6 (IPv6) Addressing Architecture  R. Hinden, S. Deering
RFC 3526  More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)  T. Kivinen, M. Kojo
RFC 3542  Advanced Sockets Application Programming Interface (API) for IPv6  W. Richard Stevens, M. Thomas, E. Nordmark, T. Jinmei
RFC 3569  An Overview of Source-Specific Multicast (SSM)  S. Bhattacharyya, Ed.
RFC 3602  The AES-CBC Cipher Algorithm and Its Use with IPsec  S. Frankel, R. Glenn, S. Kelly
RFC 3629  UTF-8, a transformation format of ISO 10646  R. Kermode, C. Vicisano
RFC 3658  Delegation Signer (DS) Resource Record (RR)  O. Gudmundsson
RFC 3678  Socket Interface Extensions for Multicast Source Filters  D. Thaler, B. Fenner, B. Quinn
RFC 3715  IPsec-Network Address Translation (NAT) Compatibility Requirements  B. Aboba, W. Dixon
Internet drafts

Internet drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Other groups may also distribute working documents as Internet drafts. You can see Internet drafts at [http://www.ietf.org/ID.html](http://www.ietf.org/ID.html).

Several areas of IPv6 implementation include elements of the following Internet drafts and are subject to change during the RFC review process.
Appendix F. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to z/OS TSO/E Primer, z/OS TSO/E User’s Guide, and z/OS ISPF User’s Guide Vol I for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at www.ibm.com/systems/z/os/zos/bkserv/
Notices

IBM may not offer all of the products, services, or features discussed in this document. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
P.O. Box 12195
3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM’s suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application
programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM’s application programming interfaces.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

IBM is required to include the following statements in order to distribute portions of this document and the software described herein to which contributions have been made by The University of California. Portions herein © Copyright 1979, 1980, 1983, 1986, Regents of the University of California. Reproduced by permission. Portions herein were developed at the Electrical Engineering and Computer Sciences Department at the Berkeley campus of the University of California under the auspices of the Regents of the University of California.

Portions of this publication relating to RPC are Copyright © Sun Microsystems, Inc., 1988, 1989.

Some portions of this publication relating to X Window System** are Copyright © 1987, 1988 by Digital Equipment Corporation, Maynard, Massachusetts, and the Massachusetts Institute Of Technology, Cambridge, Massachusetts. All Rights Reserved.

Some portions of this publication relating to X Window System are Copyright © 1986, 1987, 1988 by Hewlett-Packard Corporation.

Permission to use, copy, modify, and distribute the M.I.T., Digital Equipment Corporation, and Hewlett-Packard Corporation portions of this software and its documentation for any purpose without fee is hereby granted, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the names of M.I.T., Digital, and Hewlett-Packard not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T., Digital, and Hewlett-Packard make no representation about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.


Copyright © 1988, 1993 The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement:
   This product includes software developed by the University of California, Berkeley and its contributors.

4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software program contains code, and/or derivatives or modifications of code originating from the software program "Popper." Popper is Copyright ©1989-1991 The Regents of the University of California, All Rights Reserved. Popper was created by Austin Shelton, Information Systems and Technology, University of California, Berkeley.

Permission from the Regents of the University of California to use, copy, modify, and distribute the "Popper" software contained herein for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies. HOWEVER, ADDITIONAL PERMISSIONS MAY BE NECESSARY FROM OTHER PERSONS OR ENTITIES, TO USE DERIVATIVES OR MODIFICATIONS OF POPPER.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THE POPPER SOFTWARE, OR ITS DERIVATIVES OR MODIFICATIONS, AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE POPPER SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

Copyright © 1983 The Regents of the University of California. All rights reserved.
if you modify this software you must label your software as modified software and not distribute it in such a fashion that it might be confused with the original M.I.T. software. M.I.T. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

Copyright © 1998 by the FundsXpress, INC. All rights reserved.

Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of FundsXpress not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. FundsXpress makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.


Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND INTERNET SOFTWARE CONSORTIUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL INTERNET SOFTWARE CONSORTIUM BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright © 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscape’s SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, Ihash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young’s, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be
given attribution as the author of the parts of the library used. This can be in the
form of a textual message at program startup or in documentation (online or
textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the copyright notice, this list of
   conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this
   list of conditions and the following disclaimer in the documentation and/or
   other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must
   display the following acknowledgement: "This product includes cryptographic
   software written by Eric Young (eay@cryptsoft.com)". The word 'cryptographic'
   can be left out if the routines from the library being used are not cryptographic
   related.
4. If you include any Windows specific code (or a derivative thereof) from the
   apps directory (application code) you must include acknowledgement:
   "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTicular PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.

The license and distribution terms for any publicly available version or derivative
of this code cannot be changed. i.e. this code cannot simply be copied and put
under another distribution license [including the GNU Public License.]

This product includes cryptographic software written by Eric Young.


Permission to use, copy, modify, and distribute this software for any purpose with
or without fee is hereby granted, provided that the above copyright notice and this
permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND INTERNET SOFTWARE
CONSORTIUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS. IN NO EVENT SHALL INTERNET SOFTWARE CONSORTIUM
BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL
DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF
USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

X Window System is a trademark of The Open Group.

If you are viewing this information softcopy, photographs and color illustrations may not appear.

You can obtain softcopy from the z/OS Collection (SK3T-4269), which contains BookManager and PDF formats.

Programming interface information

This publication documents information NOT intended to be used as Programming Interfaces of z/OS Communications Server.
Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks of other companies:

Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

PostScript is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.
Bibliography

This bibliography contains descriptions of the documents in the z/OS Communications Server library.

z/OS Communications Server documentation is available in the following forms:

- Online at the z/OS Internet Library web page at www.ibm.com/systems/z/os/zos/bkserv/
- In softcopy on CD-ROM collections. See "Softcopy information" on page xxx.

z/OS Communications Server library updates

An index to z/OS Communications Server book updates is at http://www.ibm.com/support/docview.wss?uid=swg21178966. Updates to documents are also available on RETAIN and in information APARs (info APARs). Go to http://www.ibm.com/software/network/commserv/zos/support to view information APARs. In addition, Info APARs for z/OS documents are in z/OS and z/OS.e DOC APAR and PTF ++HOLD Documentation, which can be found at http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS.

z/OS Communications Server information

z/OS Communications Server product information is grouped by task in the following tables.

Planning

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS Communications Server: New Function Summary</td>
<td>GC31-8771</td>
<td>This document is intended to help you plan for new IP for SNA function, whether you are migrating from a previous version or installing z/OS for the first time. It summarizes what is new in the release and identifies the suggested and required modifications needed to use the enhanced functions.</td>
</tr>
<tr>
<td>z/OS Communications Server: IPv6 Network and Application Design Guide</td>
<td>SC31-8885</td>
<td>This document is a high-level introduction to IPv6. It describes concepts of z/OS Communications Server’s support of IPv6, coexistence with IPv4, and migration issues.</td>
</tr>
</tbody>
</table>

Resource definition, configuration, and tuning

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS Communications Server: IP Configuration Guide</td>
<td>SC31-8775</td>
<td>This document describes the major concepts involved in understanding and configuring an IP network. Familiarity with the z/OS operating system, IP protocols, z/OS UNIX System Services, and IBM Time Sharing Option (TSO) is recommended. Use this document in conjunction with the z/OS Communications Server: IP Configuration Reference.</td>
</tr>
<tr>
<td>Title</td>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| z/OS Communications Server: IP Configuration Reference               | SC31-8776 | This document presents information for people who want to administer and maintain IP. Use this document in conjunction with the z/OS Communications Server: IP Configuration Guide. The information in this document includes:  
  - TCP/IP configuration data sets  
  - Configuration statements  
  - Translation tables  
  - SMF records  
  - Protocol number and port assignments                                                                                           |
| z/OS Communications Server: SNA Network Implementation Guide         | SC31-8777 | This document presents the major concepts involved in implementing an SNA network. Use this document in conjunction with the z/OS Communications Server: SNA Resource Definition Reference.                                                                                                                                                                     |
| z/OS Communications Server: SNA Resource Definition Reference        | SC31-8778 | This document describes each SNA definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect SNA. Use this document in conjunction with the z/OS Communications Server: SNA Network Implementation Guide.                                                                 |
| z/OS Communications Server: SNA Resource Definition Samples          | SC31-8836 | This document contains sample definitions to help you implement SNA functions in your networks, and includes sample major node definitions.                                                                                                                                                                |
| z/OS Communications Server: IP Network Print Facility              | SC31-8833 | This document is for system programmers and network administrators who need to prepare their network to route SNA, JES2, or JES3 printer output to remote printers using TCP/IP Services.                                                                                                                                                  |

**Operation**

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS Communications Server: IP User’s Guide and Commands</td>
<td>SC31-8780</td>
<td>This document describes how to use TCP/IP applications. It contains requests that allow a user to log on to a remote host using Telnet, transfer data sets using FTP, send and receive electronic mail, print on remote printers, and authenticate network users.</td>
</tr>
<tr>
<td>z/OS Communications Server: IP System Administrator’s Commands</td>
<td>SC31-8781</td>
<td>This document describes the functions and commands helpful in configuring or monitoring your system. It contains system administrator’s commands, such as TSO NETSTAT, PING, TRACERTE and their UNIX counterparts. It also includes TSO and MVS commands commonly used during the IP configuration process.</td>
</tr>
<tr>
<td>z/OS Communications Server: SNA Operation</td>
<td>SC31-8779</td>
<td>This document serves as a reference for programmers and operators requiring detailed information about specific operator commands.</td>
</tr>
<tr>
<td>z/OS Communications Server: Quick Reference</td>
<td>SX75-0124</td>
<td>This document contains essential information about SNA and IP commands.</td>
</tr>
</tbody>
</table>
# Customization

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
</table>
| z/OS Communications Server: SNA Customization | SC31-6854 | This document enables you to customize SNA, and includes the following:  
• Communication network management (CNM) routing table  
• Logon-interpret routine requirements  
• Logon manager installation-wide exit routine for the CLU search exit  
• TSO/SNA installation-wide exit routines  
• SNA installation-wide exit routines |

# Writing application programs

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference</td>
<td>SC31-8788</td>
<td>This document describes the syntax and semantics of program source code necessary to write your own application programming interface (API) into TCP/IP. You can use this interface as the communication base for writing your own client or server application. You can also use this document to adapt your existing applications to communicate with each other using sockets over TCP/IP.</td>
</tr>
<tr>
<td>z/OS Communications Server: IP CICS Sockets Guide</td>
<td>SC31-8807</td>
<td>This document is for programmers who want to set up, write application programs for, and diagnose problems with the socket interface for CICS using z/OS TCP/IP.</td>
</tr>
<tr>
<td>z/OS Communications Server: IP IMS Sockets Guide</td>
<td>SC31-8830</td>
<td>This document is for programmers who want application programs that use the IMS TCP/IP application development services provided by IBM’s TCP/IP Services.</td>
</tr>
<tr>
<td>z/OS Communications Server: IP Programmer’s Guide and Reference</td>
<td>SC31-8787</td>
<td>This document describes the syntax and semantics of a set of high-level application functions that you can use to program your own applications in a TCP/IP environment. These functions provide support for application facilities, such as user authentication, distributed databases, distributed processing, network management, and device sharing. Familiarity with the z/OS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.</td>
</tr>
<tr>
<td>z/OS Communications Server: SNA Programming</td>
<td>SC31-8829</td>
<td>This document describes how to use SNA macroinstructions to send data to and receive data from (1) a terminal in either the same or a different domain, or (2) another application program in either the same or a different domain.</td>
</tr>
<tr>
<td>z/OS Communications Server: SNA Programmer’s LU 6.2 Guide</td>
<td>SC31-8811</td>
<td>This document describes how to use the SNA LU 6.2 application programming interface for host application programs. This document applies to programs that use only LU 6.2 sessions or that use LU 6.2 sessions along with other session types. (Only LU 6.2 sessions are covered in this document.)</td>
</tr>
<tr>
<td>z/OS Communications Server: SNA Programmer’s LU 6.2 Reference</td>
<td>SC31-8810</td>
<td>This document provides reference material for the SNA LU 6.2 programming interface for host application programs.</td>
</tr>
<tr>
<td>z/OS Communications Server: CSM Guide</td>
<td>SC31-8808</td>
<td>This document describes how applications use the communications storage manager.</td>
</tr>
</tbody>
</table>
### Title Number Description

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>z/OS Communications Server: CMIP Services and Topology Agent Guide</strong></td>
<td>SC31-8828</td>
<td>This document describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The document provides guide and reference information about CMIP services and the SNA topology agent.</td>
</tr>
</tbody>
</table>

### Diagnosis

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>z/OS Communications Server: IP Diagnosis Guide</strong></td>
<td>GC31-8782</td>
<td>This document explains how to diagnose TCP/IP problems and how to determine whether a specific problem is in the TCP/IP product code. It explains how to gather information for and describe problems to the IBM Software Support Center.</td>
</tr>
<tr>
<td><strong>z/OS Communications Server: ACF/TAP Trace Analysis Handbook</strong></td>
<td>GC23-8588</td>
<td>This document explains how to gather the trace data that is collected and stored in the host processor. It also explains how to use the Advanced Communications Function/Trace Analysis Program (ACF/TAP) service aid to produce reports for analyzing the trace data information.</td>
</tr>
<tr>
<td><strong>z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures and z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT</strong></td>
<td>GC31-6850, GC31-6851</td>
<td>These documents help you identify an SNA problem, classify it, and collect information about it before you call the IBM Support Center. The information collected includes traces, dumps, and other problem documentation.</td>
</tr>
<tr>
<td><strong>z/OS Communications Server: SNA Data Areas Volume 1 and z/OS Communications Server: SNA Data Areas Volume 2</strong></td>
<td>GC31-6852, GC31-6853</td>
<td>These documents describe SNA data areas and can be used to read an SNA dump. They are intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.</td>
</tr>
</tbody>
</table>

### Messages and codes

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
</table>
| **z/OS Communications Server: SNA Messages**                         | SC31-8790 | This document describes the ELM, IKT, IST, IUT, IVT, and USS messages. Other information in this document includes:  
  • Command and RU types in SNA messages  
  • Node and ID types in SNA messages  
  • Supplemental message-related information |
| **z/OS Communications Server: IP Messages Volume 1 (EZA)**            | SC31-8783 | This volume contains TCP/IP messages beginning with EZA.                                                                                   |
| **z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)**       | SC31-8784 | This volume contains TCP/IP messages beginning with EZB or EZD.                                                                               |
| **z/OS Communications Server: IP Messages Volume 3 (EZY)**            | SC31-8785 | This volume contains TCP/IP messages beginning with EZY.                                                                                   |
| **z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)**       | SC31-8786 | This volume contains TCP/IP messages beginning with EZZ and SNM.                                                                          |
| **z/OS Communications Server: IP and SNA Codes**                     | SC31-8791 | This document describes codes and other information that appear in z/OS Communications Server messages.                                     |
Index

Numerics

0.0.0.0 address in traps from the SNMP agent 625

A

ABEND dumps 13
abends 349
CEEDUMP 25
dumps of ASIDs 52
FTP client 480
FTP server 445
LPD (line printer daemon) 412
NCPROUTE 783
OMPROUTE 752
platform 25
SMTP (simple mail transfer protocol) 526
SNALINK LU0 548
SNALINK LU6.2 578
SNMP (simple network management protocol) 610
SNTP (simple network time protocol) 879
system 25
TN3270 Telnet client 515
tn3270E Telnet server 506
user 25
access control problems 407
accessibility 971
active routes versus static routes 922
ADDRBLOK data set 531
addressing on the Internet 923
Aggressive mode 935
allocation tables, DU 206
AMBLIST 18
APAR (authorized program analysis report) 5
API, TCPIPCS subcommand 197
APPE for data set fails, FTP 453
ARP cache, querying 43
AT-TLS
common errors 706
common startup errors 701
diagnosing problems 701
subcommand 703
traces 703
authorized program analysis report (APAR) 5
autolog, FTP server problems with 444

B

batch job, formatting component traces 59
bridge 917, 919, 925

C

C_INET PFS
OMPROUTE behavior in 752
CEEDUMP
description 25
for SNMP abends 610
CICS (customer information control system) 843
CICS, location of trace output 12
Class D group addresses 923
commit bit
Quick mode 940
Quick mode (phase 2) SA states 941, 942
commit-bit
support in the IKE daemon 948
common INET
OMPROUTE behavior in 752
common storage tracking 18
Communications Server for z/OS, online information xxxii
component ID, TCP/IP 22, 23
component trace
displaying status 51
genesis description 14
managed data set 8
OMPROUTE 749
stopping 52
TN3270E Telnet server 514
use with DDNS (dynamic domain name server) 585
use with OMPROUTE, see also OMPROUTE application
trace 772
viewing data without an abend 52
customer information control system (CICS) 843
customer number 22
data block, displaying 304
data buffers, displaying 304
data link control (DLC) 561
data traces
displaying 149
data traces (continued)
  general description 15
  starting 148
date and time formatting 307
DATTRACE command 148
DB2
  query support for FTP client 471
  query support for FTP server 439
DBCS (double-byte character set)
  FTP client 466
  FTP server 438
DDNS (dynamic domain name server) 579
DDNS problems
  console messages 579
  debug option 582
  name server signals 584
  nsupdate 585
  onlookup and nslookup 581
  resolver directive 583
  syslog messages 580
  TSO DIG 581
DDNS, location of trace output 9
default information, IKE 349
default information, PagentSyslogLevel 349
debug information, TSO DIG 581
debugging switches, z/OS UNIX sendmail 537
DEBUGTRC 761
debug information, FDC client 453
device interface, connecting 200, 202
device interface, displaying 55, 199
diagnosing FTP transfer failures with EZAOFP89E 474
diagnostic aids for IPv6 support 541
disability 971
dispatchable unit control block (DUCB), displaying 204
dispatchable units, displaying TCP/IP 242
DISPLAY command, use with LUs 570
display commands 18
DISPLAY TCPIP,STOR 18
DISPLAY TRACE command 51
distributed protocol interface (DPI) 605, 781
DLC (data link control) 561
DNS, online information 34
domain name server, dynamic (DDNS) 579
double-byte character set (DBCS)
  FTP client 466
  FTP server 438
DPI (distributed protocol interface) 605, 781
DU allocation tables, displaying 206
DUAET, TCPIPCS subcommand 204
DUCB, displaying 204
DUCB, TCPIPCS subcommand 206
DUMP command 52, 53
dump, selecting 7
dumps
  ABEND 8
  definition 8, 25
  FIST 14, 16, 17
  stand-alone 14
  SVC 8, 14
  SYSAEND 13
  SYMMDUMP 13
  SYSUDUMP 13
  to obtain component trace data 52
Dynamic VIPA 373

E
  enable, disable, and display the status of the OMPROUTE
  CTRACE 776
  EPSV commands fail 449
  errno reason code, displaying 293
  ERRNO, IPCS subcommand 293
  errnojr reason code, displaying 293
  error exit codes, FTP server 436
  establishing security associations, IKE 334
  Ethernet
    defining for NCPROUTE 782
    DIX V2 921
  event trace (SYSTCPIP)
    TCP/IP stacks 60
    Telnet 60
  execution environments
    FTP client 465
  Express Logon
    analyzing start problems with the DCAS 850
    client interface problems 851
    overview 849
  Express Logon, location of trace output 12
  extended tracing, FTP 462
  EZAOFP89E, diagnosing FTP transfer failures 474
  EZAFIP9P start procedure 436
  EZD1215-EZD1234 409

F
  FDDI (fiber distributed data interface) 919
  FFST (first failure support technology)
    console 17
    dump 14, 16
    general description 16
    minidump 17
    probes 907, 908
    sdump 16
  fiber distributed data interface (FDDI) 919
  field maintenance ID 23
  file tagging support
    FTP client 468
    file tagging, FTP client
      ASCII file transfers 468
      binary file transfers 470
  file transfer protocol (FTP), see also FTP 435
  File transfers, restarting 473
  firewall, FTP connections 483
  first failure support technology (FFST) 16
  first failure support technology (FFST), see also FFST 907
  formatting component traces using a batch job 59
  formatting event trace records for TCP/IP stacks 71
  formatting IKE daemon trace records 354
  formatting packet trace using a batch job 148
  FRCA, TCPIPCS subcommand 210
  FTP client 465
    customizing configuration parameters 465
    DCB 471
    DBCS (double-byte character set) 466
    execution environments 465
    exit 466
    file tagging support 468
    MBCS (mult-byte character set) 467
    naming considerations 466

988 z/OS V1R11.0 Comm Svr: IP Diagnosis Guide
FTP client (continued)
  naming MVS data sets 437
  naming z/OS UNIX file system files 437
  overview 465
  restarting file transfers 473
  SQL support 471
  trace destination 486
  translation and data conversion support 466
  using FTP.DATA 465
  using TCP/IP.DATA 465
  using traces 485

FTP client problems
  “Unknown Host” message 482
  abends 480
  AT-TLS 448
  client rejects the RESTART subcommand 473
  client rejects the SRESTART subcommand 474
  configuration values 480
  connection through SOCKS server to FTP server fails 481
  conversion between <codeset> and UCS-2 482
  data connection 483
  data transfer 482
  diagnosing FTP transfer failures with EZA2589E 474
  documentation for IBM Support Center 486
  firewall 483
  logging into the server 480
  PDS member statistics 485
  server rejects password 482
  server rejects PORT or EPRT command with 504 replies 483
  SOCKS configuration file 480
  SOCKS connection fails 481
  starting 479
  tracing 485
  wrong SOCKS server 481

FTP server
  controlling traces with MODIFY operator command 458
  customizing server parameters 436
  customizing the start procedure 436
  DB2 query support 439
  DBCS (double-byte character set) 438
  definitions and setup 436
  error exit codes 436
  LOADLIB directory information 455
  location of trace output 9
  logging activity 442
  MBCS (multibyte character set) 438
  name considerations 436
  naming MVS data sets 437
  naming z/OS UNIX file system files 437
  overview 435
  role of TCP/IP.DATA 436
  SBCS (single-byte character set) 438
  SQL 439
  SQL support 439
  traces 456
  translation and data conversion support 437

FTP server problems
  abends 445
  AT-TLS 448
  autoilog 444
  checkpoint markers 454
  client abends during RETR 454
  command failure 453
  common z/OS UNIX 443
  configuration values 444

FTP server problems (continued)
  connection 446, 448
  data set allocation 451
  data set disposition 454
  data transfer 449
  documentation for IBM Support Center 464
  incomplete initialization 443
  job entry subsystem (JES) 441
  login failure 447
  lost messages and traces 445
  MVS data set 453
  password validation fails 446
  port specification 444
  session 446
  terminated data transfer 454
  user exit routine 445
  working directory 448

FTP session, IPv4 237
FTP.DATA data set
  definition 436
  specifying working directory 448
  FTP.DATA, TRACE statement in 457

G
  gateway (router)
    definition 917
    indirect routing 926
  internet addressing 925
  LPR and LPD problems 412
  using Ping to check connectivity 38
  gateway IP address 237
  GATEWAY statement
    configuring routing tables 930
    NCPROUTE GATEWAYS data set 792
    problems uncovered by NETSTAT 42
    problems with X.25 806, 808, 812
    subnet masks 928
    use with NCPROUTE 785
    use with OMPROUTE 751
  general tracing, FTP 459
  global work area (GWA) 848
  GWA (global work area) 848

H
  hangs, collecting documentation 27
  hardware problem 12
  hash tables, displaying 212
  HASH, TCP/IPCS subcommand 212
  header fields, displaying ICMP 296
  header fields, displaying IP 297
  header fields, displaying TCP 306
  header, displaying Resolver 299
  header, displaying UDP 308
  header, system dump 216
  HEADER, TCP/IPCS subcommand 216
  HELP, TCP/IPCS subcommand 218

I
  IBM Software Support Center, contacting xxvi
  IBM Software Support Center, documentation for FTP client problems 486
  FTP server problems 464
  LPD (line printer daemon) problems 412

Index 989
LPD (line printer daemon) (continued)
timeouts, hangs, waits 412
unsupported filter 427
LPD server, location of trace output 9
LPR (line printer requester)
activating client traces 415
creating client traces 415
FILTER X option 418
garbled data 413
identifying a port 419
missing data 413
nonworking options 415
overview 411
timeouts, hangs, waits 412
X LATE option 420
LPR client, location of trace output 8

M
Main mode 934
scenarios 947, 948
mainframe
education xxx
management information base (MIB) 605, 782
MAP, TCPIPCS subcommand 223
master TCB control block, displaying 268
master trace 15
master UDP control block (MUCB), displaying 288
maximum transmission unit (MTU) 918
MAXSOC value 91
MBCS (multibyte character set)
FTP client 467
FTP server 438
memory allocation, leakage problems 678
message block, displaying the stream 304
message EZYTFT47I 480
message suppression 445
messages, EZD1215-EZD1234 409
MIB (management information base) 605, 782
MODIFY command
controlling the FTP server traces 458
with NCPROUTE 792
module ID, displaying 293
module tables, displaying 226
motif 601
Motif, location of trace output 10
MTABLE, TCPIPCS subcommand 226
MTU (maximum transmission unit) 918
MUCB (master UDP control block), displaying 288
multibyte character set (MBCS)
FTP client 467
FTP server 438
multicast, Class D addresses 923
multilevel security consistency check messages 409
multiple stacks 30, 55, 611, 752
MVS data set
FTP naming conventions 437
FTP server unable to find 453
MVS system console log for REXECD 589

N
ame considerations for z/OS UNIX FTP 436
name server signals 584
name server, diagnosing problems with 579
named command 582
NAT firewall 482
NCPROUTE
abends 783
activating global traces 792
activating selective traces 792
bad connections 783
communication with SNMP 781, 782
defining 782
incorrect output 789
overview 779
PING failure 787
RIP (routing information protocol) commands 780
session outages 790
trace example 793
NCROUTE, location of trace output 11
negotiating security associations 933
netdata, z/OS UNIX Telnet 490
Netstat ARP/-R 43
netstat command 41
displaying ADDRESS and LINK values 42
displaying device configuration values 42
displaying routing tables 42
Netstat DEVLINKS/-d 42
Netstat HOME/-h 42
Netstat ROUTE/-r 42
querying ARP cache 43
NETSTAT command
diagnosing a timeout 41
displaying data trace 149
verifying SNALINK LU6.2 569
with IMS socket interface 833, 834
with SNALINK LU0 549
Netstat DEVLINKS/-d 42
Netstat HOME/-h 42
Netstat ROUTE/-r 42
NetView 17, 606, 607, 616, 789
network IP addresses 923
Network SLAPM2 subagent
statements 609
network, problems with SNA 563, 565
NSLOOKUP command 581
nsupdate command 585
Index 991
orexed (remote execution protocol daemon), z/OS UNIX,
activating debug trace 598
orshd (remote shell daemon), z/OS UNIX
activating debug trace 598
OSPF (open shortest path first) 749
oteagent, diagnostic messages sent to wrong file 489

P

packet trace
general description 15, 94
starting 95
with IMS socket interface 833
with SNALINK LU0 551
with SNALINK LU6.2 568, 576
PageTSyslogLevel 349
PASV and EPSV commands fail 449
PASV commands fail 449
patricia trees, displaying 277
PDU s (protocol data units)
NCPROUTE 780
SNMP (simple network management protocol) 605
PFS (physical file structure) 148
phase 1 933
phase 1 SA states 936
phase 2 939
physical file structure (PFS) 148
Ping command 37
SNMP (simple network management protocol) 43
using 37
verifying device connection 38
verifying network definition 38
verifying onetstat -h 38
verifying packets sent and received 38
verifying route back to local host 38
verifying route definition 38
verifying router can forward 38
verifying TCP/IP 38
PING command
NCPROUTE client problems 787
with IMS socket interface 819
X.25 NPSI 812
Ping timeout 41
PKTTRACE statement 94
platform abends 25
policies 683
policy agent problems
gathering diagnostic information 655
initialization problems 657
LDAP problems 673
leakage problems 678
memory allocation 678
overview 653
policy definition problems 659
service policy 654
service policy scope 654
policy agent terms
differentiated services 654
integrated services 654
quality of service (QoS) 654
resource reservation protocol (RSVP) 654
service differentiation 654
service level agreement (SLA) 654
service policy 654
Policy Agent, location of trace output 10
POLICY, TCPIP/CS subcommand 227
popper, diagnostic aids for z/OS UNIX 544
Popper, location of trace output 9
prerequisite information xxx
print output, sending IPCS 304
problem number 22
Problem reporting
mailed tape 19
PROFILE, TCPIP/CS subcommand 229
program objects 18
protocol data units (PDU s)
NCPROUTE 780
SNMP (simple network management protocol) 605
PROTOCOL, TCPIP/CS subcommand 233
ptydata, z/OS UNIX Telnet 490

Q

QoS 654
policy scope 654
Quick mode
(phase 2) SA states with commit-bit support 942
(phase 2) SA states without commit-bit support 941
overview 939
with commit bit 940

R

RACF (resource access control facility), VARY command
and 148
RAW, TCPIP/CS subcommand 235
reason codes, displaying 293
register save area (RSA) 204
rejected password, FTP 482
release number, TCP/IP 22
remote execution protocol daemon, see also REXECD 587, 595
remote execution protocol, see also REXECD 587, 595
remote shell client, see also RSH 587
remote shell daemon, see also RSH 595
Resolver 853
Resolver header, displaying 299
RESOLVER trace 533
resolver, debug 583
RESOLVER, IPCS subcommand 299
Resolver, location of trace output 12
resource access control facility (RACF), VARY command
and 148
resource measurement facility 18
RESTART subcommand, client rejects 473
RETAIN database 5
RETR for data set fails, FTP 453
REXEC (remote execution protocol), non-z/OS UNIX
activating debug trace 588
overview 587
problem documentation 587
trace example 588
REXEC (remote execution protocol), z/OS UNIX
debug trace 596
trace example 596
REXECD (remote execution protocol daemon), non-z/OS
UNIX
command options for tracing 589
overview 588
problem documentation 589
trace example 589
REXECD (remote execution protocol daemon), z/OS UNIX
  trace example 598
REXECD (remote execution protocol daemon), z/OS UNIX,
  activating debug trace 598
REXX executables for TCP/IP 194
RFC (request for comments) 953
  accessing online xxxii
RIP (routing information protocol)
  NCPROUTE implementation 780, 789
  OMPROUTE implementation 749
RNFR for data set fails, FTP 453
RNTO for data set fails, FTP 453
ROUTE, TCP/IP subcommand 237
router (gateway)
  definition 917
  indirect routing 926
  internet addressing 925
  LPR and LPD problems 412
  using Ping to check connectivity 38
routing
  datagram algorithm 926
  datagram algorithm with subnets 929
  direct 926
  dynamic 931
  dynamic IPv4 931
  dynamic IPv6 931
  indirect 926
  static 930
routing control blocks, displaying 237
routing information protocol (RIP)
  NCPROUTE implementation 780, 789
  OMPROUTE implementation 749
routing information, displaying 42
routing table entry, displaying 237
RSA (register save area) 204
RSH 597
RSH (remote shell client), non-z/OS UNIX
  client trace using SEND command 591
  overview 587
  trace example 588
RSH, z/OS UNIX
  debug trace 597
  trace example 597
RSHD (remote shell daemon), z/OS UNIX
  activating debug trace 598
  random errors 599
  trace example 599
RSVP agent problems
  application problems 685
  gathering diagnostic information 684
  initialization problems 684
  log file example 686
  overview 681
  reservation objects 682
  reservation styles 682
  reservation types 682
  service policy problems 685
RSVP Agent, location of trace output 10
RSVP processing 683
secure sockets layer (SSL) encryption 513
security associations
  aggressive mode 935
  common problems 335
  interpreting IKE daemon phase 1 SA states 936
  interpreting IKE daemon phase 2 SA states 941
  ISAKMP Main mode limitations 947
  Main mode 934
  negotiating 933
  overview of negotiating 933
  phase 1 933
  phase 2 939
  problems establishing 334
  Quick mode 939
  Quick mode with commit bit 940
selecting a dump 7
selecting a service aid 12
selecting a trace 8
sendmail, diagnostic aids for z/OS UNIX 537, 539
server data block, displaying the TSDB 281
server data extension, displaying the TSDX 282
service aid, selecting 12
service aid, selection 7
service level indication processing (SLIP) 12, 19
service policy scopes
  DataTraffic 654
  RSVP 654
services component trace, TCP/IP 350
session problems
  FTP 446
  SNALINK LU0 hangs 548
  SNALINK LU0 outages 550
tsn270 Telnet client hangs 515
  TN3270E Telnet server hangs 508
  TN3270E Telnet server outages 512
  X.25 NPSI hangs 812
SETPRINT, IPCS subcommand 304
shortcut keys 971
signals, name server 584
simple mail transfer protocol, see also SMTP 525
simple network management protocol, see also SNMP 605
simple network time protocol, see also SNTP 879
single-byte character set (SBCS)
  FTP client 466
  FTP server 438
  SKMSG, IPCS subcommand 304
SLIP (service level indication processing) 12, 19
SMTP (simple mail transfer protocol)
  abends 526
  ADDRBLOCK data set 531
  delivering 527
  environment 525
  incorrect output 529
  looping problems 529
  re-resolution of queued mail 530
  receiver overview 525
  RESOLVER trace 533
  sender overview 525
  spooling problems 527
SMTP, location of trace output 9
SNALINK LU0
  abends 548
  location of trace output 9
  session hangs 548
  session outages 550

Index 993
SNALINK LU6.2
address space problems 559, 569
configuration mistakes 557
data corruption problems 568
data loss problems 566
DLC connection problems 561
location of trace output 9
network components 555, 556
network problems 563, 565
traces 573, 577, 578
using NETSTAT 569
SNAP (subnetwork access protocol) 922
SNMP (simple network management protocol)
abends 610
agent 610
agent does not respond 623
client overview 606
communication with NCPROUTE 781
connection problems 612
I/O error using PING 625
incorrect output 617
interaction with NCPROUTE 779
MIB 605
multiple stack problems 611
NetView 606
query engine 616, 617, 646
snmp command 606, 607, 628
snmpd command 606
socket calls 609
starting with snmp 628
subagent 608
traces 627, 628, 629
unknown variable in output 617
variable format incorrect 620
variable value incorrect 622
SNMP agent
configuration data sets 607
security methods 608
SNMP agent, 0.0.0.0 address in traps 625
snmp command 606, 607, 628
SNMP problems
related to NCPROUTE 783
SNMPIUCV subtask 616
SNMP traces
agent trace example 632
query engine IUCV trace example 646
query engine trace example 634
subagent trace example 633
SNMP, location of trace output 10
SNMPIUCV subtask 616
SNTP (simple network time protocol) 879
abends 879
dump trace 879
socket calls, SNMP (simple network management protocol) 609
socket control block, displaying 240
socket data, tracing 148
SOCKET, TCPIPCS subcommand 240
softcopy information xxx
specifying options after initialization 354
specifying options at initialization 351
specifying trace options 350
SQL
FTP client support 471
FTP server support 439
problems with FTP 439, 472
SRESTART subcommand, client rejects 474
SSA REXX/370 runtime libraries 194
SSL (secure sockets layer) encryption 513
stand-alone dumps 14
start procedure, FTP server 436
starting TN3270E Telnet subagent trace 630
state, displaying TCP/IP 242
STATE, TCPIPCS subcommand 242
static routes compared to active routes 922
status, displaying device 42
STDOUT, use by OMPROUTE 749
STOR for data set fails, FTP 453
storage map, displaying 223
storage, displaying TCP/IP 242, 264
STORAGE, TCPIPCS subcommand 264
stream control blocks, displaying 266
stream message block, displaying 304
STREAM, TCPIPCS subcommand 266
subagent programs, user-written for SNMP (simple network management protocol) 605
subagent, Network SLAPM2 609
subagent, OMPROUTE 608
subagent, SNMP (simple network management protocol) 608
subagents 607
subnetwork access protocol (SNAP) 922
subtasks, displaying TCP/IP 242
SVC dumps 14
SWSA 393
syntax diagram, how to read xxvii
syntax for the CTRACE command 72
SYSAEND dump 13
SYSEND dumping agents 489
SYSDUMP 13
sysplex
routing problems 397
SWSA 393
Sysplex distributor 373
diagnosing 374
overview 373
sysplex-wide dynamic source VIPAs 385
Sysplex-wide Security Association (SWSA) 393
SYSPLEX exports problems 387
SYSPRINT data set 4
SYSTCPIPS 149, 697
system abends 25
system dump header, displaying 216
system trace 15
SYSUDUMP 13
T

task interface element (TIE) 847
task related user exit (TRUE), CICS sockets 848
tasks (gerund phrase)
step for 597
access control support
overview 407

994 z/OS V1R11.0 Comm Svr: IP Diagnosis Guide
tasks (continued)
activating LPR client traces
   step for 415
activating server traces
   step for 421
activating the z/OS UNIX RSHD debug trace
   step for 598
analyzing hangs
   steps for 27
analyzing incorrect mail output
   steps for 530
analyzing incorrect output (client)
   steps for 517
analyzing incorrect output (server)
   steps for 510
analyzing mail delivery problems
   steps for 528
analyzing routing failures
   steps for 788
analyzing session hangs
   steps for 549
analyzing session hangs (client)
   steps for 516
analyzing session hangs (server)
   steps for 509
analyzing session outages (server)
   steps for 512
checking DLC connection status
   steps for 561
checking network connection problems
   steps for 563
collecting documentation
   steps for 26
creating client trace output
   step for 415
creating server trace output
   step for 422
determining how your IDS policies have been mapped by
   the stack
   step for 695
determining which IDS policies are active in Policy Agent
   step for 695
determining why IDS SyslogD output is missing
   steps for 696
diagnosing
   overview 3
diagnosing a AT-TLS problem
   steps for 701
diagnosing child-server transactions not starting
   steps for 846
diagnosing CICS listener not initialized
   steps for 844
diagnosing CICS socket interface not initialized
   steps for 844
diagnosing garbled data
   steps for 413
diagnosing hung CICS tasks
   steps for 846
diagnosing IKE daemon problems
   overview 333
diagnosing incorrect output
   steps for 790
diagnosing IP security problems
   overview 725
   steps for 726
tasks (continued)
diagnosing LDAP object retrieval problems
   steps for 673
diagnosing logon problems
   steps for 811
diagnosing LPR working with some options only
   steps for 415
diagnosing no response to commands
   steps for 837
diagnosing Policy Agent/Sysplex distribution problems
   steps for 678
diagnosing problems
   steps for 4
diagnosing server connection problems
   steps for 30
diagnosing session outages
   steps for 791
diagnosing SNMP problems
   steps for 609
diagnosing SWSA problems
   steps for 393
diagnosing sysplex distributor problems
   overview 373
diagnosing sysplex problems
   steps for 374
diagnosing sysplex routing problems
   steps for 397
diagnosing sysplex-wide dynamic source VIPAs for TCP
   connections problems
   steps for 385
diagnosing SYSPLEXPORTS problems
   steps for 387
diagnosing TCP/IP clients unable to connect
   steps for 845
diagnosing timeouts, hangs, and waits
   steps for 412
diagnosing TRMD problems
   steps for 699
diagnosing truncated or missing print data
   steps for 413
diagnostic aids for AT-TLS support
   steps for 542
disabling the CTRACE after OMPROUTE has started
   steps for 777
disabling the CTRACE after the IKE daemon has started
   steps for 354
disabling the CTRACE at IKE daemon startup
   steps for 353
disabling the CTRACE at OMPROUTE startup
   step for 776
displaying the CTRACE status
   step for 354, 777
displaying the internal trace
   steps for 848
documenting data loss problems
   steps for 566
dynamic IPSec protection
   steps for 737
enabling the CTRACE after OMPROUTE has started
   steps for 776
enabling the CTRACE after the IKE daemon has started
   steps for 353
enabling the CTRACE at IKE daemon startup
   steps for 353
enabling the CTRACE at OMPROUTE startup
   steps for 776
tasks (continued)

entering a TCP/IP IPCS subcommand  
steps for 310

formatting component traces using IPCS panels  
steps for 55

invoking z/OS UNIX RSH trace  
step for 597

IP security operation  
steps for 732

IP security policy enforcement  
steps for 740

manual IPSec protection  
steps for 735

NCPROUTE connection problems  
steps for 784

negotiating security associations  
overview 933

network access  
steps for 36

obtaining AMATERSE  
steps for 21

obtaining component trace data with an external writer  
steps for 53

obtaining PUTDOC  
steps for 21

Policy definition problems  
step for 659

resolving the hostname  
steps for 853

selecting tools and service aids  
overview 7

setting up a SNALINK LU6.2 network  
steps for 556

setting up the IMS TCP/IP services socket interface system  
steps for 817

starting manager traces  
step for 628

starting network SNIAMM2 subagent traces  
step for 630

starting OMPROUTE subagent traces  
step for 630

starting SNMP agent traces  
step for 628

starting TCP/IP subagent traces  
step for 629

starting Telnet client traces  
step for 518

starting TN3270E Telnet subagent traces  
step for 630

starting TRAPFWD traces  
step for 631

stopping SNTTPD  
steps for 879

tools and service aids  
overview 13

undeliverable mail items  
steps for undeliverable mail items 527

using FTP client SQL support  
steps for 471

using FTP server SQL support  
steps for 439

using the batch option  
step for 312

using the CTRACE command  
steps for 57

verifying IP routing to a destination  
steps for 32

tasks (continued)

verifying network interface operation  
steps for 36

verifying server operation  
steps for 31

verifying the configuration  
steps for 409

TBEs (trace buffer entries), displaying 273

TCA (trace control area), displaying 273

TCB control blocks, displaying 268

TCB, TCP/PCS subcommand 268

TCP/PCS subcommand, TTLS 284

TCP/IP  
event trace (SYSTCPIP) 60

online information xxxii

protocol specifications 953

TCP/IP services component trace, IKE daemon 350

TCP/IP socket, location of trace output 8

TCP/IP state, displaying summary 242

TCP/IP storage, displaying summary 264

TCP/IP subagent, statements 608

TCPPDATA data set 436

TCPPCS subcommand  
API 197

CONFIG 199

CONTACT 200

COUNTERS 202

DUAF 204

DUCB 206

FRCA 210

HASH 212

HEADER 216

HELP 218

IPSEC 219

LOCK 222

MAP 223

MTABLE 226

POLICY 227

PROFILE 229

PROTOCOL 233

RAW 235

REXX executables for 194

ROUTE 237

SOCKET 240

STATE 242

STORAGE 264

STREAM 266

symbols 197

syntax 194

TCB 268

TELNET 270

TECMET 272

TIME 272

TREE 273

TSDB 281

TSDX 282

TTL 284

UDP 288

VMCF 290

XCF 292

TCP/PCS subcommand, TTLS 284

Technotes xxx

Telnet  
protocol commands and options 522
TELNET, TCPIPCS subcommand 270
Telnet, tn3270
   incorrect output from client 517
   starting client traces 518
Telnet, TN3270 client
   client abends 515
   client definitions 515
   client overview 514
   client session hangs 515
   client trace example 518
Telnet, TN3270E Telnet server
   associating CTrace entry with client 514
   defining server 505
   displaying control blocks (TCPIPCS subcommand, Telnet) 270
   event trace (SYSTCPIP) 60
   incorrect output from server 510
   server abends 506
   server logon problems 506
   server overview 505
   server session hangs 508
   server session outages 512
SSL encryption 513
Telnet, z/OS UNIX
   -t -D all example 490
   arrow keys do not respond 489
   debug trace options 490
   diagnostic messages sent to wrong file 489
   improper authority error 489
   locked keyboard 489
   netdata and ptydata 490
   utmp entries 503
termination notification facility (TNF) 837
TIE (task interface element) 847
time and date formatting 307
timeout
Ping 41
timer control blocks, displaying 272
TIMER, TCPIPCS subcommand 272
TN3270E Telnet subagent 609
TN3270E Telnet subagent trace, starting 630
TNF (termination notification facility) 837
TOD, IPCS subcommand 307
token-ring
   defining for NCPROUTE 782
   IEEE 802.5 920
tool or service aid, selection 7
tool, selection 7
trace buffer entries (TBEs), displaying 273
trace control area (TCA), displaying 273
TRACE CT command 50, 70
trace entries, FTP 445
trace options, specifying 350
trace output, location of (continued)
   RSVP Agent 10
   SMTP 9
   SNALINK LU0 9
   SNMP 10
   TCP/IP socket 8
   TRMD 10
   X.25 NPSI 11
   Xwin 10
trace records, IKE daemon 354
TRACE RESOLVER
   MVS batch job environment 860
   output location 859
   resolver 857
   z/OS environment 859
   z/OS UNIX batch environment 860
   z/OS UNIX shell environment 859
trace, selecting 8
TRACE, TCPIPCS subcommand 273
ttraceroute command 43
TRACERTE command 43
traces
   AT-TLS 703
   component trace 14
   data trace 148
   data traces 15
   event trace for TCP/IP stacks 60
   for FTP client problems 485
   for FTP server problems 456
   for SNALINK LU6.2 networks 573
   GTF traces 15
   master trace 15
   packet trace 15, 551, 568, 576
   SNALINK LU0 debug 551
   socket API traces 74
   system trace 15
   TCP/IP internal 577
   tn3270 Telnet client 518
   TN3270E Telnet server 514
   VTAM buffer 578
   VTAM trace 16
   z/OS UNIX application debug traces 16
trademark information 981
translation and data conversion support 437
TRAFFWID trace example 651
traps 12
TREE, TCPIPCS subcommand 277
TRMID problems, gathering diagnostic information 699
TRMD, location of trace output 10
TRUE (task related user exist), CICS sockets 848
TSDB, TCPIPCS subcommand 281
TSDX, TCPIPCS subcommand 282
TSO console log from REXEC 588
TSO DIG command 581

U
UCBs (UDP control blocks), displaying 288
UDP control blocks (UCBs), displaying 288
UDP header, displaying 308
UDP, TCPIPCS subcommand 288
UDPHDR, IPCS subcommand 308
user abends 25
user exit routine, FTP 445
utmpx file 503
V
VARY TCPIP command  42
VERBEXIT    18
VIPA (virtual IP address)  751
virtual IP address (VIPA)  751
virtual machine communication facility (VMCF)  837
VMCF (virtual machine communication facility)  837
VMCF, TCPIPCS subcommand  290
VTAM buffer trace
   for tn3270 Telnet client session hangs  516
   for TN3270E Telnet server session hangs  509
VTAM trace  16
VTAM, online information xxxii

X
X Window System
   trace when XWTRACE=2   601
   trace when XWTRACELC=2  602
X.25 NPSI
   configuring for NPSI  807
   configuring for VTAM  807
   location of trace output  11
   logon problems  811
   overview  805
   session hangs  812
XCF, TCPIPCS subcommand  292
Xwin, location of trace output  10

Z
z/OS Basic Skills information center xxx
z/OS Basic Skills Information Center xxx
z/OS UNIX file system
   file naming conventions for FTP  437
z/OS UNIX RSH  597
z/OS, documentation library listing  983
Communicating Your Comments to IBM

If you especially like or dislike anything about this document, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Please send your comments to us in either of the following ways:
• If you prefer to send comments by FAX, use this number: 1+919-254-1258
• If you prefer to send comments electronically, use this address:
  - comsvrcf@us.ibm.com
• If you prefer to send comments by post, use this address:
  International Business Machines Corporation
  Attn:z/OS Communications Server Information Development
  P.O. Box 12195, 3039 Cornwallis Road
  Department AKCA, Building 501
  Research Triangle Park, North Carolina 27709-2195

Make sure to include the following in your note:
• Title and publication number of this document
• Page number or topic to which your comment applies.