z/OS Communications Server

IP Diagnosis Guide

Version 1 Release 7

GC31-8782-06

z/OS Communications Server

IBM

IP Diagnosis Guide

Version 1 Release 7

GC31-8782-06

Note:
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 847.

Seventh Edition (September 2005)

This edition applies to Version 1 Release 7 of z/OS (5694-A01) and Version 1 Release 7 of z/OS.e (5655-G52) 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+919-254-4028

Send the fax to "Attn: z/OS Communications Server Information Development"

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

World Wide Web:
http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

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.

Contents

Figures

Tables

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 V1R7. The information in this document supports both IPv6 and IPv4. Unless explicitly noted, information describes IPv4 networking protocol. 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 supports z/OS.e.

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 appendices provides 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

For immediate assistance, visit this Web site:
http://www.software.ibm.com/network/commserver/support/

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 of 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 875.

Conventions and terminology used in this document

This publication uses the following typographic conventions:

- Commands that you enter verbatim onto the command line are presented in **bold**.
- Variable information and parameters that you enter within commands, such as filenames, are presented in *italic*.

- System responses are presented in `monospace`.

For definitions of the terms and abbreviations 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; stronger request than recommendation
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 included in this document that do not have their own syntax described elsewhere in this document.

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.

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.

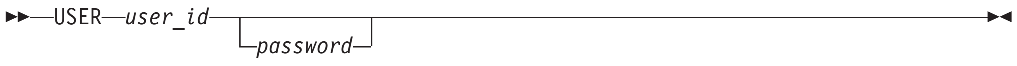
Parameters are classified as keywords or variables. For the TSO and MVS™ console commands, the keywords are not case sensitive. You can code them in uppercase or lowercase. If the keyword appears in the syntax diagram in both uppercase and lowercase, the uppercase portion is the abbreviation for the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

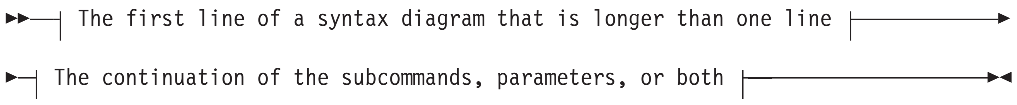
Syntax examples

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.



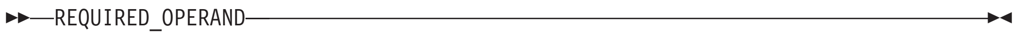
Longer than one line

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.


The first line of a syntax diagram that is longer than one line
The continuation of the subcommands, parameters, or both

Required operands

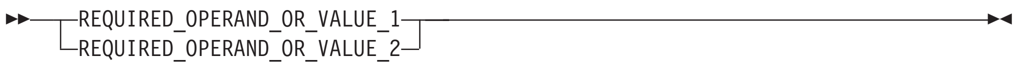
Required operands and values appear on the main path line.



You must code required operands and values.

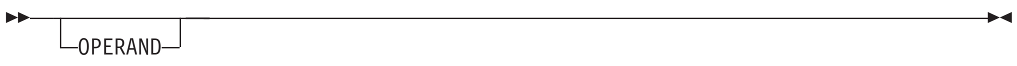
Choose one required item from a stack

If there is more than one mutually exclusive required operand or value to choose from, they are stacked vertically.



Optional values

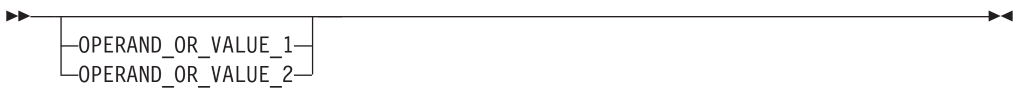
Optional operands and values appear below the main path line.



You can choose not to code optional operands and values.

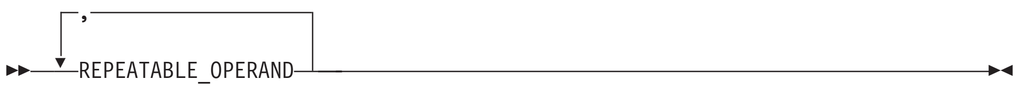
Choose one optional operand from a stack

If there is more than one mutually exclusive optional operand or value to choose from, they are stacked vertically below the main path line.



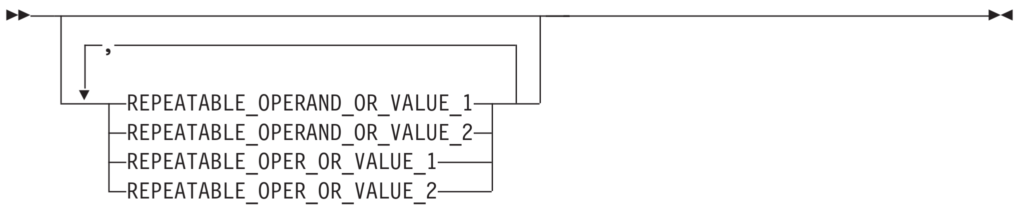
Repeating an operand

An arrow returning to the left above an operand or value on the main path line means that the operand or value can be repeated. The comma means that each operand or value must be separated from the next by a comma. If no comma appears in the returning arrow, the operand or value must be separated from the next by a blank.



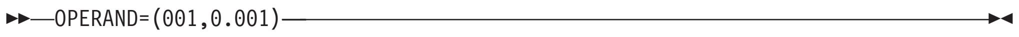
Selecting more than one operand

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.


,
REPEATABLE_OPERAND_OR_VALUE_1
REPEATABLE_OPERAND_OR_VALUE_2
REPEATABLE_OPER_OR_VALUE_1
REPEATABLE_OPER_OR_VALUE_2

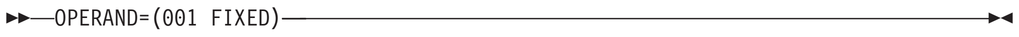
Nonalphanumeric characters

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.

```
   ┌─DEFAULT─┐
►►─┴─OPERAND─┴──────────────────────────────────────────────────►◄
```

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.

```
►►─variable─────────────────────────────────────────────────────►◄
```

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.

```
►►─┤ Syntax fragment ├──────────────────────────────────────────►◄
```

Syntax fragment:

```
├──1ST_OPERAND,2ND_OPERAND,3RD_OPERAND──────────────────────────┤
```

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 “z/OS Communications Server information” on page 857, 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.

Related information

This section contains subsections on:

- “Softcopy information”
- “Other documents” on page xxvii
- “Redbooks” on page xxviii
- “Where to find related information on the Internet” on page xxviii
- “Using LookAt to look up message explanations” on page xxx

Softcopy information

Softcopy publications are available in the following collections:

Titles	Order Number	Description
z/OS V1R7 Collection	SK3T-4269	This is the CD collection shipped with the z/OS product. It includes the libraries for z/OS V1R7, in both BookManager and PDF formats.

Titles	Order Number	Description
z/OS Software Products Collection	SK3T-4270	This CD includes, in both BookManager and PDF formats, the libraries of z/OS software products that run on z/OS but are not elements and features, as well as the *Getting Started with Parallel Sysplex®* bookshelf.
z/OS V1R7 and Software Products DVD Collection	SK3T-4271	This collection includes the libraries of z/OS (the element and feature libraries) and the libraries for z/OS software products in both BookManager and PDF format. This collection combines SK3T-4269 and SK3T-4270.
z/OS Licensed Product Library	SK3T-4307	This CD includes the licensed documents in both BookManager and PDF format.
System Center Publication IBM S/390® Redbooks™ Collection	SK2T-2177	This collection contains over 300 ITSO redbooks that apply to the S/390 platform and to host networking arranged into subject bookshelves.

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.

Title	Number
DNS and BIND, Fourth Edition, O'Reilly and Associates, 2001	ISBN 0-596-00158-4
Routing in the Internet , Christian Huitema (Prentice Hall PTR, 1995)	ISBN 0-13-132192-7
sendmail, Bryan Costales and Eric Allman, O'Reilly and Associates, 2002	ISBN 1-56592-839-3
SNA Formats	GA27-3136
TCP/IP Illustrated, Volume I: The Protocols, W. Richard Stevens, Addison-Wesley Publishing, 1994	ISBN 0-201-63346-9
TCP/IP Illustrated, Volume II: The Implementation, Gary R. Wright and W. Richard Stevens, Addison-Wesley Publishing, 1995	ISBN 0-201-63354-X
TCP/IP Illustrated, Volume III, W. Richard Stevens, Addison-Wesley Publishing, 1995	ISBN 0-201-63495-3
TCP/IP Tutorial and Technical Overview	GG24-3376
Understanding LDAP	SG24-4986
z/OS Cryptographic Service System Secure Sockets Layer Programming	SC24-5901
z/OS Integrated Security Services Firewall Technologies	SC24-5922
z/OS Integrated Security Services LDAP Client Programming	SC24-5924
z/OS Integrated Security Services LDAP Server Administration and Use	SC24-5923
z/OS JES2 Initialization and Tuning Guide	SA22-7532
z/OS MVS Diagnosis: Procedures	GA22-7587
z/OS MVS Diagnosis: Reference	GA22-7588
z/OS MVS Diagnosis: Tools and Service Aids	GA22-7589

Title	Number
z/OS MVS Using the Subsystem Interface	SA22-7642
z/OS Program Directory	GI10-0670
z/OS UNIX System Services Command Reference	SA22-7802
z/OS UNIX System Services Planning	GA22-7800
z/OS UNIX System Services Programming: Assembler Callable Services Reference	SA22-7803
z/OS UNIX System Services User's Guide	SA22-7801
z/OS XL C/C++ Run-Time Library Reference	SA22-7821
zSeries OSA-Express Customer's Guide and Reference	SA22-7935

Redbooks

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

Title	Number
| *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*	SG24-5227
| *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*	SG24-5228
| *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 4: Connectivity and Routing*	SG24-6516
| *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*	SG24-6840
IBM Communication Controller Migration Guide	SG24-6298
IP Network Design Guide	SG24-2580
Managing OS/390® TCP/IP with SNMP	SG24-5866
Migrating Subarea Networks to an IP Infrastructure	SG24-5957
| *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide: Volume 3: MVS Applications*	SG24-5229
Secureway Communications Server for OS/390 V2R8 TCP/IP: Guide to Enhancements	SG24–5631
SNA and TCP/IP Integration	SG24-5291
TCP/IP in a Sysplex	SG24-5235
TCP/IP Tutorial and Technical Overview	GG24-3376
| *Threadsafe Considerations for CICS*	SG24-6351

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/servers/eserver/zseries/zos/

z/OS Internet Library

| Use this site to view and download z/OS Communications Server
| documentation

http://www.ibm.com/servers/eserver/zseries/zos/bkserv/

IBM Communications Server product

The primary home page for information about z/OS Communications Server

http://www.software.ibm.com/network/commserver/

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

http://www.software.ibm.com/network/commserver/support/

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.

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.

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.

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS elements and features, z/VM®, VSE/ESA™, and Clusters for AIX® and Linux™:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX System Services).
- Your Microsoft® Windows® workstation. You can install code to access IBM message explanations on the *z/OS Collection* (SK3T-4269), using LookAt from a Microsoft Windows command prompt (also known as the DOS command line).
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from a disk on your *z/OS Collection* (SK3T-4269), or from the LookAt Web site (click **Download**, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Using 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. This book may refer to checks or messages associated with this component.

For additional information about checks and about IBM Health Checker for z/OS, see *IBM Health Checker for z/OS: User's Guide*. z/OS V1R4, V1R5, and V1R6 users can obtain the IBM Health Checker for z/OS from the z/OS Downloads page at http://www.ibm.com/servers/eserver/zseries/zos/downloads/.

SDSF also provides functions to simplify the management of checks. See *z/OS SDSF Operation and Customization* for additional information.

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:

- Go to the z/OS contact page at:

http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

There you will find the feedback page where you can enter and submit your comments.

- 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 you are commenting on (for example, a section number, a page number or a table number).

Summary of changes

Summary of changes
for GC31-8782-06
z/OS Version 1 Release 7

This document contains information previously presented in GC31-8782-05, which supports z/OS Version 1 Release 6.

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.

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.

New information

- Application Transparent Transport Layer Security (AT-TLS)
  - TCPIPCS TTLS subcommand, see "TCPIPCS TTLS" on page 253.
  - Chapter 28, "Diagnosing Application Transparent Transport Layer Security (AT-TLS)," on page 637
  - "Policy definition problems" on page 606
  - Policy Agent support, see "QoS policy" on page 601.
- File Transfer Protocol (FTP)
  - FTP security server enhancements, see "Server rejects password" on page 406.
- IP security
  - IPv4 Integrated IPSec/VPN support information, see "TCPIPCS IPSEC" on page 203, diagnostic steps in Chapter 4, "Diagnosing network connectivity problems," on page 27, Chapter 29, "Diagnosing IP security problems," on page 649, Chapter 8, "Diagnosing IKE daemon problems," on page 291, and Appendix C, "IKE protocol details," on page 809.
  - IKE daemon, see Chapter 8, "Diagnosing IKE daemon problems," on page 291 and Appendix C, "IKE protocol details," on page 809.
  - NAT Traversal support for Integrated IPSec/VPN traffic, see "TCPIPCS IPSEC" on page 203.
- HiperSockets
  - IPv6 support for HiperSockets, see "Packet trace (SYSTCPDA) for TCP/IP stacks" on page 86.
- Routing
  - Optimized Routing for sysplex distributor, see "Steps for diagnosing sysplex routing problems" on page 328.
- CTRACE optimization, see Table 11 on page 57.
- z/OS Load Balancing Advisor, see Chapter 7, "Diagnosing problems with the z/OS Load Balancing Advisor," on page 285.

Changed information

- CICS
  - CICS sockets enhancements, see Chapter 35, "Diagnosing problems with IP CICS sockets," on page 751.
- IP security
  - Diagnosing Policy Agent problems, see "Overview" on page 601.
  - Policy Agent support for AT-TLS, see "Policy definition problems" on page 606.
  - IPv4 integrated IPSec/VPN support information, see Table 11 on page 57.
  - Extensive additions for IPv4 Policy Agent support for IPSec, see Chapter 25, "Diagnosing Policy Agent problems," on page 601.
- QDIO OSA Express segmentation offload, see "Formatting packet traces using IPCS" on page 89.
- Promotion of the use of IP6 global unicast addresses

  Site-local addresses were designed to use private address prefixes that could be used within a site without the need for a global prefix. Until recently, the full negative impacts of site-local addresses in the Internet were not fully understood. The IETF has deprecated the special treatment given to the site-local prefix. Because of this, it is preferable to use global unicast addresses. This means we are replacing addresses and prefixes that use the site-local prefix (fec0::/10) with ones that use the global prefix for documentation (2001:0DB8::/32).

Deleted information

- OROUTED was removed from this release.

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 GC31-8782-05
z/OS Version 1 Release 6

This document contains information previously presented in GC31-8782-04, which supports z/OS Version 1 Release 5. The information in this document supports both IPv6 and IPv4. Unless explicitly noted, information describes IPv4 networking protocol. IPv6 support is qualified within the text.

New information:

- Diagnosing access control problems, see Chapter 10, "Diagnosing access control problems," on page 333.
- Debug level 16, see "Gathering diagnostic information" on page 603.
- Memory allocation/leakage problems section, see "Memory allocation/leakage problems" on page 615.

- OMPROUTE CTRACE with option DEBUGTRC description, see "Destination of OMPROUTE trace and debug output" on page 673.
- Commands and select examples and tasks are enabled for z/OS library center advanced searches.
- Packet trace variable on Options keyword, see "OPTIONS keywords" on page 94.

The following areas contain new information pertaining to IPv6 support:

- OSPF support for OMPROUTE, see Chapter 30, "Diagnosing OMPROUTE problems," on page 665, and "Starting OMPROUTE tracing and debugging from the z/OS UNIX System Services shell" on page 671.
- OMPROUTE trace option, OPACKET, see Table 61 on page 687.
- Internetworking overview, see "Dynamic routing" on page 807.
- New step in diagnosing SMTP delivery problems, see "SMTP does not deliver mail" on page 453.

Changed information:

- Sample output of the TCPIPCS PROFILE subcommand, see "Sample output of the TCPIPCS PROFILE subcommand" on page 213.
- VIPA DISTRIBUTE example in netstat vipadcfg, see Figure 30 on page 313.
- netstat vipadcfg display update, see "Steps for diagnosing SYSPLEXPORTS problems" on page 323.
- Specifying options at initialization, see "Specifying trace options" on page 684.
- Support for job-specific source IP addressing, see Chapter 9, "Diagnosing dynamic VIPA and sysplex problems," on page 311.
- Promotion of the use of IPv6 global unicast addresses - Site-local addresses were designed to use private address prefixes that could be used within a site without the need for a global prefix. Until recently, the full negative impacts of site-local addresses in the Internet were not fully understood. The IETF has deprecated the special treatment given to the site-local prefix. Because of this, it is preferable to use global unicast addresses. This means we are replacing addresses and prefixes that use the site-local prefix (fec0::/10) with ones that use the global prefix for documentation (2001:0DB8::/32). Some samples and examples in this document might display site-local prefixes instead of the now preferred global unicast addresses.

The following areas contain changed information pertaining to IPv6 support:

- IPv6 support for sysplex enhancements, see Chapter 9, "Diagnosing dynamic VIPA and sysplex problems," on page 311.

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.

Starting with z/OS V1R4, 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 GC31-8782-04 z/OS Version 1 Release 5

This document contains information previously presented in GC31-8782-03, which supports z/OS Version 1 Release 4. The information in this document supports both IPv6 and IPv4. Unless explicitly noted, information describes IPv4 networking protocol. IPv6 support is qualified within the text.

New information:

- CTRACE command syntax, see "Formatting event trace records for TCP/IP stacks and Telnet" on page 63.
- MAXSOC value information for the SELECT example, see "Reading and interpreting the SOCKAPI trace option" on page 75.
- Diagnosis information for FTP common problems, see "Secure IPv4 FTP session cannot transfer data through an NAT firewall" on page 406.
- Message suppression, see "FTP Messages and FTP trace entries" on page 370.
- Common problems with the FTP client PASV and EPSV commands, see "PASV and EPSV commands fail because no PASSIVEDATAPORTS are available" on page 375.
- Message EZYFT47I warnings, see "Enabling or suppressing message EZYFT47I during startup" on page 404.
- Information about documenting FTP problems, see "Documenting FTP client problems" on page 410.
- Message EZA2589E, see "Diagnosing FTP connection and transfer failures with EZA2589E" on page 398.
- SNMP functional components definitions, see "TN3270 Telnet subagent" on page 555.
- Telnet subagent traces, see "Starting TN3270 Telnet subagent traces" on page 577.
- Network SLAPM2 Subagent traces, see the following topics:
  - Table 2 on page 8
  - "Subagents" on page 552
  - "Network SLAPM2 subagent" on page 555
  - "Problems connecting subagents to the SNMP agent" on page 558
  - "Incorrect output" on page 564
  - "Variable format incorrect" on page 567
  - "Variable value incorrect" on page 568
  - "No response from the SNMP agent" on page 570
  - "Gathering diagnostic information" on page 603
  - "Policies and RSVP processing" on page 619
  - "Starting Network SLAPM2 subagent traces" on page 577
  - "QoS policy scope" on page 602
- Packet_trace variable on Options keyword, see "OPTIONS keywords" on page 94.
- Packet tracing methods, see "Formatting packet trace using a batch job" on page 135.
- Component tracing methods, see "Formatting component traces using a batch job" on page 52.

The following areas contain new information pertaining to IPv6 support:

- Debug switch for sendmail, see Table 32 on page 463.

- Diagnostic aids for IPv6 support, see “Diagnostic aids for IPv6 support” on page 467.
- SNMP problem diagnosis, see “0.0.0.0 address in traps from the SNMP agent” on page 572.
- DEBUGTRC trace for OMPROUTE, see Table 61 on page 687.
- Support for OMPROUTE (RIP) - RIPng, see “Commands to enable, disable, and display the status of the OMPROUTE CTRACE” on page 688.

Changed information:

- The SELECT example, see “Reading and interpreting the SOCKAPI trace option” on page 75.
- TN3270 Telnet subagent traces, see Table 2 on page 8.
- TN3270 network management information, starting with “Management information base (MIB)” on page 551 and continuing through “No response from the SNMP agent” on page 570.
- Example for TCPIPCS TREE subcommand, see “Sample output of the TCPIPCS PROFILE subcommand” on page 213.
- Storage area dumps, see step 1 on page 24.

The following areas contain changed information pertaining to IPv6 support:

- z/OS UNIX sendmail syslog.log message sample, see “Additional diagnostic aids” on page 465.
- Traps not forwarded by trap forwarder daemon, see “Traps not forwarded by trap forwarder daemon” on page 573.
- Incorrect address in forwarded trap analysis, see “Incorrect address in forwarded trap” on page 573.
- Support for OMPROUTE (RIP) - RIPng, see sections starting with “Starting OMPROUTE tracing and debugging from the z/OS UNIX System Services shell” on page 671 and continuing through “TCP/IP services component trace for OMPROUTE” on page 684.

Moved information:

- Diagnosing Sysplex Distributor problems moved from Chapter 4, “Diagnosing network connectivity problems,” on page 27 to Chapter 9, “Diagnosing dynamic VIPA and sysplex problems,” on page 311.

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.

Starting with z/OS V1R4, 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 books.

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.

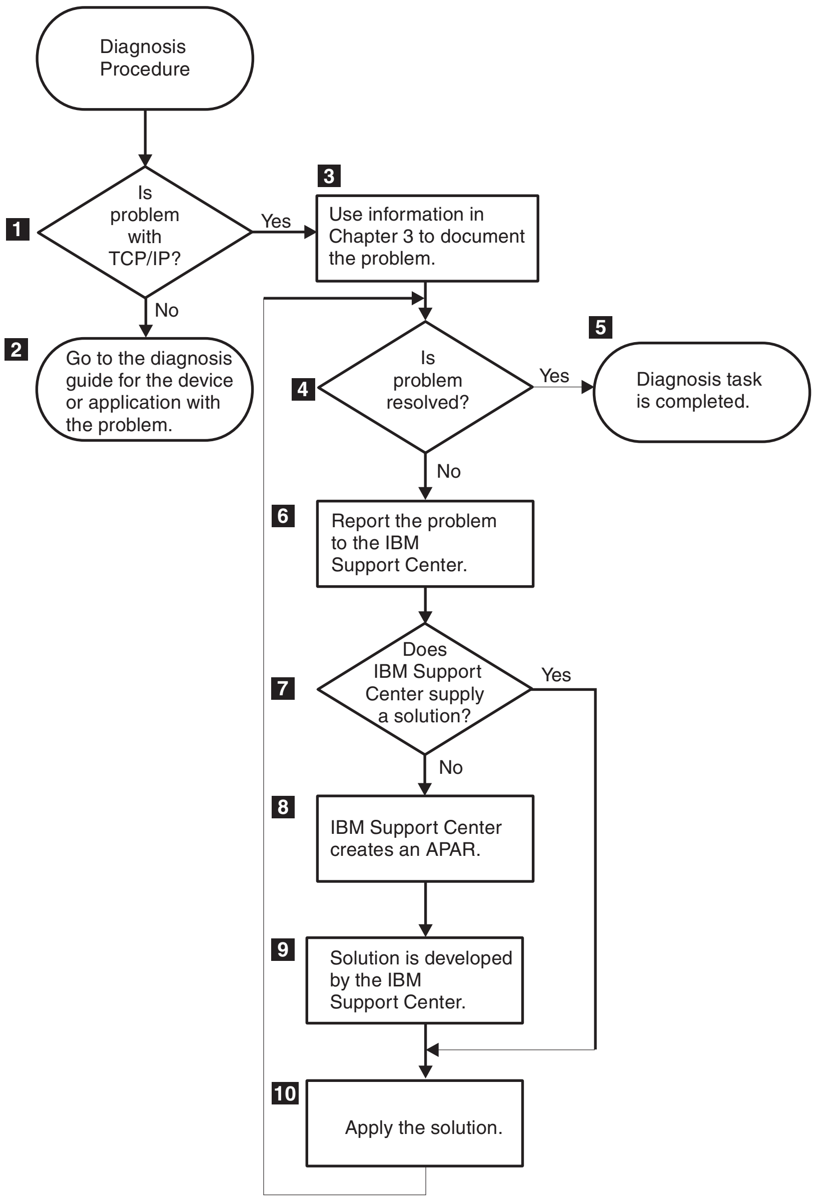

Diagnosis Procedure
1
Is problem with TCP/IP?
Yes
No
2
Go to the diagnosis guide for the device or application with the problem.
3
Use information in Chapter 3 to document the problem.
4
Is problem resolved?
Yes
5
Diagnosis task is completed.
No
6
Report the problem to the IBM Support Center.
7
Does IBM Support Center supply a solution?
Yes
No
8
IBM Support Center creates an APAR.
9
Solution is developed by the IBM Support Center.
10
Apply the solution.

Figure 1. Overview of the diagnosis 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 syslogd output 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/commserver/support/.

   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 **10**. If a solution to the problem is not found in the RETAIN database, the IBM Software Support Center representatives continues 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 chapter 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 topics are discussed in this chapter:
- “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 12 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 21 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:

Problem or need	See
Selecting a dump	Table 1 on page 8
Selecting a TCP/IP services component trace	Table 2 on page 8
Selecting a service aid	Table 3 on page 12

The tables show the problem, the corresponding tool or service aid, and the chapter 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

If the problem is ...	Then use this type of dump
Abnormal end of an authorized program or a problem program.	ABEND dump See "Analyzing abends" on page 23 for detailed information.
TCP/IP server or client address space stops processing or is stopped by the operator because of slowdown or looping condition.	SVC dump The SVC dump is created using the DUMP command. See "Analyzing loops" on page 24 for detailed information.

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

If the problem is ...	Then use this type of trace or command	Trace output location
Load balancing using the z/OS Load Balancing Advisor See Chapter 7, "Diagnosing problems with the z/OS Load Balancing Advisor," on page 285 for more information.	Log file	syslogd
Network connectivity See Chapter 4, "Diagnosing network connectivity problems," on page 27 for detailed information.	Ping, Netstat ARP/-R For information on Ping, see "Using the Ping command" on page 34. For information on Netstat ARP/-R, see "Netstat ARP/-R" on page 37.	Not applicable
	Packet trace See Chapter 5, "TCP/IP services traces and IPCS support," on page 41 for detailed information about packet trace.	CTRACE managed data set
Dynamic VIPA or Sysplex Distributor See Chapter 9, "Diagnosing dynamic VIPA and sysplex problems," on page 311 for detailed information.	Component Trace (SYSTCPIP) XCF option	TCP/IP address space or external writer
TCP/IP socket application See "Socket API traces" on page 67 for detailed information.	Component Trace (SYSTCPIP) SOCKAPI option	TCP/IP address space or external writer
LPR client See "LPR client traces" on page 341 for detailed information.	LPR command with the TRACE option	sysout

Table 2. Selecting a trace (continued)

If the problem is ...	Then use this type of trace or command	Trace output location
LPD server See "LPD server traces" on page 347 for detailed information.	See "LPD server traces" on page 347 for ways to activate traces.	SYSPRINT
z/OS UNIX FTP server See Chapter 12, "Diagnosing File Transfer Protocol (FTP) problems," on page 361 for detailed information.	z/OS UNIX FTP server trace	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.
z/OS UNIX Telnet See Chapter 13, "Diagnosing z/OS UNIX Telnet daemon (otelnetd) problems," on page 413, for detailed information.	z/OS UNIX Telnet traces	syslogd
Telnet See Chapter 14, "Diagnosing Telnet problems," on page 429 for detailed information.	Telnet traces	TCP/IP address space or external writer
SMTP See "SMTP RESOLVER trace" on page 459 for detailed information.	Resolver Trace (see also "Debugging with a resolver directive" on page 511)	Job log output
Popper See Chapter 16, "Diagnosing z/OS UNIX sendmail and popper problems," on page 463 for detailed information.	Popper Messages	syslogd
SNALINK LU0 See Chapter 17, "Diagnosing SNALINK LU0 problems," on page 473 for detailed information.	IP Packet Trace	CTRACE managed data set
	Debug Trace	SNALINK LU0 address space
SNALINK LU6.2 See Chapter 18, "Diagnosing SNALINK LU6.2 problems," on page 481 for detailed information.	TRACE DETAIL ALL	SYSPRINT
	IP Packet Trace	CTRACE managed data set
	TCP/IP Internal Trace CTRACE managed data set	CTRACE managed data set
	VTAM Buffer Trace	GTF managed data set, refer to *z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures* for detailed information.
Dynamic domain name system (DDNS) See Chapter 19, "Diagnosing name server and dynamic domain name server (DDNS) problems," on page 505 for detailed information.	Error messages	syslogd
	Resolver Trace	Job log output
	TCP/IP component trace	CTRACE managed data set

Table 2. Selecting a trace (continued)

If the problem is ...	Then use this type of trace or command	Trace output location
z/OS UNIX REXEC See Chapter 21, "Diagnosing z/OS UNIX REXEC, RSH, REXECD, and RSHD problems," on page 525.	z/OS UNIX REXEC debug trace	syslogd
z/OS UNIX REXECD See Chapter 21, "Diagnosing z/OS UNIX REXEC, RSH, REXECD, and RSHD problems," on page 525.	z/OS UNIX REXECD debug trace	syslogd
z/OS UNIX RSHD See Chapter 21, "Diagnosing z/OS UNIX REXEC, RSH, REXECD, and RSHD problems," on page 525.	z/OS UNIX RSHD debug trace	syslogd
Network database system (NDB) See Chapter 22, "Diagnosing network database system (NDB) problems," on page 531 for detailed information.	NDB Trace	Job log output
X Windows and Motif See Chapter 23, "Diagnosing X Window System and Motif problems," on page 547 for detailed information.	XWTRACE and XWTRACEC (environment variables)	stderr
SNMP See Chapter 24, "Diagnosing Simple Network Management Protocol (SNMP) problems," on page 551 for detailed information.	Manager Traces	Console (osnmp) or SYSPRINT (NetView® SNMP)
	• SNMP Agent Traces • TCP/IP Subagent Traces • OMPROUTE Subagent Traces • Network SLAPM2 Subagent Traces • SLA Subagent Traces • TN3270 Telnet Subagent Traces • TRAPFWD Traces	syslogd
Policy Agent See Chapter 25, "Diagnosing Policy Agent problems," on page 601 for detailed information.	Log file	Refer to the *z/OS Communications Server: IP Configuration Guide* for detailed information.
RSVP Agent See Chapter 26, "Diagnosing RSVP agent problems," on page 617 for detailed information.	Log file	Refer to the *z/OS Communications Server: IP Configuration Guide* for detailed information.
Traffic Regulator Management Daemon (TRMD) See Chapter 27, "Diagnosing intrusion detection problems," on page 631 for detailed information.	Log file	syslogd

Table 2. Selecting a trace (continued)

If the problem is ...	Then use this type of trace or command	Trace output location
OMPROUTE See Chapter 30, "Diagnosing OMPROUTE problems," on page 665.	Component Trace For detailed information about OMPROUTE Component Trace, see "TCP/IP services component trace for OMPROUTE" on page 684.	CTRACE managed data set
	OMPROUTE Trace For detailed information, see "OMPROUTE traces and debug information" on page 671.	stdout
NCPROUTE See Chapter 31, "Diagnosing NCPROUTE problems," on page 691 for detailed information.	NCPROUTE Traces	SYSPRINT
X.25 NPSI See Chapter 32, "Diagnosing X.25 NPSI problems," on page 717 for detailed information.	Server activity log	SYSPRINT
IMS See Chapter 33, "Diagnosing IMS problems," on page 727 for detailed information.	IP Packet Trace	CTRACE managed data set
	TCP/IP Internal Trace	CTRACE managed data set
	IMS Trace	Refer to the *IMS Version 8: Utilities Reference: System* for detailed information.
CICS See Chapter 35, "Diagnosing problems with IP CICS sockets," on page 751 for detailed information.	CICS® external trace data set (auxtrace)	Refer to the *CICS/ESA 5.2 Problem Determination Guide* for detailed information.
	TCP/IP Internal trace	CTRACE managed data set
Express Logon See Chapter 36, "Diagnosing problems with Express Logon," on page 757 for detailed information.	Log file	syslogd
Resolver See Chapter 37, "Diagnosing resolver problems," on page 761 for detailed information.	Trace Resolver	SYSPRINT or stdout
	Resolver Internal trace	CTRACE managed data set

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 on page 12.

Table 3. Selecting a service aid

If the problem is...	Then use this type of service aid
System or hardware problem: need a starting point for diagnosis or diagnosis requires an overview of system and hardware events in chronological order.	LOGREC data set or EREP Refer to *z/OS MVS Diagnosis: Tools and Service Aids* for detailed information.
Information about the contents of load modules and program objects or a problem with modules on the system.	AMBLIST Refer to *z/OS MVS Diagnosis: Tools and Service Aids* for detailed information.
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.	Service Level Indication Processing (SLIP) Refer to *z/OS MVS System Commands* for detailed information.
Diagnosis requires formatted output of problem data, such as a dump or trace.	IPCS Refer to *z/OS MVS IPCS User's Guide* for detailed information.

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 chapter 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 on page 13
- Traces, see Table 5 on page 14
- First Failure Support Technology™, see "First Failure Support Technology (FFST)" on page 15
- Display commands, see "Display commands" on page 17
- System service aids, see Table 6 on page 17

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 on page 13 describes the types of available dumps.

Table 4. Description of dumps

Type of dump	Description
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.
SVC dumps	SVC dumps can be used in two different ways: • Most commonly, a system component requests an SVC dump when an unexpected system error occurs, but the system can continue processing. • An authorized program or the operator can also request an SVC dump when diagnostic data is needed to solve a problem. SVC dumps contain a summary dump, control blocks, and other system code, but the exact areas dumped depend on whether the dump was requested by a macro, command, or SLIP trap. SVC dumps can be analyzed using IPCS. **Reference:** Refer to *z/OS MVS Diagnosis: Tools and Service Aids* for detailed information. If a console dump or SLIP is requested: • Capture the OMVS and (if applicable) affected application address spaces as well as TCP/IP. • Include all TCP/IP data spaces. • SDATA specification should contain the RGN, TRT, PSA, SUM, CSA and SQA keywords (at minimum).
FFST dumps	FFST dumps fall into two categories: SDUMPs (full dumps) and FFST™ minidumps (partial dumps). The type of dump produced depends on the characteristics of the probe that produced it. • FFST uses the operating system SDUMP macroinstruction to provide a full dump of the address space where the problem occurred. • 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 data sets that were allocated when FFST was installed.

Table 4. Description of dumps (continued)

Type of dump	Description
Stand-alone dumps	Use a stand-alone dump when: • The system stops processing. • The system enters a wait state with or without a wait state code. • The system enters an instruction loop. • The system is processing slowly. These dumps show central storage and some paged-out virtual storage occupied by the system or stand-alone dump program that failed. Stand-alone dumps can be analyzed using IPCS. See "Analyzing loops" on page 24 for detailed information.

Traces

Table 5 describes the types of available traces.

Table 5. Description of traces

Type of trace	Description
Component trace	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. **Reference:** See Chapter 5, "TCP/IP services traces and IPCS support," on page 41 for detailed information.
Data trace	Use a data trace to trace socket data (transforms) into and out of the physical file structure (PFS). **Reference:** See "Data trace (SYSTCPDA) for TCP/IP stacks" on page 135 for detailed information.
GTF trace	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. **Reference:** Refer to *z/OS MVS Diagnosis: Tools and Service Aids* for more information about GTF.
Master trace	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. **Reference:** Refer to *z/OS MVS Diagnosis: Tools and Service Aids* for detailed information.

Table 5. Description of traces (continued)

Type of trace	Description
Packet trace	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. **Reference:** See Chapter 5, "TCP/IP services traces and IPCS support," on page 41 for detailed information.
System trace	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. **Reference:** Refer to *z/OS MVS Diagnosis: Tools and Service Aids* for detailed information.
VTAM trace	z/OS Communications Server uses two VTAM components, CSM and MPC. VTAM traces contain entries for many TCP/IP events, especially I/O and storage requests. **Reference:** Refer to *z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT* for detailed information.
z/OS UNIX applications	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 chapters in this manual. | ITRACE initiated from TCPIP PROFILE processing **Reference:** Refer to the *z/OS Communications Server: IP Configuration Guide* for more detailed information about syslogd.

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 783 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 16 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.

```
EPW0401I FFST390: EVENT DETECTED BY TCP FOR PROBEID EZBXFC05
EPW0406I DUMP DATASET IS: SYSTEM DUMP DATA SET
EPW0402I PRIMARY SYMPTOM STRING FOR PROBEID EZBXFC05 FOLLOWS:
EPW0404I PIDS/5655HAL00 LVLS/50A PCSS/EZBXFC05 PCSS/FULL
EPW0404I RIDS/EZBXFMSO
EPW0701I END OF MESSAGE GROUP
```

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 6. Description of service aids

Type of service aid	**Description**
AMBLIST	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.

Table 6. Description of service aids (continued)

Type of service aid	**Description**
Common storage tracking	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 chapter of the *z/OS MVS IPCS User's Guide* for information about the IPCS VERBEXIT VSMDATA subcommand.
Dump suppression	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. **Tip:** Consider the SUPPRESSALL statement in ADYSETxx, if dumps are to be considered for suppression. Do this because the Communications Server IP Recovery Routine does not specify the VRADAE Key in the SDWA(system diagnostic work area) when requesting a dump. Refer to the *z/OS MVS Initialization and Tuning Guide* for more information about requesting dump suppression.
IPCS	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. **Reference:** Refer to the *z/OS MVS IPCS User's Guide* for detailed information.
LOGREC data set	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. **Reference:** Refer to the *z/OS MVS Diagnosis: Tools and Service Aids* for detailed information.
SLIP traps	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: • Requesting or suppressing a dump • Writing a trace or a LOGREC data set record • Giving control to a recovery routine • Putting the system in a wait state **Reference:** Refer to the SLIP command in *z/OS MVS System Commands* for detailed information.

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 (V TCPIP,,PKTTRACE) command can be RACF® protected.

Guidelines: When preparing documentation on tape for submission in an MVS environment, the 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 TRSMAIN (MVS terse) program, and send the data in BINARY mode.

Requirement: TRSMAIN 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:

1. FTP to the Web site at *ftp://service.software.ibm.com*.
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.

Steps for obtaining TRSMAIN

Perform the following steps to obtain TRSMAIN and detailed instructions on its use:

1. FTP to the Web site at *ftp://service.software.ibm.com*.
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/packlib/`, where you find two files: README.TXT and TRSMAIN.
4. Read the README file for detailed instructions.

If you require any additional directions, call the IBM Support Center.

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 TRSMAIN utility (see above) 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 22 lists the specific information that you should provide to the IBM Support Center.

Table 7. TCP/IP component name and release level

Component name and release level	System maintenance program	Field maintenance identifier/CLC
z/OS Communications Server V1R7	SMP/E	The following identifiers are associated with this stack: • HIP6170 (Base) • JIP6179 (HFS) • JIP617K (Security Level 3) • JIP617X (X-Window System X11R4)

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 chapter contains information about abends, loops, and hangs. More information is given in the individual component chapters in this document.

This chapter contains the following sections:

- "Analyzing abends"
- "Analyzing loops" on page 24
- "Steps for analyzing hangs" on page 25
-

Analyzing abends

An abend is an abnormal end.

Table 8 describes the types of abends that can occur.

Table 8. Types of abends

Type of abend	Description	Where to find help
User Abends	User abends are generated by C run-time routines. They usually start with U409x.	Refer to the *IBM Open Class Library Transition Guide* and *z/OS Communications Server: IP and SNA Codes*.
Platform abends	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.	For further assistance, call the IBM Support Center.
System abends	0C4, 0C1, and 878 are system abends.	Refer to the *z/OS MVS System Codes*.
	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.	Refer to the *z/OS MVS System Codes*. 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.
CEEDUMPs	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.	Refer to the *z/OS Language Environment Debugging Guide* publication.

A dump is usually produced when TCP/IP or a TCP/IP component address space abends. 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, 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 MVS Diagnosis: Procedures*.

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 41.
   - **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.

```
02-0029 008E7220   CLKC      078D2600 83A8192C  00001004  00000000

02-0029 008E7220   CLKC      078D2600 83A81934  00001004  00000000

02-0029 008E7220   CLKC      078D2600 83A81930  00001004  00000000

02-0029 008E7220   CLKC      078D2600 83A8192A  00001004  00000000

02-0029 008E7220   CLKC      078D2600 83A81930  00001004  00000000

02-0029 008E7220   CLKC      078D2600 83A81938  00001004  00000000
```

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

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 793.

Isolating network problems is an essential step in successful implementation of a network application. This chapter 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 hierarchical file system (HFS) are written in lowercase (for example, /etc/hosts).

Table 9 lists the name of the commands in each environment.

Table 9. Diagnostic commands

UNIX command	TSO command	Refer to:
ping/oping	PING	“Using the Ping command” on page 34

Table 9. Diagnostic commands (continued)

UNIX command	TSO command	Refer to:
netstat/onetstat	NETSTAT	"Using the Netstat command" on page 36
traceroute/otracert	TRACERTE	"Using the Traceroute command" on page 37

Guideline: The RESOLVER and NAMESERVER functions, which translate symbolic names to IP addresses, should be avoided 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 z/OS UNIX **netstat** command parameter, TCP, or the UNIX **netstat** command parameter, -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 the *z/OS Communications Server: IP Configuration Reference* for information about search order.

For more information about running multiple stacks, refer to *z/OS Communications Server: IP Configuration Guide*.

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 29 for more information.

3. Verify IP routing to the server or the client. See "Steps for verifying IP routing to a destination" on page 30 for more information.

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 33 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 650 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.

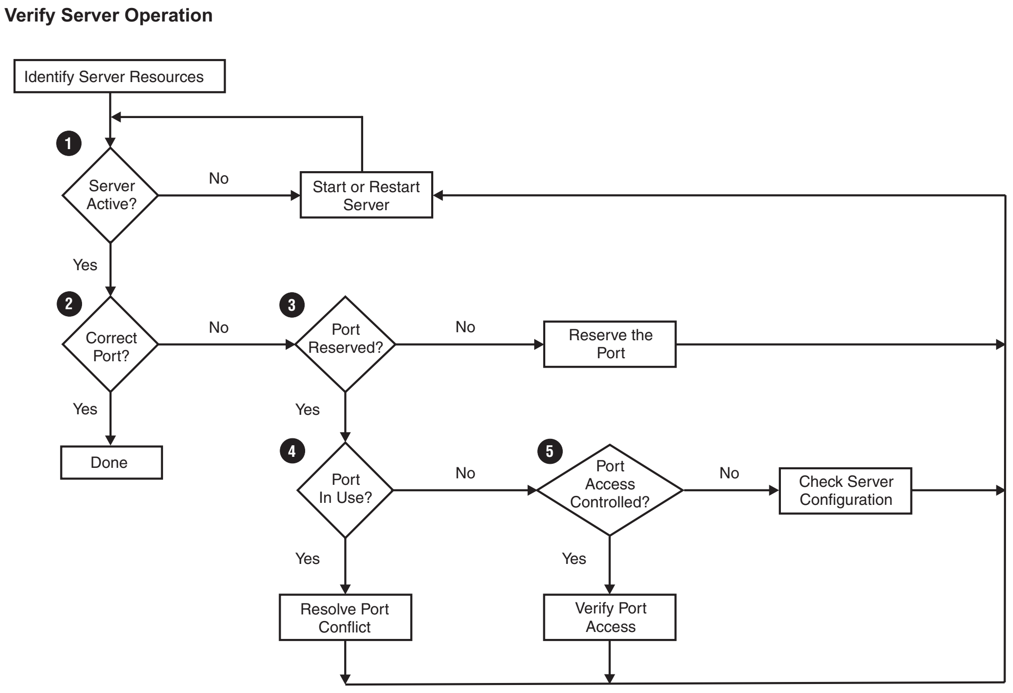


Figure 3. Overview of verifying server operation

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). 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*. For example:

```
NETSTAT SOCKETS (CLIENT SMTP
```

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

Figure 3 on page 29 shows the decisions involved for verifying IP routing to a destination.

Verify IP Routing to a Destination

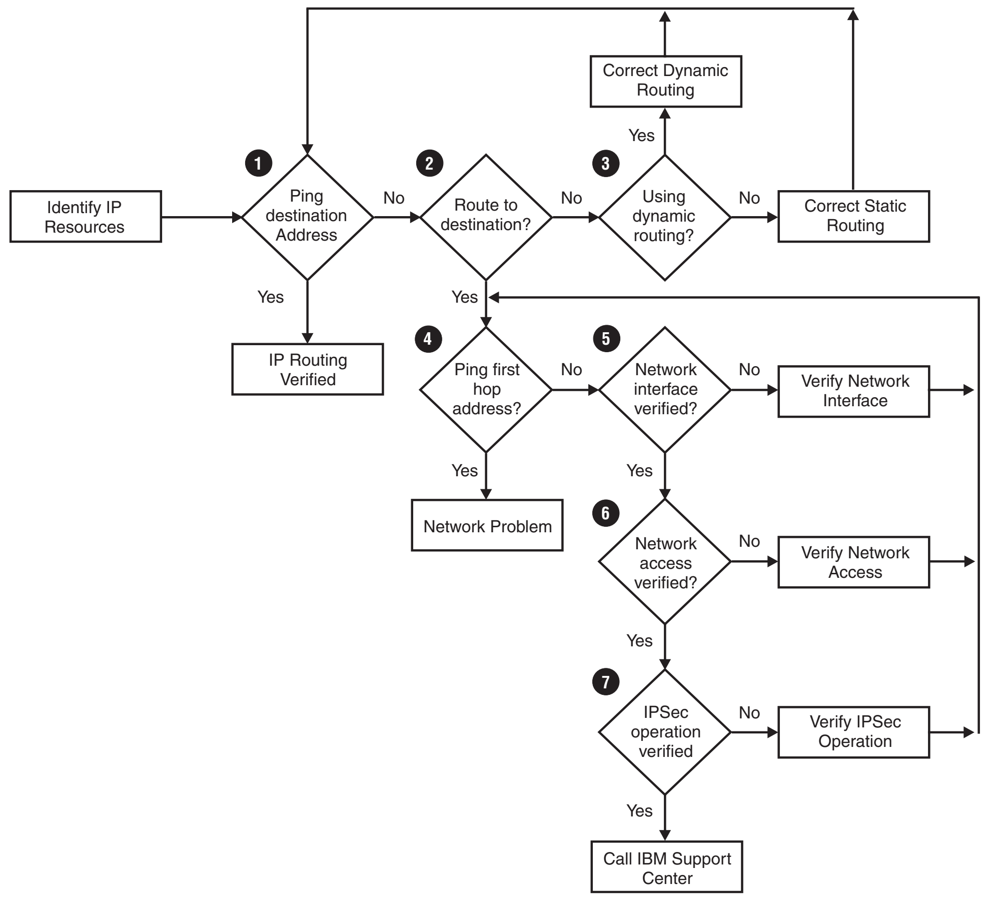


Figure 4. Overview of verifying IP routing to a destination

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 *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 *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 *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" 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 33 for more information.

---

7. Use the Netstat CONFIG/-f command to determine whether IP address security is enabled. If the output report field, FireWall, contains the value, `Yes`, then IP filtering is enabled. See *z/OS Integrated Security Services Firewall Technologies* for information on how to verify IP Filtering is correctly configured. If the output report field, IpSecurity, contains the value, `Yes`, then IP security is enabled. See "Steps for verifying IP security operation" on page 651 for information on how to that verify that IP security is correctly configured. If the problem still exists, see "Documentation for the IBM Support Center" on page 38 to understand what problem documentation should be gathered, and then call the IBM Support Center.

Steps for verifying network interface operation

Figure 5 on page 33 shows the decisions involved for verifying network interface operation.

Verify Network Interface Operation

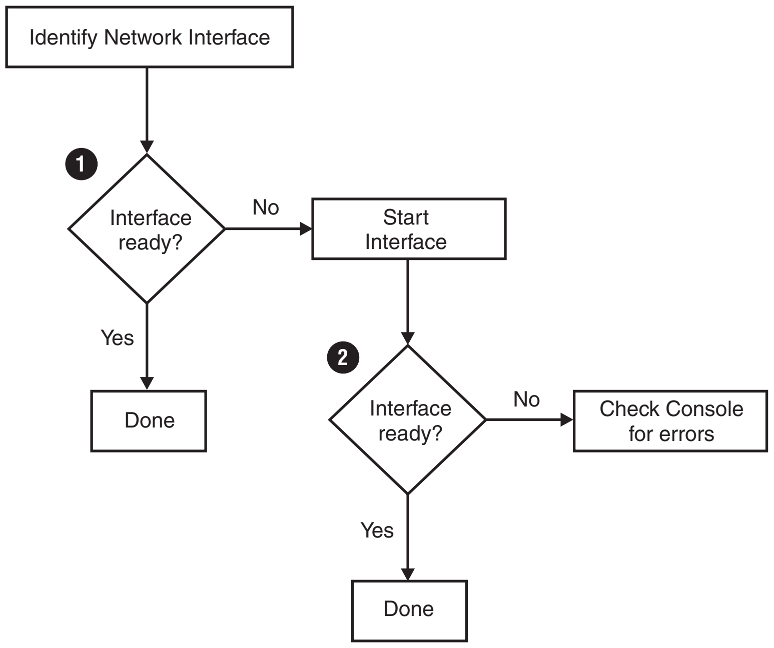


Figure 5. Overview of verifying network interface operation

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 34 shows the decisions involved for verifying network access.

Verify Network Access

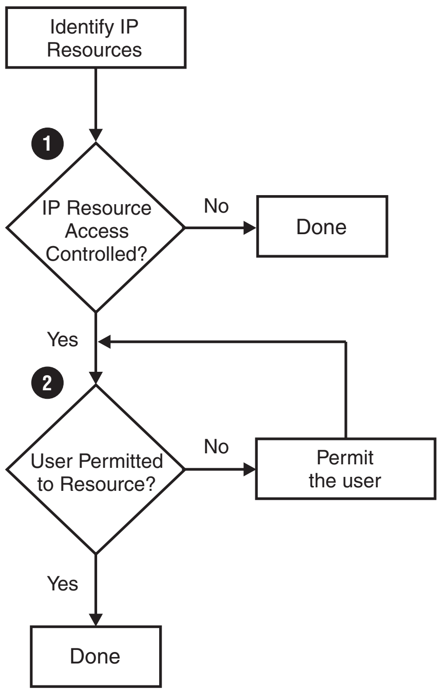


Figure 6. Overview of verifying network access

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 10, "Diagnosing access control problems," on page 333 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) Echo Request to a host, gateway, or router with the expectation of receiving a reply. The Ping function can be invoked by using the TSO Ping command or the z/OS UNIX shell ping/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 sequence number (icmp_seq) to correlate requests with replies. Instead, it uses icmp_id plus an 8-byte TOD time stamp field to correlate requests with replies. The TOD time stamp is the first 8-bytes in the icmp_data field.

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 examples 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. Whenever applicable, use this option to test connectivity.
  - 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 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.

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 on page 36.

Table 10. Diagnosis of a timeout

If the problem is ...	Then use these diagnostic techniques
The device is not transmitting packets to the local network.	Use **Netstat DEVLINKS/-d** to collect information to help you diagnose the problem. (See DEVLINKS/-d report option in *z/OS Communications Server: IP System Administrator's Commands*.)
The remote host is not receiving or transmitting packets on the network.	**IPv4-only stack:** Use Netstat ARP/-R to display the entry for the remote host. (See the ARP/-R report option in *z/OS Communications Server: IP System Administrator's Commands*.) **IPv6-enabled stack:** Use Netstat ND/-n to display the entry for the remote host. (See the ND/-n report option in *z/OS Communications Server: IP System Administrator's Commands*).
The remote host does not have a route back to the local z/OS server.	Use Netstat ROUTE/-r on the remote host to make sure it has a route back. (See ROUTE/-r report option in *z/OS Communications Server: IP System Administrator's Commands*.)
An intermediate router or gateway is not correctly forwarding IP packets.	Use a packet trace. (See Chapter 5, "TCP/IP services traces and IPCS support," on page 41.)
The IP reassembly timeout value might be set too low.	Refer to the TCP/IP Profile statements, IPCONFIG and IPCONFIG6, in *z/OS Communications Server: IP Configuration Reference*.

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 on page 37
- Netstat DEVLINKS/-d on page 37
- Netstat ROUTE/-r on page 37
- Netstat ARP/-R on page 37
- Netstat ND/-n on page 37

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 ADDRESS and LINK values returned by onetstat. Be sure that correct values are coded on 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.

If you note any errors, check *hlq*.PROFILE.TCPIP for the following:

- Make sure no statements were flagged in either the initial profile or in 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*.)
- Make sure the HOME and INTERFACE statements have been coded correctly.
- If static routing is provided by way of the BEGINROUTES statement, make sure the entries in the statement correlate to a valid interface name.
- If static routing is provided by way of the BEGINROUTES statement, ensure there is a ROUTE entry that correlates to the NETWORK or 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. Whenever applicable, use this to test connectivity.

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 to contacting the IBM Support Center:
- PROFILE.TCPIP
- TCPIP.DATA
- Output from Netstat commands
- Output from Netstat ROUTE/-r or Ping traces
- Network diagram or layout
- Error messages received Refer to *z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)* for information about messages.
- Component traces, see Chapter 5, “TCP/IP services traces and IPCS support,” on page 41
- If using dynamic routing protocols for IP route table management, see the following for related information:
  – Chapter 30, “Diagnosing OMPROUTE problems,” on page 665
  – Chapter 31, “Diagnosing NCPROUTE problems,” on page 691

Part 2. Traces and control blocks

Chapter 5. TCP/IP services traces and IPCS support

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

Telnet running in its own address space continues to use 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 books:

- *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.

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, and SYSTCPIS components.

Modifying with the PARMLIB member

Because TCPIP, OMPROUTE, RESOLVER, IKE daemon, 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 86).

SYSTCPIK
IKE daemon, parmlib = CTIKE00 (see "TCP/IP services component trace for the IKE daemon" on page 304).

SYSTCPIP
TCP/IP event trace, parmlib = CTIEZB*xx*, where xx is any 2 alphanumeric characters (see "Event trace (SYSTCPIP) for TCP/IP stacks and Telnet" on page 52).

SYSTCPIS
TCP/IP intrusion detection service, parmlib = CTIIDS*xx* (see "Intrusion Detection Services trace (SYSTCPIS)" on page 136).

SYSTCPRE
Resolver, parmlib = CTIRES*xx*, (see Chapter 37, "Diagnosing resolver problems," on page 761).

SYSTCPRT
OMPROUTE, parmlib = CTIORA00 (see "TCP/IP services component trace for OMPROUTE" on page 684).

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

SUB
Indicates the started procedure name for TCP/IP, the OMPROUTE application, the RESOLVER, IKE daemon started task name, started task name, or the Telnet started task name for which the trace is run. If you use the *S procname.jobname* method of starting TCP/IP, OMPROUTE, IKE daemon, or Telnet, the value specified for *jobname* must be the same as that for the SUB parameter. 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.

PARM
: Identifies the PARMLIB member containing the trace options (see "COMP" on page 42). All options can be respecified. However, the buffer size cannot be changed during the execution of OMPROUTE, IKE daemon, or the Resolver. If a different size is required, you must stop OMPROUTE, IKE daemon, 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 CTIEZB*xx* member is specified on the TRACE CT,ON command:

  ```
  IEE538I CTIEZBxx MEMBER NOT FOUND IN SYS1.PARMLIB
  ITT010I COMPONENT TRACE PROCESSING FAILED FOR PARMLIB MEMBER=CTIEZBxx:
  PARMLIB MEMBER NOT FOUND.
  ```
- An incorrect CTIEZB*xx* member is specified on the CTRACE() keyword of the EXEC statement of the TCP/IP started procedure:

  ```
  IEE538I CTIEZBxx MEMBER NOT FOUND IN SYS1.PARMLIB
  ```
- An incorrect CTIORA*xx* member is specified on the TRACE CT,ON command:

  ```
  IEE5381 CTIORAxx MEMBER NOT FOUND in SYS1.PARMLIB
  ITT01011 COMPONENT TRACE PROCESSING FAILED FOR PARMLIB MEMBER=CTIORAxx:
  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 ITT066A 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 chapter; options valid for use with OMPROUTE are listed in Chapter 30, “Diagnosing OMPROUTE problems,” on page 665; and options for SYSTCPRE (the Resolver component) are listed in Chapter 37, “Diagnosing resolver problems,” on page 761.

Options valid for use with IKE daemon are listed in Chapter 8, “Diagnosing IKE daemon problems,” on page 291

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 46.)

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 page “COMP” on page 42 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 4 on page 47, 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.

Stopping a component trace

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

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

See page “COMP” on page 42 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)
    R n,OPTIONS=(NONE),END
```

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

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 dataspace. TCPIPDS1 is the name of the dataspace for each TCP/IP in an MVS image. It contains the trace data for the SYSTCPIP, SYSTCPDA, and SYSTCPIS components.

To view the trace records for a problem where no abend has occurred, use the DUMP command. The following example illustrates an 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 DCB parameters. 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 48 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
 --------------------------------------------------------------
  TCPCS2              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.

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 11 on page 49), as shown below.

```
ITTPC503----------CTRACE DISPLAY PARAMETERS--------------------------

System      =======>           (System name or blank)
Component   =======>           (Component name (required))
Subnames    =======>

GMT/LOCAL   =======>            (Greenwich Mean Time or Local; GMT is default)
Start time  =======>            (mm/dd/yy,hh:mm:ss.dddddd)
Stop time   =======>
Limit       =======>      Exception  =======>
Report type =======> FULL       (SHort, SUmmary, Full, Tally)
User exit   =======>            (Exit program name)
Override source =======>
Options     =======>

To enter/verify required values, type any character
Entry IDS =======>     Jobnames =======>     ASIDs =======>     OPTIONS =======>     SUBS =======>

CTRACE COMP(xx) FULL

COMMAND =======>
     F1=Help  F2=Split  F3=End    F4=RETURN    F5=RFIND    F6=MORE    F7=UP
     F8=DOWN  F9=Swap  F10=LEFT  F11=RIGHT    F12=CURSOR
```

Figure 11. IPCS CTRACE

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 42.
- SYSTCPIK, see "TCP/IP services component trace for the IKE daemon" on page 304.
- SYSTCPIP, see "COMP" on page 42.
- SYSTCPIS, see "COMP" on page 42.
- SYSTCPRE, see Chapter 37, "Diagnosing resolver problems," on page 761.
- SYSTCPRT, see "TCP/IP services component trace for OMPROUTE" on page 684.

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.dddddd)──STOP(mm/dd/yy,hh.mm.ss.dddddd)──EXCEPTION──────►

►─LIMIT(nnnnnnnn)──ENTIDLIST(entidlist)──USEREXIT(exitname)────────────────────────►

►─OPTIONS((component routine parameters))─────────────────────────────────────────┤
```

Address Space Selection:

```
├───┬─ALL────────────────────┬─────────────────────────────────────────────┤
    ├─CURRENT────────────────┤
    ├─ERROR──────────────────┤
    ├─TCPERROR───────────────┤
    ├─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 formaters:

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 formaters, except for the packet trace formaters. 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 89.

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

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

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

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

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

 COMP(SYSTCPIP)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,
                      OPMSGS,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:

```
//jobname   DD (accounting),pgmname,CLASS=A,MSGCLASS=A
//DUMP    EXEC PGM=IKJEFT01
//STEPLIB DD DISP=SHR,DSN=SYS1.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
//IPCSPRNT  DD SYSOUT=*
//IPCSTOC   DD SYSOUT=*
//IPCSDDIR  DD DISP=SHR,DSN=userid.IPCS.DMPDIR
//SYSTSIN   DD *
 IPCS NOPARM
 SETDEF DA('ctrace.dataset')
 CTRACE COMP(SYSTCPIP) SUBNAME((tcpiprocname)) OPTIONS((systcpip_options_string)) +
  FULL LOCAL
 END
/*

Note: IPCSPARM DD should be modified as follows:
//IPCSPARM DD DISP=SHR,DSN=SYS1.PARMLIB
//         DD DISP=SHR,DSN=CPAC.PARMLIB
//         DD DISP=SHR,DSN=SYS1.IBM.PARMLIB

These concatenations will be used to locate the BLSCECT member that
is required by IPCS
```

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 304.

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

The TN3270 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 Telnet server with the following exceptions:

- The 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, CTIEZB*xx*.

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(CTIEZB*xx*)

where CTIEZB*xx* 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 57.

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

```
/*******************************************************************/
/*                                                                 */
/*  IBM Communications Server for z/OS                             */
/*  SMP/E Distribution Name: CTIEZB00                              */
/*                                                                 */
/*  MEMBER:  CTIEZB00                                              */
/*                                                                 */
/*                                                                 */
/* Copyright:    Licensed Materials - Property of IBM              */
/*                                                                 */
/*               5694-A01                                          */
/*                                                                 */
/*               (C) Copyright IBM Corp. 1996, 2005.               */
/*                                                                 */
/*     STATUS =  CSV1R7                                            */
/*                                                                 */
/*  DESCRIPTION = This parmlib member causes component trace for   */
/*                the TCP/IP product to be initialized with a      */
/*                trace buffer size of 8 megabytes.                */
/*                                                                 */
/*                This parmlib member only lists those TRACEOPTS   */
/*                value specific to TCP/IP.  For a complete list   */
/*                of TRACEOPTS keywords and their values see       */
/*                z/OS MVS INITIALIZATION AND TUNING REFERENCE.    */
/*                                                                 */
/* $MAC(CTIEZB00) PROD(TCPIP): Component Trace SYS1.PARMLIB member  */
/*                                                                 */
/*******************************************************************/
 TRACEOPTS
 /* ---------------------------------------------------------------- */
 /*   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 ROUTE, SERIAL, SOCKAPI, STORAGE,  */
 /*             TIMER, PFSMIN and TCPMIN                             */
 /*   CSOCKET = PFS + SOCKET                                         */
```

Figure 12. SYS1.PARMLIB member CTIEZB00 (Part 1 of 3)

```
/*   DLC     = CLAW + INTERNET + LCS + VTAM + VTAMDATA              */
/*   IN      = CONFIG + INIT + IOCTL + OPCMDS + OPMSGS              */
/*   LATCH   = SERIAL                                               */
/*   MINIMUM = INIT + OPCMDS + OPMSGS                               */
/*   ALLMIN  = INIT + OPCMDS + OPMSGS + MINPFS + MINTCP             */
/*   OETCP   = ENGINE + PFS + QUEUE + TCP                           */
/*   OEUDP   = ENGINE + PFS + QUEUE + UDP                           */
/*   PING    = ARP + ICMP + RAW + ND                                */
/*   RW      = ENGINE + PFS + QUEUE + RAW + SOCKET                  */
/*   SMTP    = ENGINE + IOCTL + PASAPI + PFS + QUEUE + SOCKET + TCP */
/*   SYSTEM  = INIT + OPCMDS + OPMSGS + SERIAL + STORAGE + TIMER +  */
/*             WORKUNIT                                             */
/*   TC      = ENGINE + PFS + QUEUE + SOCKET + TCP                  */
/*   TN      = PFS + TCP + TELNET + TELNVTAM                        */
/*   UD      = ENGINE + PFS + QUEUE + SOCKET + UDP                  */
/*                                                                  */
/* ---------------------------------------------------------------- */
/*                                                                  */
/*   PFSMIN  = Reduced set of PFS trace data                        */
/*   TCPMIN  = Reduced set of TCP trace data                        */
/*   ALLMIN  = PFSMIN + TCPMIN                                      */
/*                                                                  */
/*                                                                  */
/*   NOTE:    The xxxMIN and the corresponding xxx options should   */
/*            not be active at the same time.  The will collect     */
/*            duplicate information.                                */
/*                                                                  */
/*   OPTIONS: NAMES OF FUNCTIONS OR GROUPS TO BE TRACED:            */
/*                                                                  */
/* ---------------------------------------------------------------- */
/*      OPTIONS(            */
/*              'ALL     '  */
/*             ,'ALLMIN  '  */
/*             ,'ACCESS  '  */
/*             ,'AFP     '  */
/*             ,'ARP     '  */
/*             ,'CLAW    '  */
/*             ,'CONFIG  '  */
/*             ,'CSOCKET '  */
/*             ,'DLC     '  */
/*             ,'EID(hhhhhhh,hhhhhhhh) '            */
/*             ,'ENGINE  '  */
/*             ,'FIREWALL'  */
/*             ,'ICMP    '  */
/*             ,'IN      '  */
/*             ,'INIT    '  */
/*             ,'INTERNET'  */
```

Figure 12. SYS1.PARMLIB member CTIEZB00 (Part 2 of 3)

```
/*                ,'IOCTL   '   */
/*                ,'IPADDR(nnn.nnn.nnn.nnn/mmm.mmm.mmm, */
/*                         nnn.nnn.nnn.nnn/pp,          */
/*                         hhhh::hhhh/ppp)              */
/*                ,'IPSEC   '   */
/*                ,'LATCH   '   */
/*                ,'LCS     '   */
/*                ,'MESSAGE '   */
/*                ,'MINIMUM '   */
/*                ,'MISC    '   */
/*                ,'ND      '   */
/*                ,'NONE    '   */
/*                ,'OETCP   '   */
/*                ,'OEUDP   '   */
/*                ,'OPCMDS  '   */
/*                ,'OPMSGS  '   */
/*                ,'PASAPI  '   */
/*                ,'PFS     '   */
/*                ,'PFSMIN  '   */
/*                ,'PING    '   */
/*                ,'POLICY  '   */
/*                ,'PORT(ppppp,ooooo,rrrrr,ttttt) ' */
/*                ,'QUEUE   '   */
/*                ,'RAW     '   */
/*                ,'ROUTE   '   */
/*                ,'RW      '   */
/*                ,'SERIAL  '   */
/*                ,'SMTP    '   */
/*                ,'SNMP    '   */
/*                ,'SOCKAPI '   */
/*                ,'SOCKET  '   */
/*                ,'STORAGE '   */
/*                ,'SYSTEM  '   */
/*                ,'TC      '   */
/*                ,'TCP     '   */
/*                ,'TCPMIN  '   */
/*                ,'TELNET  '   */
/*                ,'TELNVTAM'   */
/*                ,'TIMER   '   */
/*                ,'TN      '   */
/*                ,'UD      '   */
/*                ,'UDP     '   */
/*                ,'VTAM    '   */
/*                ,'VTAMDATA'   */
/*                ,'WORKUNIT'   */
/*                ,'XCF     '   */
/*                 )            */
```

Figure 12. SYS1.PARMLIB member CTIEZB00 (Part 3 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 11. Trace options and groups

Trace Event	Description
ALL	All types of records except ROUTE, SERIAL, STORAGE, and TIMER. **Slow Performance:** Using this option slows performance considerably, so use with caution. Also available for stand-alone Telnet.
ALLMIN	Turns on the following trace options: • INIT • OPCMDS • OPMSGS • 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.

Table 11. Trace options and groups (continued)

Trace Event	Description
FIREWALL	Turns on trace for firewall events. **Tip:** Synonymous with IPSEC option.
ICMP	Turns on trace for the ICMP protocol.
IN	Turns on the following trace options: • CONFIG • INIT • IOCTL • OPCMDS • OPMSGS
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 stand-alone Telnet.
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 stand-alone Telnet.
MINIMUM	Turns on the following trace options: • INIT • OPCMDS • OPMSGS
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 stand-alone Telnet.

Table 11. Trace options and groups (continued)

Trace Event	Description
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.
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.
OPMSGS	Turns on message 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.

Table 11. Trace options and groups (continued)

Trace Event	Description
QUEUE	Turns on trace for stream queue 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.
RAW	Turns on trace for the RAW transport protocol.
ROUTE	Trace manipulation of IP Routing Tree.
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 stand-alone Telnet.
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 67)
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 stand-alone Telnet.
SYSTEM	Turns on the following trace options: • INIT • OPCMDS • OPMSGS • SERIAL • STORAGE • TIMER • WORKUNIT

Table 11. Trace options and groups (continued)

Trace Event	Description
TC	Turns on the following trace options: • ENGINE • PFS • QUEUE • SOCKET • TCP
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. Also available for stand-alone Telnet.
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 stand-alone Telnet.
TN	Turns on the following trace options: • PFS • TCP • TELNET • TELNVTAM
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 nondata-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.

Table 11. Trace options and groups (continued)

Trace Event	Description
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.

Additional filters for SYSTCPIP

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

- The IPADDR 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 identifer 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=(IPADDR(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 12. Data types

Data type	Description
Inbound	Data received at the IP layer is considered inbound data. The source IP address and the destination port number are used.
Outbound	Data sent in the PFS layer is considered outbound data. The destination IP address and the source port number are used.

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

- TYPE

- JOBNAME
- ASID
- IPADDR
- PORT

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 a 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=(TCPIP_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::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, 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

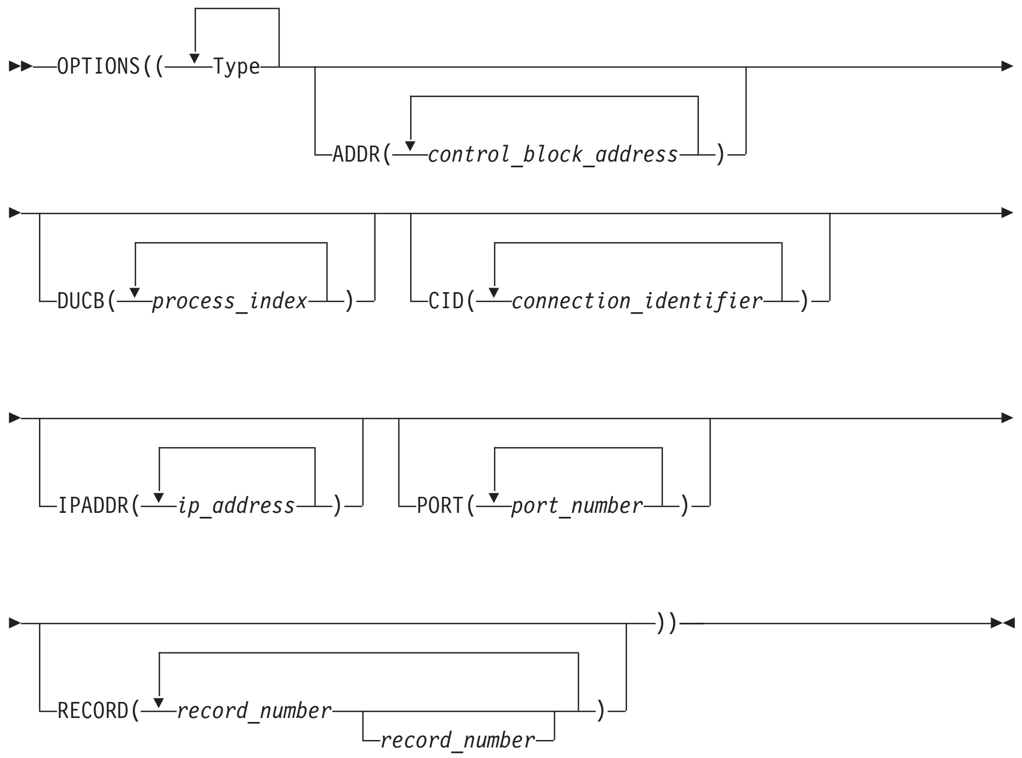
You can format event trace records using IPCS panels or a combination of IPCS panels and the CTRACE command. For a description of the relevant IPCS panels, see “Steps for formatting component traces using IPCS panels” on page 48.

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 57.

ADDR
A control block address. Up to 16 control block addresses can be specified. Addresses in hexadecimal should be entered as `x'hhhhhhhh'`.

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'hhhhhhh':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
     SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
     -------   --------  --------  --------------- -------------
1     VIC142    PFS       60010018  14:57:59.207826   Socket IOCTL Exit
2    HASID..001E      PASID..000E      SASID..001E      USER...OMPROUTE
3    TCB....007E7A68  MODID..EZBPFIOC  REG14..161D86C0  DUCB...0000000C
4    CID....0000003A  PORT.....0
   IPADDR. 3F98::D002:A521
5      ADDR...00000000  14D9EED0  LEN....000000A0  OSI
6         +0000  D6E2C940  000000A0  00000000  00000000  | OSI ........... |
          +0010  0500001B  14D9EF70  00500AC8  00000000  | .....R...&.H.... |
          +0020  00000000  00000000  00000000  00000000  | ................ |
          +0030  00000000  00000000  00000000  00281080  | ................ |
          +0040  14D9FC0C  00000C00  14D9FFE8  00000000  | .R.......R.Y.... |
          +0050  00000000  00000000  00000000  00000000  | ................ |
          +0060  00000000  00000000  00000000  00000000  | ................ |
          +0070  00000000  00000000  00000000  00000000  | ................ |
          +0080  00000000  00000000  00000000  00000000  | ................ |
          +0090  00000000  00000000  00000000  00000000  | ................ |
       ADDR...00000000  12D7F874  LEN....00000004  SCB Flags
          +0000  00280000                                | ....             |
       ADDR...00000000  12E88598  LEN....00000010  Return Value Errno ErrnoJr
          +0000  C5D9D9D5  FFFFFFFF  00000462  740E006B  | ERRN..........., |
       ERRNO..-1,     462,  740E006B
       ADDR...00000000  14D9F4E4  LEN....00000048  IOCTL Request
          +0000  C3C6C7D4  D9C5D840  0000008E  00000462  | CFGMREQ ........ |
          +0010  00000320  00000500  00000000  00000000  | ................ |
          +0020  740E0005  00000000  14B4C7C0  00000000  | ..........G{.... |
          +0030  00000000  00050063  00000000  00000000  | ................ |
          +0040  F3F1F0F1  00000000                      | 3101....         |
7    ==================================================================0000573E
```

Figure 13. Start of component trace full format

The parts of the TCP/IP CTRACE record are:

1 Standard IPCS header line, which includes the system name (VIC142), TCP/IP option name (PFS), time stamp, and record description.

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

3 TCP/IP header line with task and module information.

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

5 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).

6 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.

7 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 65 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.

```
 BOTSWANA  TCP       40030002  20:51:35.652462   Select/Poll Exit Detail
HASID..0082      PASID..0088      SASID..000E      USER...POLAGENT
TCB....007E4640  MODID..EZBTCFSP  REG14..10FD7C5E  DUCB...00000016
CID....000004DC  PORT....1925
IPADDR. 197.011.106.001
  ADDR...00000000  116B04DC  LEN....00000004  Select function code
     +0000  00000002                            | ....             |
  ADDR...00000000  116B0668  LEN....00000004  Output condition indicators
     +0000  40000000                            |  ...             |
  ADDR...00000000  7F60C508  LEN....000003D8  Transmission Control Block
     +0000  E3C3C240  C3E3D9D3  00050009  81801000  | TCB CTRL....a... |
     +0010  00000000  00000000  00000000  138C4F08  | .............|. |
      ...
     +0170  00000000  00020000  00003000  45000028  | ................ |
     +0180  1CB14000  40060000  C50B6A01  C50B6A01  | .. . ...E...E... |
     +0190  00000000  00000000  00000000  00000000  | ................ |
     +01A0  00000000  00000000  00000000  00000000  | ................ |
     +01B0  00000000  00000000  0000FFFF  FFFF4000  | ............. . |
     +01C0  00000000  00000000  00000000  00000001  | ................ |
     +01D0  07850185  F4258CA0  F425A310  50107F32  | .e.e4...4.t.&.". |
     +01E0  00000000  0004FFCB  01030300  0101080A  | ................ |
      ...
     +03D0  010E1301  0E21010E                      | ........         |
  TcpState..ESTAB     TpiState..WLOXFER
  SrcPort..1925       SrcIPAddr. 197.11.106.1
  DstPort..389        DstIPAddr. 197.11.106.1
  FLAGS.....ACK
```

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 67 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.

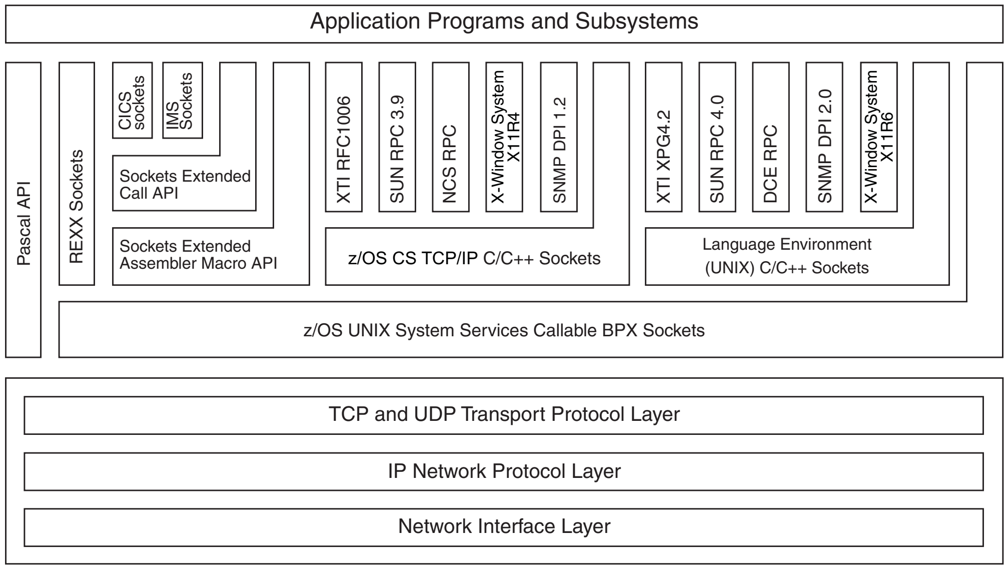

Application Programs and Subsystems
Pascal API
REXX Sockets
CICS sockets
IMS Sockets
Sockets Extended Call API
Sockets Extended Assembler Macro API
XTI RFC1006
SUN RPC 3.9
NCS RPC
X-Window System X11R4
SNMP DPI 1.2
z/OS CS TCP/IP C/C++ Sockets
XTI XPG4.2
SUN RPC 4.0
DCE RPC
SNMP DPI 2.0
X-Window System X11R6
Language Environment (UNIX) C/C++ Sockets
z/OS UNIX System Services Callable BPX Sockets
TCP and UDP Transport Protocol Layer
IP Network Protocol Layer
Network Interface Layer

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 41.

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 SYSTCPIS 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,JOBNAME=(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:

```
  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

Socket call	**Filtering active?**	
	Local port	**Remote IP address**
ACCEPT	Yes	No (1)
BIND	Yes/No (2)	No
CONNECT	Yes/No (3)	Yes

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

Socket call	Filtering active?	
	Local port	Remote IP address
CANCEL FREEADDRINFO GETADDRINFO GETCLIENTID GETHOSTBYADDR GETHOSTBYNAME GETHOSTID GETHOSTNAME GETNAMEINFO INITAPI RECVFROM RECVMSG SELECT SELECTEX SENDMSG SENDTO SOCKET TAKESOCKET TERMAPI	No	No
LISTEN	Yes	No
CLOSE GETPEERNAME GETSOCKNAME GETSOCKOPT GIVESOCKET FNCTL IOCTL READ READV RECV SHUTDOWN SEND SETSOCKOPT WRITE WRITEV	Yes	Yes

Where Yes is indicated in Table 13 on page 71, 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. Assumes 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  IEE843I 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.                           MODE BUFFER HEAD SUBS
 9.                           =====================
10.                           OFF        HEAD    1
11.          NO HEAD OPTIONS
12.       SUBTRACE            MODE BUFFER HEAD SUBS
13.      -----------------------------------------------------
14.       TCPIPROC            ON   0008M
15.          ASIDS      *NONE*
16.          JOBNAMES   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.

Line	**Description**
1	The MVS DISPLAY TRACE command. For more information on this command, see *z/OS MVS System Commands*.
2–4	These are explained in the *z/OS MVS System Messages, Vol 1 (ABA-AOM)* for IEE843I.
5–7	Show that this is the CTRACE component SYSTCPIP.
8–11	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).
14	Shows the ″subtrace″ 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.

15–16 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).

17 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.

18 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.

```
DUMP COMM=(Sample dump for SOCKAPI)
R n,JOBNAME=(tcpiproc,ezasokjs),DSPNAME=('tcpiproc'.TCPIPDS1),CONT
R n,SDATA=(ALLNUC,CSA,LPA,LSQA,RGN,SWA,SQA,TRT),CONT
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 70, 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 173 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(SYSTCPIP) SUB((TCPIPROC)) FULL` produced the following output for a dump:

```
COMPONENT TRACE QUERY SUMMARY

 COMP(SYSTCPIP)SUBNAME((TCPIPROC))

     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 77).

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.

```
 1.  ======================================================================00007FE8
 2.   MVS026    SOCKAPI   60050042  19:31:08.338135   READ Entry
 3.  HASID....0027     PASID....0027     SASID..0027     JOBNAME..EZASOKGS
 4.  TCB......006E6A68 TIE......00008DF8 PLIST..00008E0C DUCB.....0000000C KEY..8
 5.  ADSNAME..GTASOKGS SUBTASK..MACROGIV                 TOKEN....7F6F3798 09902FB0
 6.  LOCAL  PORT..12035      IPADDR.. 9.67.113.58
 7.  REMOTE PORT..1034       IPADDR.. F901::32E1
 8.   REQAREA..:  00008D90x                                          Addr..00008D90
 9.   SOCKET...:  1                                                  Addr..00008A38
10.   NBYTE....:  40                                                 Addr..00008A34
11.   ALET.....:  00000000x                                          Addr..000089A8
12.   BUF......:  (NO DATA)                                          Addr..000089A8
```

Line	Description
1	This separator line shows the previous SYSTCPIP component trace record number in hexadecimal.

2 The first data line has the host name (MVS026), trace option (SOCKAPI), trace code (60050042), time, and trace record name.

3 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.

4 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.

5 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.

6–7 If applicable, the ports and IP addresses are shown. The ports are formatted in decimal; the IP addresses are in dotted decimal.

8 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.

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

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

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

12 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 78
- "API call fails synchronously with parameter not addressable" on page 78
- "API call fails synchronously with diagnostic reason code" on page 79
- "Resolver API calls" on page 79
- "External IOCTL commands" on page 81
- "API call with an IOV parameter" on page 82
- "Default INITAPI" on page 82
- "Default TERMAPI" on page 82
- "SELECT" on page 82
- "SELECTEX" on page 83
- "Token Error" on page 84
- "Unsolicited event exit" on page 84

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.

```
==================================================================00007B01
 MVS026    SOCKAPI   60050012  19:27:08.111729   GETHOSTID Exit
HASID....0027     PASID....0027     SASID..0027     JOBNAME..EZASOKOS
TCB......006E6A68 TIE......00006DF8 PLIST..00006E0C DUCB.....0000000C KEY..8
ADSNAME..EZASOKOS SUBTASK..00000000 00000000        TOKEN....7F6F3798 09902FB0
 REQAREA..:  00006D90x                                          Addr..00006D90
 R15......:  0 (CALL ACCEPTED)
==================================================================00007B05
 MVS026    SOCKAPI   60050032  19:27:08.111741   GETHOSTID Async Complete
HASID....0027     PASID....0027     SASID..0027     JOBNAME..EZASOKOS
TCB......006E6A68 TIE......00006DF8 PLIST..00006E0C DUCB.....0000000C KEY..8
ADSNAME..EZASOKOS SUBTASK..00000000 00000000        TOKEN....7F6F3798 09902FB0
 REQAREA..:  00006D90x                                          Addr..00006D90
 R0.......:  0x (NORMAL RETURN)
 ASYNC PTR:  00006B1C
 EXIT TOKEN: 00006B98x
 EXIT KEY.:  8x
 AUTHORIZATION STATE: PROBLEM
 RETCODE..:  9.67.113.58     (HOST IP ADDRESS)                  Addr..00006EB4
```

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* for information about TCP/IP Sockets Extended Return Codes.

```
==================================================================00007B93
 MVS026    SOCKAPI   60050057  19:27:13.817195   WRITE Exit
HASID....0027     PASID....0027     SASID..0027     JOBNAME..EZASOKOS
TCB......006E6A68 TIE......00006DF8 PLIST..00006E0C DUCB.....00000009 KEY..8
ADSNAME..EZASOKOS SUBTASK..00000000 00000000        TOKEN....7F6F3798 09902FB0
LOCAL  PORT..11007      IPADDR.. 9.67.113.58
REMOTE PORT..1031       IPADDR.. 9.67.113.58
 REQAREA..:  00006D90x                                          Addr..00006D90
 SOCKET...:  1                                                  Addr..00006BDC
 R15......:  NON-ZERO (CALL WAS NOT ACCEPTED)
 ERRNO....:  10184 (EIBMWRITELENZERO)                           Addr..00006EB0
 RETCODE..:  -1                                                 Addr..00006EB4
```

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.

```
==================================================================00021347A
 VIC102    SOCKAPI   60050050  17:36:51.302111   SEND Entry
```

```
HASID....0026     PASID....0026     SASID..0026     JOBNAME..USER2
TCB......006D6D50 TIE......0000BDF8 PLIST..0000BE0C DUCB.....00000009 KEY..8
ADSNAME..USER2    SUBTASK..EZASOKEC                 TOKEN....7F755798 09806FB0
LOCAL  PORT..0         IPADDR ..0.0.0.0
REMOTE PORT..11007     IPADDR ..9.37.65.134
 SOCKET...:  0                                                 Addr..0000BA50
 NBYTE....:  96                                                Addr..0000BA6C
 BUF......:  (** PARAMETER NOT ADDRESSABLE **)                 Addr..00015F38
 FLAGS....:  0 (NONE)                                          Addr..0000BC04
```

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*.

```
===================================================================000085C1
 MVS026    SOCKAPI   60050004  19:36:01.934828   ACCEPT Exit
HASID....01F6     PASID....01F6     SASID..01F6     JOBNAME..EZASOKUE
TCB......006E6A68 TIE......00006DF0 PLIST..00006E04 DUCB.....0000000D KEY..8
ADSNAME..EZASOKUE SUBTASK..EZASOKUE                 TOKEN....7F6F3798 09902FB0
LOCAL  PORT..11007     IPADDR ..9.67.113.58
REMOTE PORT..0         IPADDR ..0.0.0.0
 REQAREA..:  00000000x                                         Addr..00006D80
 SOCKET...:  0                                                 Addr..00006BA8
 NAME.....:  (NO DATA)                                         Addr..00006BAC
 DIAG. RSN:  76620291x
 ERRNO....:  5 (EIO)                                           Addr..00006EA8
 RETCODE..:  -1                                                Addr..00006EAC
```

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.

```
===================================================================000051CB
 MVS026    SOCKAPI   60050066  19:02:01.426345   GETHOSTBYADDR Exit

HASID....0027     PASID....0027     SASID..0027     JOBNAME..EZASOKGH
TCB......006E6A68 TIE......00007DF8 PLIST..00007E0C DUCB.....0000000A KEY..0
ADSNAME..EZASOKGH SUBTASK..00000000 00000000        TOKEN....00000000 09902FB0
 HOSTENT..:                                                    Addr..00005F08
 HOSTNAME.:                                                    Addr..00005F30
 Loopback
 FAMILY...:  2                                                 Addr..00005F10
 ADDR LEN.:  4                                                 Addr..00005F14
 HOSTADDR.:  127.0.0.1                                         Addr..00005F54
 ALIAS....:  LOOPBACK                                          Addr..00005F3C
 ALIAS....:  LOCALHOST                                         Addr..00005F48
 RETCODE..:  0                                                 Addr..00007EB4
```

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*.

```
===================================================================0000134C
  MVS150    SOCKAPI   6005006D  15:06:07.294268   GETADDRINFO Entry
 HASID....002D     PASID....002D     SASID..002D     JOBNAME..USER1X
 TCB......007F63B0 TIE......0A90AAD8 PLIST..0A90AAEC DUCB.....00000009 KEY..8
 ADSNAME.......... SUBTASK..EZASO6CS                 TOKEN....7F694220 0A97EFB0
  NODELEN..:  6                                                Addr..0A973490
  NODE.....:                                                   Addr..0A973390
      MVS150
  SERVLEN..:  0                                                Addr..0A9734B8
  SERVICE..:  (NO DATA)                                        Addr..0A973498
  HINTS....:  0A913F70x (ADDRINFO Address)                     Addr..0A913F90
  ADDRINFO Structure..:
   AF.......  0 (AF_UNSPEC)     FLAGS..... 00000002x
   SOCTYPE..  0 (UNKNOWN)       PROTO..... 0 (IPPROTO_IP)
   NAME.....  00000000x         NAMELEN... 0
   CANONNAME  00000000x         NEXT...... 00000000x
  CANNLEN..:  (NO DATA)                                        Addr..0A9734C0
  RES......:  (NO DATA)                                        Addr..0A913F94
 ===================================================================0000134D
  MVS150    SOCKAPI   6005006E  15:06:09.997756   GETADDRINFO Exit
 HASID....002D     PASID....002D     SASID..002D     JOBNAME..USER1X
 TCB......007F63B0 TIE......0A90AAD8 PLIST..0A90AAEC DUCB.....00000009 KEY..8
 ADSNAME.......... SUBTASK..EZASO6CS                 TOKEN....7F694220 0A97EFB0
  HINTS....:  0A913F70x (ADDRINFO Address)                     Addr..0A913F90
  ADDRINFO Structure..:
   AF.......  0 (AF_UNSPEC)     FLAGS..... 00000002x
   SOCTYPE..  0 (UNKNOWN)       PROTO..... 0 (IPPROTO_IP)
   NAME.....  0002111Cx         NAMELEN... 0
    PORT....  0                 IPADDR.... 0.0.0.0
    FAMILY..  0 (UNKNOWN)       RESERVED.. 0000000000000000x
   CANONNAME  00000000x         NEXT...... 00000000x
  CANNLEN..:  22                                               Addr..0A9734C0
  RES......:  0002111Cx (ADDRINFO Address)                     Addr..0A913F94
  ADDRINFO Structure..:
   AF.......  2 (AF_INET)       FLAGS..... 00000000x
   SOCTYPE..  1 (STREAM)        PROTO..... 0 (IPPROTO_IP)
   NAME.....  0002114Cx         NAMELEN... 16
    PORT....  0                 IPADDR.... 9.67.113.117
    FAMILY..  2 (AF_INET)       RESERVED.. 0000000000000000x
   CANONNAME  0002101Cx         NEXT...... 00000000x
   MVS150.raleigh.ibm.com
```

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.*

```
===================================================================0000134E
  MVS150    SOCKAPI   6005006F  15:06:09.998002   FREEADDRINFO Entry
 HASID....002D     PASID....002D     SASID..002D     JOBNAME..USER1X
 TCB......007F63B0 TIE......0A90AAD8 PLIST..0A90AAEC DUCB.....00000009 KEY..8
 ADSNAME.......... SUBTASK..EZASO6CS                 TOKEN....7F694220 0A97EFB0
  ADDRINFO.:  0002111Cx (ADDRINFO Address)                     Addr..0A913F94
  ADDRINFO Structure..:
   AF.......  2 (AF_INET)       FLAGS..... 00000000x
   SOCTYPE..  1 (STREAM)        PROTO..... 0 (IPPROTO_IP)
   NAME.....  0002114Cx         NAMELEN... 16
    PORT....  0                 IPADDR.... 9.67.113.117
    FAMILY..  2 (AF_INET)       RESERVED.. 0000000000000000x
   CANONNAME  0002101Cx         NEXT...... 00000000x
   MVS150.raleigh.ibm.com
 ===================================================================0000134F
  MVS150    SOCKAPI   60050070  15:06:09.999021   FREEADDRINFO Exit
 HASID....002D     PASID....002D     SASID..002D     JOBNAME..USER1X
 TCB......007F63B0 TIE......0A90AAD8 PLIST..0A90AAEC DUCB.....00000009 KEY..8
 ADSNAME.......... SUBTASK..EZASO6CS                 TOKEN....7F694220 0A97EFB0
```

The GETNAMEINFO for IPv4 or IPv6 Resolver API shows the call is requesting the name of the IPv6 address ::1 and the service name for port 1031.

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 *z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference*.

```
=====================================================================0000135F
  MVS150    SOCKAPI   6005006B  15:06:45.481639   GETNAMEINFO Entry
 HASID....0025     PASID....0025     SASID..0025     JOBNAME..USER1Y
 TCB......007F62F8 TIE......0A903AD8 PLIST..0A903AEC DUCB.....00000008 KEY..8
 ADSNAME.......... SUBTASK..EZSO6CC                  TOKEN....7F6E2220 0A977FB0
  NAMELEN..:  28                                                Addr..0A96C348
  NAME.....:                                                    Addr..0A96C500
    PORT... 1031          IPADDR.. ::1
    FAMILY. 19 (AF_INET6) LENGTH.. 0
    FLOWINFO. 00000000x   SCOPID.. 00000000x
  HOSTLEN..:  255                                               Addr..0A96C450
  HOST.....:  (NO DATA)                                         Addr..0A96C350
  SERVLEN..:  32                                                Addr..0A96C478
  SERVICE..:  (NO DATA)                                         Addr..0A96C458
  FLAGS....:  00000004x                                         Addr..0A96C480
 =====================================================================00001360
  MVS150    SOCKAPI   6005006C  15:06:46.707053   GETNAMEINFO Exit
 HASID....0025     PASID....0025     SASID..0025     JOBNAME..USER1Y
 TCB......007F62F8 TIE......0A903AD8 PLIST..0A903AEC DUCB.....00000008 KEY..8
 ADSNAME.......... SUBTASK..EZSO6CC                  TOKEN....7F6E2220 0A977FB0
  HOSTLEN..:  23                                                Addr..0A96C450
  HOST.....:                                                    Addr..0A96C350
       loop6int.resdns.ibm.com
  SERVLEN..:  4                                                 Addr..0A96C478
  SERVICE..:  1031                                              Addr..0A96C458
  FLAGS....:  00000004x                                         Addr..0A96C480
```

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.

```
=====================================================================00001734
  MVS026    SOCKAPI   6005001F  20:42:44.805938   IOCTL Entry

 HASID....19       PASID....19       SASID..19       JOBNAME..USER1
 TCB......006AFD40 TIE......68DF8    PLIST..00068E0C DUCB.....00000008 KEY..8
 ADSNAME..USER1    SUBTASK..00000000 00000000        TOKEN....7F67F798 0A2B4FB0
 LOCAL  PORT..11007     IPADDR ..9.67.113.58
 REMOTE PORT..0         IPADDR ..0.0.0.0
  SOCKET...:  0                                                 Addr..000685A0
  COMMAND..:  SIOCGIFCONF                                       Addr..0006782C
  REQARG...:                                                    Addr..00068928
    BUFFER LENGTH.. 99
 =====================================================================00000323
  MVS026    SOCKAPI   60050020  20:42:44.806101   IOCTL Exit

 HASID....19       PASID....19       SASID..19       JOBNAME..USER1
 TCB......006AFD40 TIE......68DF8    PLIST..00068E0C DUCB.....00000008 KEY..8
 ADSNAME..USER1    SUBTASK..00000000 00000000        TOKEN....7F67F798 0A2B4FB0
 LOCAL  PORT..11007     IPADDR ...9.67.113.58
 REMOTE PORT..0         IPADDR ...0.0.0.0
  SOCKET...:  0                                                 Addr..000685A0
  COMMAND..:  SIOCGIFCONF                                       Addr..0006782C
  RETARG...:                                                    Addr..000685C4
  Socket Name.. TR1
```

```
    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.

```
=====================================================================00001773
 MVS026    SOCKAPI   60050045  19:19:20.954789   READV Exit

HASID....0024     PASID....0024     SASID..0024     JOBNAME..EZASOKKS
TCB......006E6A68 TIE......00007DF8 PLIST..00007E0C DUCB.....0000000B KEY..8
ADSNAME..EZASOKKS SUBTASK..EZASOKKS                 TOKEN....7F6F3798 09902FB0
LOCAL  PORT..11007     IPADDR ..9.67.113.58
REMOTE PORT..1032      IPADDR ..9.67.113.58
 REQAREA..:  00007D90x                                          Addr..00007D90
 SOCKET...:  1                                                  Addr..0000776C
 IOVCNT...:  3                                                  Addr..000077B4
 IOVENTRY.:  LENGTH..10     ALET..0x                            Addr..00007890
     +0000  E38889A2  4089A240  8396                | This is co        |
 IOVENTRY.:  LENGTH..10     ALET..0x                            Addr..0000789A
     +0000  99998583  A34B                          | rrect.            |
 IOVENTRY.:  LENGTH..10     ALET..0x                            Addr..000078A4
 RETCODE..:  16 BYTES TRANSFERRED                               Addr..00007EB4
```

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).

```
=====================================================================000077EC
 MVS026    SOCKAPI   60050040  19:24:11.552924   Default INITAPI

HASID....0027     PASID....0027     SASID..0027     JOBNAME..EZASOKSX
TCB......006E6A68 TIE......00007DF0 PLIST..00007E04 DUCB.....0000000A KEY..8
ADSNAME..EZASOKSX SUBTASK..00000000 00000000        TOKEN....7F6F3798 09902FB0
 MAXSNO...:  49
 APITYPE..:  2
 RETCODE..:  0
```

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).

```
 =====================================================================00000168
  MVS026    SOCKAPI   60050069  22:46:48.185419   Default TERMAPI

 HASID....01F9     PASID....01F9     SASID..01F9     JOBNAME..EZASOKQS
 TCB......006E6A68 TIE......08920888 PLIST..00000000 DUCB.....00000008 KEY..6
 ADSNAME..EZASOKQS SUBTASK..EZASOKQS                 TOKEN....7F6F3798 00000000
```

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.

```
===================================================================00024EDF
  BOTSWANA  SOCKAPI   6005004C  20:51:35.477605   SELECT Entry

HASID....0078     PASID....0078     SASID..0078     JOBNAME..TN1
TCB......007F6988 TIE......1463227C PLIST..1477EF18 DUCB.....00000016 KEY..8
ADSNAME..         SUBTASK..14632138                 TOKEN....7F75FFC8 1468FA90
 REQAREA..:  1477EEF0x                                          Addr..1477EF98
 MAXSOC...:  100                                                Addr..14632258
 TIMEOUT..:  SECOND..0      MICRO SECOND..500000                Addr..1463226C
 RSNDMSK..:                                                     Addr..14632108
 SOCKET NO.    READ SOCKET MASK (INPUT)
 (Decimal)     (Binary)
 31  0  00000000 00000000 00000000 00001111
 63  32 00111011 11111111 11111111 11111101
 95  64 11111111 11111111 10111111 11111111
 127 96 00000000 00000000 00000000 11110111
   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.

```
===================================================================00024EDF
  BOTSWANA  SOCKAPI   6005004C  20:51:35.477605   SELECT Entry (read mask)

HASID....0078     PASID....0078     SASID..0078     JOBNAME..TN1
TCB......007F6988 TIE......1463227C PLIST..1477EF18 DUCB.....00000016 KEY..8
ADSNAME..         SUBTASK..14632138                 TOKEN....7F75FFC8 1468FA90
 REQAREA..:  1477EEF0x                                          Addr..1477EF98
 MAXSOC...:  65535                                              Addr..14632258
 TIMEOUT..:  SECOND..0      MICRO SECOND..500000                Addr..1463226C
 RSNDMSK..:                                                     Addr..14632108
 SOCKET NO.    READ SOCKET MASK (INPUT)
 (Decimal)     (Binary)
```

SELECTEX: The SELECTEX call can contain a list of ECBs. The high-order bit on the SELECB address indicates whether or not a list of ECBs was specified. Since the high-order bit is on in this example, there is a list of ECBs. The end of the list is indicated by the high-order bit in the ECB address. In this example, the time limit expired before any ECBs were posted. Since no selected sockets were ready, the read, write, and error masks indicate there is no data to report.

```
======================================================================000078FB
 MVS026    SOCKAPI   6005004F  19:25:48.610379   SELECTEX Exit

HASID....0027     PASID....0027     SASID..0027     JOBNAME..EZASOKX4
TCB......006E6A68 TIE......00007DF8 PLIST..00007E0C DUCB.....0000000C KEY..8
ADSNAME..EZASOKX4 SUBTASK..BARBARA                  TOKEN....7F6F3798 09902FB0
 MAXSOC...:  33                                                 Addr..00007AE8
 TIMEOUT..:  SECOND..0      MICRO SECOND..35                    Addr..00007AF4
 RRETMSK..:  (NO DATA)                                          Addr..00007B0C
 WRETMSK..:  (NO DATA)                                          Addr..00007B14
 ERETMSK..:  (NO DATA)                                          Addr..00007B1C
 SELECB...:                                                     Addr..80007B60
 ECB......:  00000000x                                          Addr..00007B70
 ECB......:  00000000x                                          Addr..00007B74
 ECB......:  00000000x                                          Addr..00007B78
 ECB......:  00000000x                                          Addr..80007B7C
 RETCODE..:  0 (TIME LIMIT EXPIRED)                             Addr..00007EB4
```

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.

```
======================================================================00000158
 MVS026    SOCKAPI   6005006A  22:46:48.173348   Token Error

HASID....01F9     PASID....01F9     SASID..01F9     JOBNAME..EZASOKQS
TCB......006E6A68 TIE......00006DF8 PLIST..00006E0C DUCB.....00000008 KEY..8
ADSNAME..         SUBTASK..                         TOKEN....7F6F3798 09902FB0
 CALL.....:  BIND
 TOKEN....:  7F6F3798 09902FB0 FFFF0000 00003FC5x
 ERRNO....:  1028 (EIBMINVTCPCONNECTION)
 RETCODE..:  -1
```

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.

```
======================================================================000086FC
 MVS026    SOCKAPI   60050041  19:36:04.965468   Unsolicited Event Exit Invoked

HASID....0024     PASID....0024     SASID..0024     JOBNAME..TCPIPROC
TCB......006E6A40 TIE......00006DF0 PLIST..00000000 DUCB.....00000000 KEY..0
ADSNAME..EZASOKUE SUBTASK..EZASOKUE                 TOKEN....7F6F3798 00000000
 UEEXIT...:  ADDRESS..00006B30  TOKEN..00006D80x   ASCB.....00F94C80x KEY..8
             REASON...1 (TCP/IP TERMINATION)
```

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 85 shows the data trace record corresponding to the READ Exit SOCKAPI trace entry in Figure 17 on page 86. 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.

```
 MVS026    DATA      00000003  20:08:09.181239  Data Trace

 JOBNAME =    EZASOKAS          FROM  FULL
  TOD CLOCK = XB395B2C2  40035C03
  PKT 2           LOST RECORDS = 0      HDR SEQUENCE NUM = 1
 SOURCE IP ADDR = 9.67.113.58      DEST IP ADDR = 9.67.113.58
 SOURCE PORT = 11007  DEST PORT = 1040   ASID = X0024  TCB = X006E6A68
  DATA LENGTH = X0078
0000 E38889A2 4089A240 8140A2A3 99899587 *This is a string|....@..@.@......*
0010 40A689A3 88408696 99A3A840 83888199 * with forty char|@....@.....@....*
0020 8183A385 99A24B40 E38889A2 4089A240 *acters. This is |......K@....@..@*
0030 8140A2A3 99899587 40A689A3 88408696 *a string with fo|.@......@....@..*
0040 99A3A840 83888199 8183A385 99A24B40 *rty characters. |...@..........K@*
0050 E38889A2 4089A240 8140A2A3 99899587 *This is a string|....@..@.@......*
0060 40A689A3 88408696 99A3A840 83888199 * with forty char|@....@.....@....*
0070 8183A385 99A24B40                   *acters.         |......K@        *
```

Figure 16. Data trace record.

```
=====================================================================00002403
 MVS026    SOCKAPI   60050043  20:08:09.181354   READ Exit

HASID....0024     PASID....0024     SASID..0024     JOBNAME..EZASOKAS
TCB......006E6A68 TIE......00006DF8 PLIST..00006E0C DUCB.....00000009 KEY..8
ADSNAME..EZASOKAS SUBTASK..EZASOKAS                 TOKEN....7F6F3798 09902FB0
LOCAL  PORT..11007     IPADDR ..9.67.113.58
REMOTE PORT..1040      IPADDR ..9.67.113.58
REQAREA..:  00006D90x                                          Addr..00006D90
SOCKET...:  1                                                  Addr..00006B94
NBYTE....:  120                                                Addr..00006B90
BUF......:                                                     Addr..00006B96
    +0000  E38889A2  4089A240  8140A2A3  99899587  | This is a string |
    +0010  40A689A3  88408696  99A3A840  83888199  |  with forty char |
    +0020  8183A385  99A24B40  E38889A2  4089A240  | acters. This is  |
    +0030  8140A2A3  99899587  40A689A3  88408696  | a string with fo |
    +0040  99A3A840  83888199  8183A385  99A24B40  | rty characters.  |
    +0050  E38889A2  4089A240  8140A2A3  99899587  | This is a string |
RETCODE..:  120 BYTES TRANSFERRED                              Addr..00006EB4
=====================================================================00002407
```

Figure 17. SOCKAPI trace record.

The packet trace, on the other hand, does not correlate well with the SOCKAPI trace option. The packet trace records data being sent or received between the TCP/IP stack and the network. The packet trace data has headers and the data can be segmented or packed.

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 87 illustrates the overall control and data flow in the IP packet tracing facility.

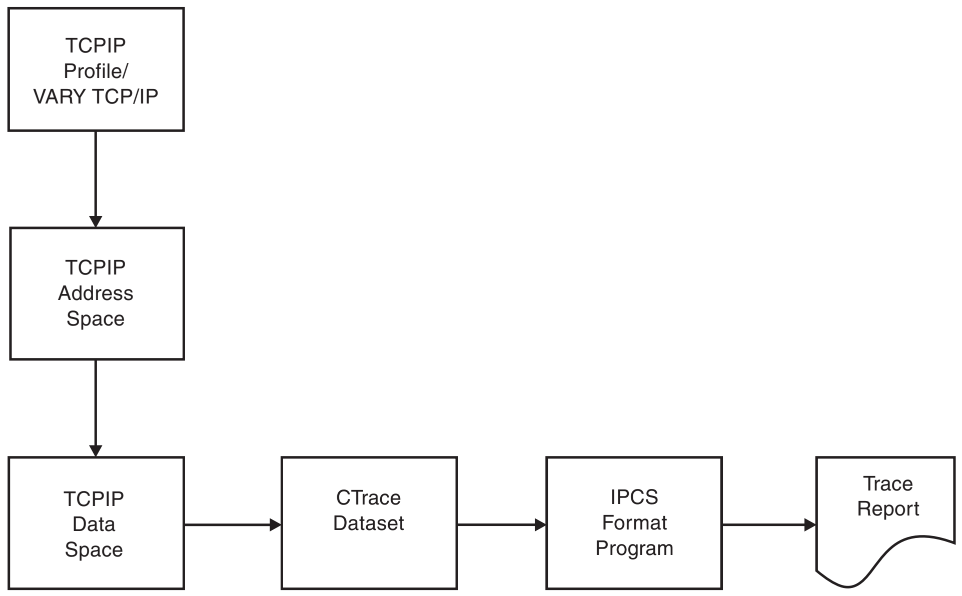


Figure 18. Control and data flow in the IP packet tracing facility

Supported devices

IP packet tracing is supported for all network interfaces supported by TCP/IP (including loopback). However, packets sent and received locally on IP addresses in the PROFILE.TCPIP HOME list are not traced.

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*.

Starting packet trace

To start packet trace, use the following command:

```
V TCPIP,tcpprocname,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 onetstat CS V1R7       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
  SrcPort:  *                DestPort: *
 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 onetstat:

```
NETSTAT -p TCPCS -d
MVS TCP/IP onetstat CS V1R7       TCPIP Name: TCPCS           18:17:48
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
    SrcPort:  *                DestPort: *
    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*.

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 48. 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.

EXCEPTION
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=SYSTCPDA). See "Syntax" on page 49 for the complete syntax of CTRACE.

OPTIONS syntax

OPTIONS component

```
►►──OPTIONS──((──┤ Data Selection ├──┤ Report Generation ├──))────────────────►◄
```

Data Selection:

```
├──┤ Device Type ├──┤ IP Identifier ├──┤ IP Address ├──┤ Name ├──────────────►

►──┤ Port Number ├──┤ Protocol ├──┤ Record Number ├──┤ Record Type ├──────────┤
```

Device Type:

```
                  ┌──────────┐
                  ▼          │
├──DEVTYPE──(────devtype────┴──)──────────────────────────────────────────────┤
```

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─────────────────────────►

        ┌────────────────────────────────┐                          ┌──────────────────────┐
        ▼                                │                          ▼                      │
►──QOS────(──quality_of_service──)──────┴──SITELOCAL──TRAFFICCLASS────(──traffic_class──)──┴──┤
```

IP Address:

```
├──┬─ipv4_address─┬──────────────────┬─┬──┬────┬──────────────────────────┤
   │              ├─/mmm.mmm.mmm.mmm─┤ │  └─-p─┘
   │              └─/nn──────────────┘ │
   ├─ip6_address─┬──────┬──────────────┤
   │             └─/nnn─┘              │
   ├─┬─A─┬─────────────────────────────┤
   │ ├─B─┤                             │
   │ ├─C─┤                             │
   │ ├─D─┤                             │
   │ ├─E─┤                             │
   │ ├─H─┤                             │
   │ ├─L─┤                             │
   │ ├─M─┤                             │
   │ ├─O─┤                             │
   │ └─*─┘                             │
   └─X'hhhhhhhh' ──────────────────────┘
```

Name:

```
                        ┌───────────────────────┐
                        ▼                       │
├──┬─┬─INTERFACE─┬───(─interface_name─)─┬─┬─────────────────────────────┤
   │ └─LINKNAME──┘                          │
   │                  ┌──────────────┐      │
   │                  ▼              │      │
   └─┬─JOBLIST──(─jobname─)─┬────────────┘
     └─JOBNAME──────────────────┘
```

Port Number:

```
├──BOOTP──┬───────────────────────────────┬──┬─DNS────┬──┬─────────────────────┬──►
          │     ┌─67 68─────────────┐      │  └─DOMAIN─┘  │    ┌─53──────────┐  │
          └─(───┼─port_number───────┼──)───┘              └─(──┴─port_number─┴─)┘
                └─port_number───────┘

►─EE──┬──────────────────────────┬──FINGER──┬───────────────────────────┬──►
      │   ┌─12000────────┐       │          │    ┌─79───────────┐       │
      └─(─┴─port_number──┴──)────┘          └─(──┴─port_number──┴──)────┘

►─FTP──┬────────────────────────────────────────────────────┬──►
       │   ┌─20───────────┐   ┌─21───────────┐               │
       └─(─┴─port_number──┴───┴─port_number──┴──)────────────┘

►─GOPHER──┬─────────────────────────────┬──►
          │    ┌─70───────────┐         │
          └─(──┴─port_number──┴──)──────┘
```

```
►──┬─HTTP─┬──┬─────────────────────────┬─IKE──┬──────────────────────────┬──────────►
   └─WWW──┘  │    ┌─80──────────┐      │      │    ┌─500─────────┐       │
             └─(──┴─port_number─┴──)───┘      └─(──┴─port_number─┴──)────┘


                                    ┌──────────────┐
                                    ▼              │
►─NTP──┬─────────────────────────┬─PORT──(───port_number──┴─)────────────────────►
       │    ┌─123─────────┐      │
       └─(──┴─port_number─┴──)───┘

►──┬─RIP────┬──┬─────────────────────────┬─RIPNG──┬─────────────────────────┬─────►
   └─ROUTER─┘  │    ┌─520─────────┐      │        │    ┌─521─────────┐      │
               └─(──┴─port_number─┴──)───┘        └─(──┴─port_number─┴──)───┘

►─RPC──┬─────────────────────────┬─SASP──┬──────────────────────────┬─────────────►
       │    ┌─111─────────┐      │       │    ┌─3860────────┐       │
       └─(──┴─port_number─┴──)───┘       └─(──┴─port_number─┴──)────┘

►─SMTP──┬─────────────────────────┬───────────────────────────────────────────────►
        │    ┌─25──────────┐      │
        └─(──┴─port_number─┴──)───┘

►─SNMP──┬──────────────────────────────────────────────────┬──────────────────────►
        │    ┌─161─────────┐   ┌─162─────────┐             │
        └─(──┴─port_number─┴───┴─port_number─┴──)──────────┘

►─TELNET──┬─────────────────────────────────────────────────────────────────┬─────►
          │    ┌─23──────────┐   ┌─80───────────┐                           │
          └─(──┴─port_number─┴───┴─screen_width─┴──┬───────────┬──)─────────┘
                                                   ├─SUMMARY───┤
                                                   └─DETAIL────┘

►─TFTP──┬─────────────────────────┬─TIME──┬─────────────────────────┬─IKE──────────┤
        │    ┌─69──────────┐      │       │    ┌─37──────────┐      │
        └─(──┴─port_number─┴──)───┘       └─(──┴─port_number─┴──)───┘
```

Protocol:

```
                                              ┌─────────────────┐
                                              ▼                 │
├──GRE──ICMP──┬─ICMP6──┬─IGMP──OSPFI──PROTOCOL(──protocol_number─┴─)──TCP──UDP──AH──ESP──────┤
              └─ICMPV6─┘
```

Record Number:

```
                  ┌─999999999─┐                                  ┌─────────────────┐
                  │           │                                  ▼                 │
├──LIMIT──(───────┴───────────┴─record count──)──RECORDS──(───record number───┴─)────┤
```

Record Type:

```
              ┌──────────────┐                    ┌─0─────────────────┐
├──CID──(──────▼─cid-number───┴──)──DATASIZE──(───┤                   ├──────────────────────►
                                                  └─(──data size──)───┘

►─DATTRACE──FLAGS──(──ABBREV──ACK──DATA──DF────────────────────────────────────────────────►

►─FIC──FIN──FULL──HOME──IN──IPO──IPV6EXT──IPV4──IPV6──IPTEXT───────────────────────────────►

►─LIC──MIC──OFFLOAD──OIC──OUT──PING──PSH──RSM──RST──SEG──SYN──TCPO──TOS────────────────────►

►─TUNNEL──URG──ZWIN──)───┬─PACKETTRACE─┬──X25──USEREXIT(exitname)──GMT──LOCAL──────────────►
                         └─PKT─────────┘

                                           ┌─10────┐   ┌─10─────┐
►─SESSION─┬─────────────────────┬─SPEED(───┴─local─┴─,─┴─remote─┴─)────────────────────────►
          │       ┌─DETAIL───┐  │
          └─(─────┼─SUMMARY──┼─)┘
                  ├─STATE────┤
                  └─PIPE─────┘

►─SNIFFER─┬─────────────────────────────────────┬──────────────────────────────────────────┤
          │      ┌─200───┐   ┌─ETHERNET──┐      │
          └─(────┴─nnnnn─┴───┼─TCPDUMP───┼──)───┘
                             └─TOKENRING─┘
```

Report Generation:

```
   ┌─BOTH───┐
├──┼─ASCII──┼──BASIC─┬──────────────────────────────┬──────────────────────────────────────►
   ├─EBCDIC─┤        │  ┌─BASIC (SUMMARY)────────┐  │
   └─HEX────┘        └──┴─(──BASIC (DETAIL)──)───┴──┘

►─CHECKSUM─┬─────────────────────────────────┬─NOCHECKSUM─────────────────────────────────►
           │  ┌─CHECKSUM (SUMMARY)────────┐  │
           └──┴─(──CHECKSUM (DETAIL)──)───┴──┘

►─CLEANUP─┬─────────────────────┬─DATASIZE─┬─────────────────────┬─DEBUG─┬──────────────┬──►
          │     ┌─500───┐       │          │     ┌─0─────┐       │       │    ┌────┐    │
          └─(───┴─nnnnn─┴──)────┘          └─(───┴─nnnnn─┴──)────┘       └─(──▼─nn─┴─)──┘

           ┌─200──┐                                                  
►─DELAYACK(─┴─nnnn─┴──)──DUMP─┬───────────────────────┬─EXPORT─┬────────────────────┬──────►
                              │      ┌─65535─┐        │        │    ┌─SUMMARY─┐     │
                              └─(────┴─nnnnn─┴──)─────┘        └─(──┴─DETAIL──┴──)──┘
```

```
                                                   ┌─125────┐   ┌─150─────┐
►──┬─FMT────┬──┬────────────────────────┬──FULL──GAIN(──┴─rtgain─┴──,──┴─vargain─┴──)──NOT──OPTION──────►
   └─FORMAT─┘  │       ┌─DETAIL──┐      │
               └─(─────┴─SUMMARY─┴──)───┘

   ┌─┤ REASSEMBLY ├─┐   ┌─SEGMENT───┐
►──┴─NOREASSEMBLY───┴───┴─NOSEGMENT─┴──SESSION──────────────────────────────────────────────────────►
                                                └─(──┬─DETAIL───┬──)─┘
                                                     ├─SUMMARY──┤
                                                     ├─STATE────┤
                                                     └─PIPE─────┘

          ┌─10────┐     ┌─10─────┐
►─SPEED(──┴─local─┴──,──┴─remote─┴──)──SNIFFER───────────────────────────────────────────────────────►
                                               │    ┌─200───┐   ┌─ETHERNET──┐      │
                                               └─(──┴─nnnnn─┴───┼─TCPDUMP───┼──)───┘
                                                                └─TOKENRING─┘

►──┬──┬─STATISTICS─┬──┬───────────────────────┬──┬──STREAMS─────────────────────────────────────────►
   │  └─STATS──────┘  │      ┌─SUMMARY─┐       │  │         │    ┌─128───┐   ┌─SUMMARY─┐      │
   │                  └─(────┴─DETAIL──┴──)───┘  │         └─(──┴─nnnnn─┴───┴─DETAIL──┴──)───┘
   └─TALLY───────────────────────────────────────┘

►─SUMMARY──TOD──────────────────────────────────────────────────────────────────────────────────────┤
```

REASSEMBLY:

```
├──REASSEMBLY──┬────────────────────────────────────────┬──────────────────────────────────────────┤
               │     ┌─32767─┐   ┌─SUMMARY─┐          │
               └─(───┴─nnnnn─┴───┴─DETAIL──┴────)─────┘
```

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(255.255.255.255/255.255.255.255).

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.

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

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).

| **ESP**
| Select packets with a protocol number of 50 and also NAT packets on UDP
| port 4500. Equivalent to PROTOCOL(50).

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).

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.
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.
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.
OFFLOAD	Select outbound packets for which segmentation has been offloaded.
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 NOREASSEMBLY 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.

SUMMARY
Format the IP header and protocol header. DETAIL is the default.

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 107.

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'[]-nnnnn[])
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

     `X'7f000001'`
   - 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(ipid_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 4500). 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().

MULTICAST
Select packets with either an IPv4 or IPv6 multicast address. Equivalent to CLASSD IPADDR(FF00::/8).

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
- | 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
- ESP
- GRE
- ICMP
- ICMP6,
- ICMPV6
- IGMP
- OSPFI
- TCP
- UDP

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|TOKENRING)]
Writes the trace records in a format acceptable for downloading to other trace analysis programs, such as Network Associates' Sniffer Network Analyzer or programs from http://www.tcpdump.

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 http://www.tcpdump files with an Ethernet header.

TOKENRING
If this keyword is specified, the output is formatted for the token-ring analysis application of the analyzer. This keyword specifies the file format only and does not imply that only packets traced on a token ring are collected. Packets from all devices can be collected using this option.

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 7954 (8000 - 46) for SNIFFER(TOKENRING) and the maximum 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.

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.

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).

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 (X'hh').

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.

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))
|  z/OS TCP/IP Packet Trace Formatter, (C) IBM 2000-2005, 2004.365
|  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(70) 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.

```
**** 2004/01/26
I - Inbound packet
O - Outbound packet

DP   Nr hh:mm:ss.mmmmmm IpId    Seq_num    Ack_num  Wndw Flags            DatLn Source/Destination
OT    1 14:18:00.447462 19E6 1452693653          0 32768 SYN                  0 10.7.1.61-3470
                                                                                192.168.248.44-5000
IT    2 14:18:00.601784 4E3C 3454024895 1452693654 32768 ACK SYN              0 192.168.248.44-5000
                                                                                10.7.1.61-3470
OT    3 14:18:00.601917 1A00 1452693654 3454024896 32768 ACK                  0 10.7.1.61-3470
                                                                                192.168.248.44-5000
IT    4 14:18:01.111074 4E3D 3454024896 1452693654 32768 ACK PSH             47 192.168.248.44-5000
                        3232302D 2057656C *.......% 220- Wel*                   10.7.1.61-3470
IT    5 14:18:01.126148 4E3E 3454024943 1452693654 32768 ACK PSH             65 192.168.248.44-5000
                        32323020 52582E69 *........ 220 RX.i*                   10.7.1.61-3470
OT    6 14:18:01.126248 1A46 1452693654 3454025008 32703 ACK                  0 10.7.1.61-3470
                                                                                192.168.248.44-5000
OT    7 14:18:03.290611 1B7F 1452693654 3454025008 32703 ACK PSH             10 10.7.1.61-3470
                        55534552 20637673 *........ USER cvs*                   192.168.248.44-5000
IT    8 14:18:03.373175 4E3F 3454025008 1452693664 32768 ACK PSH             32 192.168.248.44-5000
                        33333120 50617373 *....&/.. 331 Pass*                   10.7.1.61-3470
```

Figure 19. Example of a SUMMARY report

D Direction of the packet:

I	Inbound
O	Outbound

P The packet protocol

T	TCP
U	UDP
I	ICMP
G	IGMP
D	Data Trace
P	Neither TCP, UDP, ICMP, nor IGMP

Nr The Ctrace record number

hh:mm:ss.mmmmmmm
The time stamp of the record

Source
The source IP address and port number

Destination
The destination IP address and port number

IpId
The packet ID number in hexadecimal

- For TCP

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

cccccccccc
: The type of ICMP message

xxxxxxxxxx
: The first eight bytes of the user data in hexadecimal

- For IGMP

nnnnn
: The maximum response time

cccccccccc
: The type of IGMP message

nnn.nnn.nnn.nnn
: The IGMP group address

- Other protocols

cccccccccc
: 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((TCPCS)) 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,
  "OSAQDIOLINK     ","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,
  "OSAQDIOLINK     ","9.67.115.69     ","9.67.115.5      ",
    5971,     56,"ICMP","? LEN=28"
  "I O   ",      37,"19:52:27.783944",164.995736,    0.000134,
  "LOOPBACK        ","9.67.115.5      ","9.67.115.5      ",
     129,     56,"ICMP","? LEN=28"
  "O O   ",      40,"19:52:39.284802",176.496595,    5.500260,
  "OSAQDIO46       ","FEC9:C2D4::6:2900:EDC:217C","FEC9:C2D4::9:67:115:17",
      20,     60,"UDP","S=32810 D=33435 LEN=20"
  "O O   ",      41,"19:52:39.284870",176.496662,    0.000067,
  "OSAQDIO46       ","FEC9:C2D4::6:2900:EDC:217C","FF02::1:FF15:17",
      32,     72,"ICMPv6","ICMPv6"
  "I O   ",      42,"19:52:39.285955",176.497748,    0.001085,
  "OSAQDIO46       ","FEC9:C2D4::9:67:115:17","FEC9:C2D4::6:2900:EDC:217C",
      32,     72,"ICMPv6","ICMPv6"
  "O O   ",      49,"19:52:58.286347",195.498140,   13.972912,
  "LOOPBACK6       ","FEC9:C2D4::9:67:115:5","FEC9:C2D4::9:67:115:5",
      20,     60,"UDP","S=32810 D=33435 LEN=20"
  "I O   ",      50,"19:52:58.286530",195.498323,    0.000182,
  "LOOPBACK6       ","FEC9:C2D4::9:67:115:5","FEC9:C2D4::9:67:115:5",
      68,    108,"ICMPv6","ICMPv6"
  ```

  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)

```
"Flags ","Packet","Delta Time","Source         ",
"Destination     ","Protocol  ","Summary"
"I O   ",      10," 69.006010","9.67.115.5     ",
"9.67.115.5      ","IP"," S=9.67.115.5 D=9.67.115.5 LEN=71 ID=110"
"I O   ",      10," 69.006010","9.67.115.5     ",
"9.67.115.5      ","UDP"," S=1036 D=161 LEN=51"
"I O   ",      10," 69.006010","9.67.115.5     ",
"9.67.115.5      ","SNMP","GetRequest dpiPathNameForUnixStream.0"
"O O   ",      24,"  0.002956","9.67.115.5     ",
"9.67.115.69     ","IP"," S=9.67.115.5 D=9.67.115.69 LEN=40 ID=121"
"O O   ",      24,"  0.002956","9.67.115.5     ",
"9.67.115.69     ","UDP"," S=32810 D=33436 LEN=20"
"O O   ",      51,"  0.002695","FEC9:C2D4::9:67:115:5",
"FEC9:C2D4::9:67:115:5","IP"," S=FEC9:C2D4::9:67:115:5 D=FEC9:C2D4::9:67:115:5 LEN=60 ID=20"
"O O   ",      51,"  0.002695","FEC9:C2D4::9:67:115:5",
"FEC9:C2D4::9:67:115:5","UDP"," S=32810 D=33436 LEN=20"
"I O   ",      52,"  0.000244","FEC9:C2D4::9:67:115:5",
"FEC9:C2D4::9:67:115:5","IP"," S=FEC9:C2D4::9:67:115:5 D=FEC9:C2D4::9:67:115:5 LEN=108 ID=68"
"I O   ",      52,"  0.000244","FEC9:C2D4::9:67:115:5",
"FEC9:C2D4::9:67:115:5","ICMPv6"
```

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(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((FORMAT))
```

Examples:

```
1  3 MVSJ     PACKET   00000001 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.

Port	Keyword
67, 68	BOOTP
67, 68	DHCP
53	Domain
520	RIP
520	Router
161,162	SNMP
23	TELNET
69	TFTP

```
1   17 MVSN     PACKET   00000004 19:43:02.541728 Packet Trace
2 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: 120B 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

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 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. 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).

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 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(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((DUMP))
```

Examples:

```
1 MVS073   PACKET   00000001 19:49:42.788207 Packet Trace
 From Interface    : OSAQDIOLINK      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

 1  IP Header          : 20
000000 4500004E 43060000 8011FEC2 09437311  0943733F

 2  Protocol Header    : 8
000000 00890089 003AD7D7

 3  Data               : 50     Data Length: 50
000000 AD3D0110 00010000 00000000 20464845 |................ .=.......... FHE|
000010 50464345 4C454846 43455046 46464143 |&...<.....&..... PFCELEHFCEPFFFAC|
000020 41434143 41434143 41434142 4C000020 |............<... 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(`*`nnnnn`*`)`, where *nnnnn* is the maximum size of a reassembled packet.

Format:

```
CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((REASSEMBLY(DETAIL) STAT))
```

Examples:

```
1  Reassembly of: 9.67.113.65-0 9.27.11.173-0 Id: 0043 status: +Fic +Lic
2   Rcd Nr Time            Delta           Offset Length   Next   Gap  Data Flags
    1638 15:28:49.975479 00:00:00.000000      0   3976   3976     0  3976 Fic
    1639 15:28:49.975501 00:00:00.000022   3976   3976   7952     0  3976 Mic
    1640 15:28:49.975524 00:00:00.000044   7952   3976  11928     0  3976 Mic
    1641 15:28:49.975545 00:00:00.000066  11928   3976  15904     0  3976 Mic
    1642 15:28:49.975567 00:00:00.000088  15904   3976  19880     0  3976 Mic
    1643 15:28:49.975594 00:00:00.000115  19880   3976  23856     0  3976 Mic
    1644 15:28:49.975620 00:00:00.000141  23856   3976  27832     0  3976 Mic
    1645 15:28:49.975689 00:00:00.000210  27832   3976  31808     0  3976 Mic
    1646 15:28:49.975737 00:00:00.000258  31808   3976  35784     0  3976 Mic
    1647 15:28:49.975771 00:00:00.000292  35784   3976  39760     0  3976 Mic
    1648 15:28:49.975804 00:00:00.000325  39760   3976  43736     0  3976 Mic
    1649 15:28:49.975835 00:00:00.000356  43736   3976  47712     0  3976 Mic
    1650 15:28:49.975865 00:00:00.000386  47712   3976  51688     0  3976 Mic
    1651 15:28:49.975898 00:00:00.000419  51688   3976  55664     0  3976 Mic
    1652 15:28:49.975926 00:00:00.000447  55664   3976  59640     0  3976 Mic
    1653 15:28:49.975962 00:00:00.000483  59640   3976  63616     0  3976 Mic
    1654 15:28:49.975986 00:00:00.000507  63616    392  64008     0   392 Lic
  64,008 bytes is the final length of the IP packet
      17 packets were used for reassembly
  64,008 bytes were accumulated for reassembly
================================================================================
3  Packet Reassembly Report
Maximum reassembly buffer size is 65535
4  Reassembly of: 9.27.11.173-0 9.67.113.65-0 Id: 3694 status: +Fic +Lic
  Rcd Nr Time            Delta           Offset Length   Next   Gap  Data Flags
    1655 15:28:50.024685 00:00:00.000000      0   1480   1480     0  1480 Fic
    1656 15:28:50.024705 00:00:00.000019   1480   1480   2960     0  1480 Mic
    1657 15:28:50.024739 00:00:00.000053   2960   1480   4440     0  1480 Mic
    1658 15:28:50.024772 00:00:00.000086   4440   1480   5920     0  1480 Mic
    1659 15:28:50.025506 00:00:00.000820   5920   1480   7400     0  1480 Mic
    1660 15:28:50.030534 00:00:00.005848   7400   1480   8880     0  1480 Mic
    1661 15:28:50.030592 00:00:00.005906   8880   1480  10360     0  1480 Mic
    1662 15:28:50.030607 00:00:00.005921  10360   1480  11840     0  1480 Mic
    1663 15:28:50.030650 00:00:00.005964  11840   1480  13320     0  1480 Mic
    1664 15:28:50.030683 00:00:00.005997  13320   1480  14800     0  1480 Mic
    1665 15:28:50.030698 00:00:00.006012  14800   1480  16280     0  1480 Mic
    1666 15:28:50.042927 00:00:00.018241  16280   1480  17760     0  1480 Mic
    1667 15:28:50.042946 00:00:00.018261  17760   1480  19240     0  1480 Mic
    1668 15:28:50.043006 00:00:00.018320  19240   1480  20720     0  1480 Mic
    1669 15:28:50.043021 00:00:00.018335  20720   1480  22200     0  1480 Mic
    1670 15:28:50.043058 00:00:00.018372  22200   1480  23680     0  1480 Mic
    1671 15:28:50.043114 00:00:00.018428  23680   1480  25160     0  1480 Mic
    1672 15:28:50.043130 00:00:00.018444  25160   1480  26640     0  1480 Mic
    1673 15:28:50.043174 00:00:00.018488  26640   1480  28120     0  1480 Mic
    1674 15:28:50.043222 00:00:00.018536  28120   1480  29600     0  1480 Mic
    1675 15:28:50.043257 00:00:00.018571  29600   1480  31080     0  1480 Mic
    1676 15:28:50.043544 00:00:00.018858  31080   1480  32560     0  1480 Mic
    1677 15:28:50.043592 00:00:00.018906  32560   1480  34040     0  1480 Mic
    1678 15:28:50.043607 00:00:00.018921  34040   1480  35520     0  1480 Mic
    1679 15:28:50.044618 00:00:00.019932  35520   1480  37000     0  1480 Mic
    1680 15:28:50.044649 00:00:00.019963  37000   1480  38480     0  1480 Mic
    1681 15:28:50.044698 00:00:00.020012  38480   1480  39960     0  1480 Mic
    1682 15:28:50.044712 00:00:00.020026  39960   1480  41440     0  1480 Mic
    1683 15:28:50.044745 00:00:00.020059  41440   1480  42920     0  1480 Mic
    1684 15:28:50.044778 00:00:00.020092  42920   1480  44400     0  1480 Mic
    1685 15:28:50.050178 00:00:00.025492  44400   1480  45880     0  1480 Mic
    1686 15:28:50.050212 00:00:00.025527  45880   1480  47360     0  1480 Mic
    1687 15:28:50.050244 00:00:00.025558  48840   1480  50320 -1480  1480 Mic
    1688 15:28:50.050275 00:00:00.025589  50320   1480  51800     0  1480 Mic
    1689 15:28:50.050328 00:00:00.025642  51800   1480  53280     0  1480 Mic
    1690 15:28:50.050343 00:00:00.025657  53280   1480  54760     0  1480 Mic
    1691 15:28:50.054558 00:00:00.029872  54760   1480  56240     0  1480 Mic
    1692 15:28:50.054614 00:00:00.029928  57720   1480  59200 -1480  1480 Mic
    1693 15:28:50.054628 00:00:00.029942  59200   1480  60680     0  1480 Mic
    1694 15:28:50.054680 00:00:00.029994  62160   1480  63640 -1480  1480 Mic
    1695 15:28:50.054694 00:00:00.030008  63640    368  64008     0   368 Lic
  64,008 bytes is the final length of the IP packet
      41 packets were used for reassembly
  59,568 bytes were accumulated for reassembly
--------------------------------------------------------------------------------
5   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 type of packet:

Fic	First in chain. The offset was zero.
Mic	Middle in chain. The offset was nonzero and the more fragment flag was set.
Lic	Last in chain. 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.

5

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((TCPCS)) SHORT OPTIONS((SESSION TCP))
```

Examples:

```
-------------------------------------------------------------------------------
 1  2 packets summarized
 Local Ip Address:          FEC9:C2D4::6:2900:EDC:217C
 Remote Ip Address:         FEC9:C2D4::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:                                 1,               1
  (x) Untraced Packets:                   0,               0
  (.) In-order data:                      0,               0 ( 0.00%), ( 0.00%)
  (a) Acknowledgments:                    1,               0 (100.00%), ( 0.00%)
  (+) Data and ACK:                       0,               0 ( 0.00%), ( 0.00%)
  (u) Duplicate ACKs:                     0,               0 ( 0.00%), ( 0.00%)
  (w) Window size updates:                0,               0 ( 0.00%), ( 0.00%)
  (z) Zero window sizes:                  0,               0 ( 0.00%), ( 0.00%)
  (p) Window probes:                      0,               0 ( 0.00%), ( 0.00%)
  (k) Keepalive segments:                 0,               0 ( 0.00%), ( 0.00%)
  (r) Retransmissions:                    0,               0 ( 0.00%), ( 0.00%)
  (o) Out-of-order:                       0,               0 ( 0.00%), ( 0.00%)
  (d) Delayed ACKs:                       0,               0 ( 0.00%), ( 0.00%)
  (f) Fragments:                          0,               0 ( 0.00%), ( 0.00%)
 7  Time Spent on:                      Inbound,        Outbound
  (.) In-order data:        00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (a) Acknowledgments:      00:00:00.000957, 00:00:00.000000 (106.33%), ( 0.00%)
  (+) Data and ACK:        00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (u) Duplicate ACKs:      00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (w) Window size updates: 00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (z) Zero window sizes:   00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (p) Window probes:       00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (k) Keepalive segments:  00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (r) Retransmissions:     00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (o) Out-of-order:        00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (d) Delayed ACKs:        00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (f) Fragments:           00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
 8 Number of:                          Inbound,       Outbound
  (   S ) SYN:                          0,              1 ( 0.00%), (100.00%)
  ( A S ) ACK SYN:                      0,              0 ( 0.00%), ( 0.00%)
  (    F) FIN:                          0,              0 ( 0.00%), ( 0.00%)
  ( A  F) ACK FIN:                      0,              0 ( 0.00%), ( 0.00%)
  (  R  ) RST:                          1,              0 (100.00%), ( 0.00%)
  (U    ) URG:                          0,              0 ( 0.00%), ( 0.00%)
 9 Time Spent on:                      Inbound,       Outbound
  (    S ) SYN:            00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  ( A  S ) ACK SYN:        00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (     F) FIN:            00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  ( A   F) ACK FIN:        00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
  (   R  ) RST:            00:00:00.000957, 00:00:00.000000 (106.33%), ( 0.00%)
  (U     ) URG:            00:00:00.000000, 00:00:00.000000 ( 0.00%), ( 0.00%)
Messages:
2) The largest inbound window is less than twice the inbound MSS.
2) This may reduce inbound throughput for bulk data transfers.
2) It is usually desirable for the window size to be at twice the MSS.
3) The outbound side of the connection appears to be a bulk data transfer.
          I - Inbound packet
          O - Outbound packet
10 TcpHdr IO F       Seq        Ack RcvWnd  Data Delta Time      TimeStamp  RcdNr    State     Inf Ip_id   Rtt    TimeStmpV  Time
      S   O     29260429          0  32768     0   0.000000 19:55:46.934032    101    SYN_SENT   4  0028    0.00      0.000 316250
   A R    I a         0   29260430      0     0   0.000957 19:55:46.934989    102      CLOSED   4  0014    0.00      0.000 316250
```

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

These statistics are about the number of data packets that flow for carrying data. The percentages are based on the number of packets.

7

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.

8

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.

9

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.

10

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.

f The packet was a fragment of a complete IP packet.

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
1 Index Count Link            Address
    1     5 OSAQDIOLINK      9.67.115.63
    2    42 LOOPBACK         127.0.0.1
    3    18 OSAQDIOLINK      9.67.115.5
    4    31 OSAQDIO46        FEC9:C2D4::6:2900:EDC:217C
    5    21 OSAQDIO46        FE80::6:2900:EDC:217C
    6     6 LOOPBACK6        ::1
================================================================================

Sniffer Report
2     126 records written to USER2.SNIFFER.TRC
3  13,982 bytes written
4       0 packets were abbreviated
5       5 packets were truncated to 200 bytes
```

1
The list of device names found in the selected records. Each device is assigned an interface index.

2
This record count includes the two header records and one trailer record written to the SNIFFER data set.

3
The number of data bytes written to the SNIFFER data set. This is the amount of data to be downloaded.

4
The number of abbreviated records. This number is included in **5**.

5
The number of truncated records. Records were truncated because the size of the packet exceeded the logical record length of the SNIFFER file. You can avoid this 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 record type, device type, device name, job name, protocol, IP address, TCP port number, and UDP port number.

Format:

```
CTRACE COMP(SYSTCPDA) SUB((TCPCS)) SHORT OPTIONS((STATISTICS(DETAIL)))
```

Examples:

```
1    No packets required reassembly
================================================================================
SYSTCPDA Trace Statistics
   123 ctrace records processed
     0 segmented trace records read
     0 segmented trace records were lost
   123 trace records read
     0 records could not be validated
   123 records passed filtering
   123 packet trace records processed
     0 data trace records processed
================================================================================
2  Record Type Report
  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Record Type
     65     55       3814     10        644     1 2002/02/12 19:49:42   123 2002/02/12 19:57:45 1(Packet Trace)
     58     19       1828     39       2840    40 2002/02/12 19:52:39   117 2002/02/12 19:56:29 4(Packet Trace)
    123     74       5642     49       3484 Total
     2 Record Type(s) found
Ip Version Report
  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Ip Version
     65     55       3814     10        644     1 2002/02/12 19:49:42   123 2002/02/12 19:57:45 4
     58     19       1828     39       2840    40 2002/02/12 19:52:39   117 2002/02/12 19:56:29 6
    123     74       5642     49       3484 Total
     2 Ip Version(s) found
3  Device Type Report
  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Device Type
     42     42       2667      0          0     4 2002/02/12 19:49:48   123 2002/02/12 19:57:45 34(Loopback)
     23     13       1147     10        644     1 2002/02/12 19:49:42   103 2002/02/12 19:55:48 39(QDIO Ethernet)
      6      3        324      3        180    49 2002/02/12 19:52:58    54 2002/02/12 19:52:58 51(Loopback6)
     52     16       1504     36       2660    40 2002/02/12 19:52:39   117 2002/02/12 19:56:29 53(QDIO Ethernet6)
    123     74       5642     49       3484 Total
     4 Device Type(s) found
4  Interface Report
  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Interface
     42     42       2667      0          0     4 2002/02/12 19:49:48   123 2002/02/12 19:57:45 LOOPBACK
      6      3        324      3        180    49 2002/02/12 19:52:58    54 2002/02/12 19:52:58 LOOPBACK
     23     13       1147     10        644     1 2002/02/12 19:49:42   103 2002/02/12 19:55:48 OSAQDIOL
     52     16       1504     36       2660    40 2002/02/12 19:52:39   117 2002/02/12 19:56:29 OSAQDIO4
    123     74       5642     49       3484 Total
     4 Interface(s) found
Interface Address Report
  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Interface
      5      5        699      0          0     1 2002/02/12 19:49:42   103 2002/02/12 19:55:48 OSAQDIOLINK
                                                  Addr: 9.67.115.63
     42     42       2667      0          0     4 2002/02/12 19:49:48   123 2002/02/12 19:57:45 LOOPBACK
                                                  Addr: 9.67.115.5
     18      8        448     10        644    16 2002/02/12 19:51:17    33 2002/02/12 19:52:21 OSAQDIOLINK
                                                  Addr: 9.67.115.5
     31     14       1360     17       1340    40 2002/02/12 19:52:39   116 2002/02/12 19:56:29 OSAQDIO46
                                                  Addr: FEC9:C2D4::6:2900:EDC:217C
     21      2        144     19       1320    46 2002/02/12 19:52:44   117 2002/02/12 19:56:29 OSAQDIO46
                                                  Addr: FE80::6:2900:EDC:217C
      6      3        324      3        180    49 2002/02/12 19:52:58    54 2002/02/12 19:52:58 LOOPBACK6
                                                  Addr: FEC9:C2D4::9:67:115:5
    123     74       5642     49       3484 Total
     6 Interface Address(s) found
Asid Report
  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Asid
     34      0          0     34       2440    16 2002/02/12 19:51:17   111 2002/02/12 19:56:24 002A
     89     74       5642     15       1044     1 2002/02/12 19:49:42   123 2002/02/12 19:57:45 002B
    123     74       5642     49       3484 Total
     2 Asid(s) found
5  Protocol Report
  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Protocol
     30     29       1624      1        284     5 2002/02/12 19:49:48   123 2002/02/12 19:57:45 1(ICMP)
      4      2        120      2        160    99 2002/02/12 19:55:20   102 2002/02/12 19:55:46 6(TCP)
     53     26       2190     27       1440     1 2002/02/12 19:49:42   122 2002/02/12 19:57:45 17(UDP)
     36     17       1708     19       1600    41 2002/02/12 19:52:39   117 2002/02/12 19:56:29 58(ICMPv6)
    123     74       5642      49       3484 Total
     4 Protocol(s) found
 6  IP Address Report
  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss
      9      5        280       4        404    16 2002/02/12 19:51:17    27 2002/02/12 19:52:21
                                                  Addr: 9.67.115.1
     60     50       3115      10        644     4 2002/02/12 19:49:48   123 2002/02/12 19:57:45
                                                  Addr: 9.67.115.5
      5      5        699       0          0     1 2002/02/12 19:49:42   103 2002/02/12 19:55:48
                                                  Addr: 9.67.115.17
      5      5        699       0          0     1 2002/02/12 19:49:42   103 2002/02/12 19:55:48
                                                  Addr: 9.67.115.63
      9      3        168       6        240    22 2002/02/12 19:52:21    33 2002/02/12 19:52:21
                                                  Addr: 9.67.115.69
      4      0          0       4        240    55 2002/02/12 19:53:17    67 2002/02/12 19:53:32
                                                  Addr: FE80::6:2900:ADC:217C
     21      2        144      19       1320    46 2002/02/12 19:52:44   117 2002/02/12 19:56:29
                                                  Addr: FE80::6:2900:EDC:217C
      4      2        144       2        144    45 2002/02/12 19:52:44   105 2002/02/12 19:55:51
                                                  Addr: FE80::202:55FF:FE64:2DE7
      2      1         72       1         72   116 2002/02/12 19:56:29   117 2002/02/12 19:56:29
                                                  Addr: FE80::206:2AFF:FE66:C800
      6      3        216       3        216    76 2002/02/12 19:54:02    94 2002/02/12 19:55:09
                                                  Addr: FE80::206:2AFF:FE71:4400
     31     14       1360      17       1340    40 2002/02/12 19:52:39   116 2002/02/12 19:56:29
                                                  Addr: FEC9:C2D4::6:2900:EDC:217C
      6      3        324       3        180    49 2002/02/12 19:52:58    54 2002/02/12 19:52:58
                                                  Addr: FEC9:C2D4::9:67:115:5
      8      4        348       4        260    40 2002/02/12 19:52:39   102 2002/02/12 19:55:46
                                                  Addr: FEC9:C2D4::9:67:115:17
      3      2        376       1        304   110 2002/02/12 19:56:24   113 2002/02/12 19:56:24
                                                  Addr: FEC9:C2D4::206:2AFF:FE66:C800
      8      4        348       4        260    88 2002/02/12 19:55:04   100 2002/02/12 19:55:20
                                                  Addr: FEC9:C2D4::206:2AFF:FE71:4400
      5      0          0       5        300    74 2002/02/12 19:53:57    82 2002/02/12 19:54:17
                                                  Addr: FEC9:C2D4:1::9:67:114:44
      1      0          0       1         72    41 2002/02/12 19:52:39    41 2002/02/12 19:52:39
                                                  Addr: FF02::1:FF15:17
      1      0          0       1         72   111 2002/02/12 19:56:24   111 2002/02/12 19:56:24
                                                  Addr: FF02::1:FF66:C800
      1      0          0       1         72    89 2002/02/12 19:55:04    89 2002/02/12 19:55:04
                                                  Addr: FF02::1:FF71:4400
      9      0          0       9        648    56 2002/02/12 19:53:17    66 2002/02/12 19:53:31
                                                  Addr: FF02::1:FFDC:217C
    198    103       8293      95       6788 Total
    20 IP Address(s) found
Qos Report
  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Qos
      5      5        280       0          0    18 2002/02/12 19:51:54    27 2002/02/12 19:52:21 6(Internetwork)
      1      1         60       0          0   100 2002/02/12 19:55:20   100 2002/02/12 19:55:20 96(Unknown)
      6      6        432       0          0    77 2002/02/12 19:54:02   116 2002/02/12 19:56:29 112(Unknown)
     12     12        772       0          0 Total
     3 Qos(s) found
 7  Tcp Port Report
  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Tcp Port
      4      2        120       2        160    99 2002/02/12 19:55:20   102 2002/02/12 19:55:46 21(ftp)
      2      1         60       1         80    99 2002/02/12 19:55:20   100 2002/02/12 19:55:20 1026()
      2      1         60       1         80   101 2002/02/12 19:55:46   102 2002/02/12 19:55:46 1027()
      8      4        240       4        320 Total
     3 Tcp Port(s) found
 8  Udp Port Report
  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Udp Port
      3      3        234       0          0     1 2002/02/12 19:49:42     3 2002/02/12 19:49:44 137(netbios-ns)
      2      2        465       0          0    83 2002/02/12 19:54:38   103 2002/02/12 19:55:48 138(netbios-dgm)
     21     21       1491       0          0     4 2002/02/12 19:49:48   122 2002/02/12 19:57:45 161(snmp)
      3      3        213       0          0     4 2002/02/12 19:49:48     8 2002/02/12 19:49:57 1035()
      3      3        213       0          0    10 2002/02/12 19:51:06    14 2002/02/12 19:51:15 1036()
      3      3        213       0          0    34 2002/02/12 19:52:24    38 2002/02/12 19:52:33 1037()
      3      3        213       0          0    68 2002/02/12 19:53:42    72 2002/02/12 19:53:51 1038()
      3      3        213       0          0    84 2002/02/12 19:55:00    97 2002/02/12 19:55:09 1039()
      3      3        213       0          0   106 2002/02/12 19:56:18   114 2002/02/12 19:56:27 1040()
      3      3        213       0          0   118 2002/02/12 19:57:36   122 2002/02/12 19:57:45 1041()
     27      0          0      27       1440    17 2002/02/12 19:51:54    95 2002/02/12 19:55:09 32810()
      7      0          0       7        380    17 2002/02/12 19:51:54    88 2002/02/12 19:55:04 33435()
      7      0          0       7        380    19 2002/02/12 19:51:54    92 2002/02/12 19:55:04 33436()
      7      0          0       7        380    20 2002/02/12 19:51:59    95 2002/02/12 19:55:09 33437()
      3      0          0       3        160    28 2002/02/12 19:52:21    81 2002/02/12 19:54:12 33438()
      2      0          0       2        100    30 2002/02/12 19:52:21    82 2002/02/12 19:54:17 33439()
      1      0          0       1         40    32 2002/02/12 19:52:21    32 2002/02/12 19:52:21 33440()
    101     47       3681      54       2880 Total
    17 Udp Port(s) found
Protocol Summary Report
```

```
              Input               Output               Total
Protocol Packets      Bytes  Packets      Bytes  Packets      Bytes
Tcp            2        120        2        160        4        280
Udp           26       2190       27       1440       53       3630
Icmp          46       3332       20       1884       66       5216
Other          0          0        0          0        0          0
Session Summary Report
 Input Output  First yyyy/mm/dd hh.mm.ss   Last yyyy/mm/dd hh.mm.ss Protocol
     5      1     16 2002/02/12 19:51:17     27 2002/02/12 19:52:21 ICMP     Lcl:      9.67.115.5-0
                                                                             Rmt:      9.67.115.1-0
     0      1     17 2002/02/12 19:51:54     17 2002/02/12 19:51:54 UDP      Lcl:      9.67.115.5-32810
                                                                             Rmt:      9.67.115.1-33435
     0      1     19 2002/02/12 19:51:54     19 2002/02/12 19:51:54 UDP      Lcl:      9.67.115.5-32810
                                                                             Rmt:      9.67.115.1-33436
     0      1     20 2002/02/12 19:51:59     20 2002/02/12 19:51:59 UDP      Lcl:      9.67.115.5-32810
                                                                             Rmt:      9.67.115.1-33437
    21      0      5 2002/02/12 19:49:48    123 2002/02/12 19:57:45 ICMP     Lcl:      9.67.115.5-0
                                                                             Rmt:      9.67.115.5-0
     3      0      4 2002/02/12 19:49:48      8 2002/02/12 19:49:57 UDP      Lcl:      9.67.115.5-161
                                                                             Rmt:      9.67.115.5-1035
     3      0     10 2002/02/12 19:51:06     14 2002/02/12 19:51:15 UDP      Lcl:      9.67.115.5-161
                                                                             Rmt:      9.67.115.5-1036
     3      0     34 2002/02/12 19:52:24     38 2002/02/12 19:52:33 UDP      Lcl:      9.67.115.5-161
                                                                             Rmt:      9.67.115.5-1037
     3      0     68 2002/02/12 19:53:42     72 2002/02/12 19:53:51 UDP      Lcl:      9.67.115.5-161
                                                                             Rmt:      9.67.115.5-1038
     3      0     84 2002/02/12 19:55:00     97 2002/02/12 19:55:09 UDP      Lcl:      9.67.115.5-161
                                                                             Rmt:      9.67.115.5-1039
     3      0    106 2002/02/12 19:56:18    114 2002/02/12 19:56:27 UDP      Lcl:      9.67.115.5-161
                                                                             Rmt:      9.67.115.5-1040
     3      0    118 2002/02/12 19:57:36    122 2002/02/12 19:57:45 UDP      Lcl:      9.67.115.5-161
                                                                             Rmt:      9.67.115.5-1041
     3      0     29 2002/02/12 19:52:21     33 2002/02/12 19:52:21 ICMP     Lcl:      9.67.115.5-0
                                                                             Rmt:     9.67.115.69-0
     0      1     22 2002/02/12 19:52:21     22 2002/02/12 19:52:21 UDP      Lcl:      9.67.115.5-32810
                                                                             Rmt:     9.67.115.69-33435
     0      1     24 2002/02/12 19:52:21     24 2002/02/12 19:52:21 UDP      Lcl:      9.67.115.5-32810
                                                                             Rmt:     9.67.115.69-33436
     0      1     26 2002/02/12 19:52:21     26 2002/02/12 19:52:21 UDP      Lcl:      9.67.115.5-32810
                                                                             Rmt:     9.67.115.69-33437
     0      1     28 2002/02/12 19:52:21     28 2002/02/12 19:52:21 UDP      Lcl:      9.67.115.5-32810
                                                                             Rmt:     9.67.115.69-33438
     0      1     30 2002/02/12 19:52:21     30 2002/02/12 19:52:21 UDP      Lcl:      9.67.115.5-32810
                                                                             Rmt:     9.67.115.69-33439
     0      1     32 2002/02/12 19:52:21     32 2002/02/12 19:52:21 UDP      Lcl:      9.67.115.5-32810
                                                                             Rmt:     9.67.115.69-33440
     3      0      1 2002/02/12 19:49:42      3 2002/02/12 19:49:44 UDP      Lcl:     9.67.115.63-137
                                                                             Rmt:     9.67.115.17-137
     2      0     83 2002/02/12 19:54:38    103 2002/02/12 19:55:48 UDP      Lcl:     9.67.115.63-138
                                                                             Rmt:     9.67.115.17-138
     0      1     55 2002/02/12 19:53:17     55 2002/02/12 19:53:17 UDP      Lcl: FE80::6:2900:EDC:217C-32810
                                                                             Rmt: FE80::6:2900:ADC:217C-33435
     0      1     59 2002/02/12 19:53:22     59 2002/02/12 19:53:22 UDP      Lcl: FE80::6:2900:EDC:217C-32810
                                                                             Rmt: FE80::6:2900:ADC:217C-33436
     0      1     63 2002/02/12 19:53:27     63 2002/02/12 19:53:27 UDP      Lcl: FE80::6:2900:EDC:217C-32810
                                                                             Rmt: FE80::6:2900:ADC:217C-33437
     0      1     67 2002/02/12 19:53:32     67 2002/02/12 19:53:32 UDP      Lcl: FE80::6:2900:EDC:217C-32810
                                                                             Rmt: FE80::6:2900:ADC:217C-33438
     0      2     46 2002/02/12 19:52:44    105 2002/02/12 19:55:51 ICMPv6   Lcl: FE80::6:2900:EDC:217C-0
                                                                             Rmt: FE80::202:55FF:FE64:2DE7-0
     0      1    117 2002/02/12 19:56:29    117 2002/02/12 19:56:29 ICMPv6   Lcl: FE80::6:2900:EDC:217C-0
                                                                             Rmt: FE80::206:2AFF:FE66:C800-0
     2      3     76 2002/02/12 19:54:02     94 2002/02/12 19:55:09 ICMPv6   Lcl: FE80::6:2900:EDC:217C-0
                                                                             Rmt: FE80::206:2AFF:FE71:4400-0
     0      9     56 2002/02/12 19:53:17     66 2002/02/12 19:53:31 ICMPv6   Lcl: FE80::6:2900:EDC:217C-0
                                                                             Rmt: FF02::1:FFDC:217C-0
     2      0     45 2002/02/12 19:52:44    104 2002/02/12 19:55:51 ICMPv6   Lcl: FEC9:C2D4::6:2900:EDC:217C-0
                                                                             Rmt: FE80::202:55FF:FE64:2DE7-0
     1      0    116 2002/02/12 19:56:29    116 2002/02/12 19:56:29 ICMPv6   Lcl: FEC9:C2D4::6:2900:EDC:217C-0
                                                                             Rmt: FE80::206:2AFF:FE66:C800-0
     1      0     93 2002/02/12 19:55:09     93 2002/02/12 19:55:09 ICMPv6   Lcl: FEC9:C2D4::6:2900:EDC:217C-0
                                                                             Rmt: FE80::206:2AFF:FE71:4400-0
     1      1    101 2002/02/12 19:55:46    102 2002/02/12 19:55:46 TCP      Lcl: FEC9:C2D4::6:2900:EDC:217C-1027
                                                                             Rmt: FEC9:C2D4::9:67:115:17-21
     0      1     40 2002/02/12 19:52:39     40 2002/02/12 19:52:39 UDP      Lcl: FEC9:C2D4::6:2900:EDC:217C-32810
                                                                             Rmt: FEC9:C2D4::9:67:115:17-33435
     0      1     44 2002/02/12 19:52:39     44 2002/02/12 19:52:39 UDP      Lcl: FEC9:C2D4::6:2900:EDC:217C-32810
                                                                             Rmt: FEC9:C2D4::9:67:115:17-33436
     0      1     47 2002/02/12 19:52:44     47 2002/02/12 19:52:44 UDP      Lcl: FEC9:C2D4::6:2900:EDC:217C-32810
                                                                             Rmt: FEC9:C2D4::9:67:115:17-33437
     3      0     42 2002/02/12 19:52:39     48 2002/02/12 19:52:44 ICMPv6   Lcl: FEC9:C2D4::6:2900:EDC:217C-0
                                                                             Rmt: FEC9:C2D4::9:67:115:17-0
```

```
      2      1    110 2002/02/12 19:56:24    113 2002/02/12 19:56:24 ICMPv6   Lcl: FEC9:C2D4::6:2900:EDC:217C-0
                                                                              Rmt: FEC9:C2D4::206:2AFF:FE66:C800-0
      1      1     99 2002/02/12 19:55:20    100 2002/02/12 19:55:20 TCP      Lcl: FEC9:C2D4::6:2900:EDC:217C-1026
                                                                              Rmt: FEC9:C2D4::206:2AFF:FE71:4400-21
      0      1     88 2002/02/12 19:55:04     88 2002/02/12 19:55:04 UDP      Lcl: FEC9:C2D4::6:2900:EDC:217C-32810
                                                                              Rmt: FEC9:C2D4::206:2AFF:FE71:4400-33435
      0      1     92 2002/02/12 19:55:04     92 2002/02/12 19:55:04 UDP      Lcl: FEC9:C2D4::6:2900:EDC:217C-32810
                                                                              Rmt: FEC9:C2D4::206:2AFF:FE71:4400-33436
      0      1     95 2002/02/12 19:55:09     95 2002/02/12 19:55:09 UDP      Lcl: FEC9:C2D4::6:2900:EDC:217C-32810
                                                                              Rmt: FEC9:C2D4::206:2AFF:FE71:4400-33437
      3      0     90 2002/02/12 19:55:04     96 2002/02/12 19:55:09 ICMPv6   Lcl: FEC9:C2D4::6:2900:EDC:217C-0
                                                                              Rmt: FEC9:C2D4::206:2AFF:FE71:4400-0
      0      1     74 2002/02/12 19:53:57     74 2002/02/12 19:53:57 UDP      Lcl: FEC9:C2D4::6:2900:EDC:217C-32810
                                                                              Rmt: FEC9:C2D4:1::9:67:114:44-33435
      0      1     75 2002/02/12 19:54:02     75 2002/02/12 19:54:02 UDP      Lcl: FEC9:C2D4::6:2900:EDC:217C-32810
                                                                              Rmt: FEC9:C2D4:1::9:67:114:44-33436
      0      1     78 2002/02/12 19:54:07     78 2002/02/12 19:54:07 UDP      Lcl: FEC9:C2D4::6:2900:EDC:217C-32810
                                                                              Rmt: FEC9:C2D4:1::9:67:114:44-33437
      0      1     81 2002/02/12 19:54:12     81 2002/02/12 19:54:12 UDP      Lcl: FEC9:C2D4::6:2900:EDC:217C-32810
                                                                              Rmt: FEC9:C2D4:1::9:67:114:44-33438
      0      1     82 2002/02/12 19:54:17     82 2002/02/12 19:54:17 UDP      Lcl: FEC9:C2D4::6:2900:EDC:217C-32810
                                                                              Rmt: FEC9:C2D4:1::9:67:114:44-33439
      0      1     41 2002/02/12 19:52:39     41 2002/02/12 19:52:39 ICMPv6   Lcl: FEC9:C2D4::6:2900:EDC:217C-0
                                                                              Rmt: FF02::1:FF15:17-0
      0      1    111 2002/02/12 19:56:24    111 2002/02/12 19:56:24 ICMPv6   Lcl: FEC9:C2D4::6:2900:EDC:217C-0
                                                                              Rmt: FF02::1:FF66:C800-0
      0      1     89 2002/02/12 19:55:04     89 2002/02/12 19:55:04 ICMPv6   Lcl: FEC9:C2D4::6:2900:EDC:217C-0
                                                                              Rmt: FF02::1:FF71:4400-0
      0      1     49 2002/02/12 19:52:58     49 2002/02/12 19:52:58 UDP      Lcl: FEC9:C2D4::9:67:115:5-32810
                                                                              Rmt: FEC9:C2D4::9:67:115:5-33435
      0      1     51 2002/02/12 19:52:58     51 2002/02/12 19:52:58 UDP      Lcl: FEC9:C2D4::9:67:115:5-32810
                                                                              Rmt: FEC9:C2D4::9:67:115:5-33436
      0      1     53 2002/02/12 19:52:58     53 2002/02/12 19:52:58 UDP      Lcl: FEC9:C2D4::9:67:115:5-32810
                                                                              Rmt: FEC9:C2D4::9:67:115:5-33437
      3      0     50 2002/02/12 19:52:58     54 2002/02/12 19:52:58 ICMPv6   Lcl: FEC9:C2D4::9:67:115:5-0
                                                                              Rmt: FEC9:C2D4::9:67:115:5-0
 9     55 session(s) found
 10    123 records processed for this report
 11  Recording ended at   2002/02/12 19:57:45.836155
Recording started at 2002/02/12 19:49:42.788207
The duration was             00:08:03.047947
 12    304 is the maximum packet data length
 13  16384 bytes of storage used to create this report
   123 requests for 14992 bytes of storage were made
```

1

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.

2
The totals by record type (Packet trace, X25, and data trace).

3
The totals by device type for packet trace records.

4
The totals by Interface or Link Name for packet trace records.

5
The totals by Protocol number for packet trace records.

6
The totals by IP Address. Both the destination and source IP addresses are counted except when they are the same with in a record.

7
The totals by TCP Port number. Both the destination and source port numbers are counted except when they are the same within a record.

8
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.

9
The totals by session partner pairs (IP addresses, protocol number, and port numbers).

10
The number of records processed for the statistics report.

11
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.

12
The size of the largest packet found in the input file.

13
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

-------------------------------------------------------------------------------
 2  Session: FEC9:C2D4::206:2AFF:FE71:4400-0 FEC9:C2D4::6:2900:EDC:217C-0 ICMPv6
 From: 2002/02/12 19:55:04.615118 to: 2002/02/12 19:55:09.636234
      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  D Rcd #            Time           Delta      Seq #   Position Length    End_Pos
I    90 19:55:04.615118 00:00:00.000000          0          0     24         24
000000 FEC9C2D4 00000000 02062AFF FE714400  02010006 2A714400                   |..........*..qD.....*qD.        |
I    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.|
000040 802A829B 0014A3B6 01010000 3C697318  00095CAB                            |.*..........<is.....
```

1
After all the records have processed, the number of streams and the amount of storage required for the report and buffers are shown.

2
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.

3
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
//IPCSPRNT  DD SYSOUT=*
//IPCSTOC   DD SYSOUT=*
//IPCSDDIR  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* 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 or onetstat command to display data traces. Figure 21 shows a data trace for a single entry.

```
onetstat -p TCPCS -f
...
Data Trace Setting:
JobName: *          TrRecCnt: 00000006  Length: FULL
IpAddr:  *                SubNet: *
```

Figure 21. Data trace: Single entry

Figure 22 shows a data trace for multiple entries.

```
netstat -p TCPCS -f
...
Data Trace Setting:
JobName: MEGA4      TrRecCnt: 00000000  Length: FULL
IpAddr:  127.0.0.3        SubNet: *

JobName: *          TrRecCnt: 00000000  Length: FULL
IpAddr:  127.0.0.9        SubNet: *
```

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 ENTIDLIST parameter to isolate data trace records and packet trace records from each other. See "Formatting packet traces using IPCS" on page 89 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 CTIIDS*xx* PARMLIB member. In the following example, the trace searches for PARMLIB member CTIIDSA3.

```
S 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.

See Chapter 27, “Diagnosing intrusion detection problems,” on page 631 for additional information about diagnosing policy problems.

```
/*******************************************************************/
/*                                                                 */
/*  IBM Communications Server for z/OS                             */
/*  SMP/E Distribution Name: CTIIDS00                              */
/*                                                                 */
/*  MEMBER:  CTIIDS00                                              */
/*                                                                 */
/*                                                                 */
/*  Copyright:                                                     */
/*              Licensed Materials - Property of IBM               */
/*              5694-A01                                           */
/*              (C) Copyright IBM Corp. 2001, 2003                 */
/*                                                                 */
/*                                                                 */
/*  Status:     CSV1R5                                             */
/*                                                                 */
/*                                                                 */
/*  DESCRIPTION = This parmlib member causes IDS component trace   */
/*              for the TCP/IP product to be initialized with a    */
/*              trace buffer size of 32M.                          */
/*                                                                 */
/*              This parmlib members only lists those TRACEOPTS    */
/*              value specific to TCP/IP.  For a complete list     */
/*              of TRACEOPTS keywords and their values see         */
/*              z/OS MVS INITIALIZATION AND TUNING REFERENCE.      */
/*                                                                 */
/* $MAC(CTIIDS00) PROD(TCPIP): Component Trace SYS1.PARMLIB member  */
/*                                                                 */
/*                                                                 */
/*******************************************************************/
 TRACEOPTS
 /* ----------------------------------------------------------------- */
 /*   ON OR OFF: PICK 1                                               */
 /* ----------------------------------------------------------------- */
          ON
 /*       OFF                                                         */
 /* ----------------------------------------------------------------- */
 /*   BUFSIZE: A VALUE IN RANGE 16M TO 256M                           */
 /* ----------------------------------------------------------------- */
          BUFSIZE(32M)
 /*       WTR(wtr_procedure) WRAP|NOWRAP                              */
```

Figure 23. SYS1.PARMLIB member CTIIDS00

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.

SYSTCPIS OPTIONS

The syntax for the OPTIONS component routine parameters is:

OPTIONS component:

```
►►──OPTIONS──((──┤ Data Selection ├──┤ Report Generation ├──))──────────────────►◄
```

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 ├────┴──)──BROADCAST──CLASSA──CLASSB──CLASSC──CLASSD──────►

                             ┌──────────────────┐                ┌─────────────────┐
                             ▼                  │                ▼                 │
►──CLASSE──HOST──IPADDR──(───┤ IP Address ├─────┴──)──IPID──(────ip_id_number──────┴──)──►

                    ┌─────────────────────────┐
                    ▼                         │
►──LOOPBACK──QOS──(───quality_of_service──────┴──)──────────────────────────────────┤
```

IP Address:

```
├──┬──ddd.ddd.ddd.ddd──┬─────────────────────┬──┬──┬─────────┬──────────────────────┤
   │                   ├──/mmm.mmm.mmm.mmm───┤  │  └──-port──┘
   │                   └──/nn────────────────┘  │
   ├──┬──A──┬───────────────────────────────────┤
   │  ├──B──┤                                   │
   │  ├──C──┤                                   │
   │  ├──D──┤                                   │
   │  ├──E──┤                                   │
   │  ├──H──┤                                   │
   │  ├──L──┤                                   │
   │  ├──M──┤                                   │
   │  ├──O──┤                                   │
   │  └──*──┘                                   │
   └──X'hhhhhhhh'───────────────────────────────┘
```

Name:

```
                 ┌──────────────┐                       ┌────────────┐
                 ▼              │                       ▼            │
├──LINKNAME──(───linkname───────┴──)──┬──JOBLIST──(─────jobname──────┴──)──┬──────────┤
                                      └──JOBNAME───────────────────────────┘
```

Port Number:

```
├──┬──BOOTP──┬──┬────────────────────────────────────────────────────────┬─────────────►
   └──DHCP───┘  │         ┌──67────────────┐    ┌──68────────────┐      │
                └──(──────┴──port_number───┴────┴──port_number───┴──)───┘
```

```
►──┬─DNS────┬──┬─────────────────────────┬──FINGER──┬─────────────────────────┬──────►
   └─DOMAIN─┘  │      ┌─53──────────┐    │          │      ┌─79──────────┐    │
               └─(────┴─port_number─┴──)─┘          └─(────┴─port_number─┴──)─┘

►──FTP──┬────────────────────────────────────────────────┬───────────────────────────►
        │      ┌─20──────────┐ ┌─21──────────┐           │
        └─(────┴─port_number─┴─┴─port_number─┴─────)─────┘

►──GOPHER──┬─────────────────────────┬──NTP──┬─────────────────────────┬─────────────►
           │      ┌─70──────────┐    │       │      ┌─123─────────┐    │
           └─(────┴─port_number─┴──)─┘       └─(────┴─port_number─┴──)─┘

►──┬─HTTP─┬──┬─────────────────────────┬──IKE──┬─────────────────────────┬───────────►
   └─WWW──┘  │      ┌─80──────────┐    │       │      ┌─3860────────┐    │
             └─(────┴─port_number─┴──)─┘       └─(────┴─port_number─┴──)─┘

                  ┌─────────────┐
                  ▼             │
►──PORT──(────────port_number───┴──)──┬─RIP────┬──┬─────────────────────────┬────────►
                                      └─ROUTER─┘  │      ┌─520─────────┐    │
                                                  └─(────┴─port_number─┴──)─┘

►──RPC──┬─────────────────────────┬──SMTP──┬─────────────────────────┬───────────────►
        │      ┌─111─────────┐    │        │      ┌─25──────────┐    │
        └─(────┴─port_number─┴──)─┘        └─(────┴─port_number─┴──)─┘

►──SNMP──┬───────────────────────────────────────────────┬───────────────────────────►
         │      ┌─161─────────┐ ┌─162─────────┐          │
         └─(────┴─port_number─┴─┴─port_number─┴─────)────┘

►──TELNET──┬──────────────────────────────────────────────────────────────┬──────────►
           │      ┌─23──────────┐ ┌─80───────────┐                        │
           └─(────┴─port_number─┴─┴─screen_width─┴──┬───────────┬───)────┘
                                                    ├─SUMMARY───┤
                                                    └─DETAIL────┘

►──TFTP──┬─────────────────────────┬──TIME──┬─────────────────────────┬──────────────┤
         │      ┌─69──────────┐    │        │      ┌─37──────────┐    │
         └─(────┴─port_number─┴──)─┘        └─(────┴─port_number─┴──)─┘
```

Protocol:

```
                                ┌──────────────────┐
                                ▼                  │
├──GRE──ICMP──IGMP──OSPFI──PROTOCOL(──protocol_number──)──TCP──UDP──────────────┤
```

Record Number:

```
           ┌──────────────────┐
           ▼                  │
├──RECORDS(──record_number──)──────────────────────────────────────────────────┤
```

Record Identifiers:

```
       ┌─────────────────────────────────┐
       ▼                                 │
├──CID(──communication identifier──)────────────────────────────────────────────►

              ┌─────────────────────────────┐          ┌───────────────────────┐
              ▼                             │          ▼                       │
►──CORRELATOR(──correlator identifier──)──GROUP(──group identifier──)───────────►

            ┌─────────────────────┐            ┌─────────────┐
            ▼                     │            ▼             │
►──INSTANCE(──instance number──)──PROBEID(──probe id──)─────────────────────────►

        ┌───────────────┐
        ▼               │
►──TYPE(──probe type──)─────────────────────────────────────────────────────────┤
```

Record Type:

```
├──FLAGS──(──ABBREV──ACK──DATA──DF──FIC──FIN──FULL──HOME──IN──IPO──────────────►

►──LIC──MIC──OIC──OUT──PING──PSH──RSM──RST──SEG──SYN──TCPO──TOS────────────────►

►──TUNNEL──URG──ZWIN──)──USEREXIT(exitname)─────────────────────────────────────┤
```

Report Generation:

```
                                                          ┌──────┐
                                                          ▼      │
├──┬─BOTH────┬──BASIC──CLEANUP──┬──────────────────┬──DEBUG(──nn──)────────────►
   ├─EBCDIC──┤                  │     ┌─500────┐   │
   ├─ASCII───┤                  └─(───┴─nnnnn──┴─)─┘
   └─HEX─────┘

           ┌─200───┐
►──DELAYACK(┴─nnnn──┴─)──DUMP──┬──────────────────┬──EXPORT──┬───────────────────┬──►
                               │    ┌─65535──┐    │          │    ┌─SUMMARY─┐    │
                               └─(──┴─nnnnn──┴─)──┘          └─(──┴─DETAIL──┴─)──┘
```

```
                                          ┌─125────┐   ┌─150─────┐
►──┬─FMT────┬─┬────────────────────┬─FULL──GAIN(─┴─rtgain─┴─,─┴─vargain─┴─)──────►
   └─FORMAT─┘ │      ┌─DETAIL──┐   │
              └─(────┴─SUMMARY─┴─)─┘

            ┌─NOREASSEMBLY─────────────────────────────────┐ ┌─SEGMENT───┐
►─NOT──OPTION──┴─REASSEMBLY─┬────────────────────────────────┬─┴─┴─NOSEGMENT─┴──────►
                        │    ┌─32767─┐  ┌─SUMMARY─┐      │
                        └─(──┴─nnnnn─┴──┴─DETAIL──┴──)───┘

►─SESSION─┬────────────────────┬─SNIFFER─┬──────────────────────────────────┬─────►
          │    ┌─DETAIL──┐     │         │    ┌─200───┐   ┌─ETHERNET──┐     │
          └─(──┼─SUMMARY─┼──)──┘         └─(──┴─nnnnn─┴───┼─TCPDUMP───┼──)──┘
               └─STATE───┘                                └─TOKENRING─┘

          ┌─10────┐   ┌─10─────┐
►─SPEED(──┴─local─┴─,─┴─remote─┴─)──┬─┬─STATISTICS─┬─┬──────────────────────┬─┬──►
                                    │ └─STATS──────┘ │    ┌─SUMMARY─┐       │ │
                                    │                └─(──┴─DETAIL──┴──)────┘ │
                                    └─TALLY───────────────────────────────────┘

►─STREAMS─┬──────────────────────────────────┬─SUMMARY──TOD──────────────────────┤
          │    ┌─128───┐   ┌─SUMMARY─┐       │
          └─(──┴─nnnnn─┴───┴─DETAIL──┴──)────┘
```

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 (nnnnnnnnnn) 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).

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 NOREASSEMBLY 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:

- TCPTR
- 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 107.

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	A class A address, 0.0.0.0/128.0.0.0
B	A class B address, 128.0.0.0/192.0.0.0
C	A class C address, 192.0.0.0/224.0.0.0
D	A class D address, 224.0.0.0/240.0.0.0
E	A class E address, 240.0.0.0/248.0.0.0
H	A local host address, 0.0.0.0/0.0.255.255
L	A loopback address, 127.0.0.0/255.0.0.0
M	The 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 recombined. 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.

OSPFI

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'hhhhhhhh'). 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 (nnnnnnnnn) 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[(*nnnnn*|200, ETHERNET|TCPDUMP|TOKENRING)]
Writes the trace records in a format acceptable for downloading to other trace analysis programs, such as Network Associates' Sniffer Network Analyzer or programs from http://www.tcpdump.

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 http://www.tcpdump files with an Ethernet header.

TOKENRING
If this keyword is specified, the output is formatted for the token-ring analysis application of the analyzer. This keyword specifies the file format only and does not imply that only packets traced on a token ring are collected. Packets from all devices can be collected using this option.

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 7954 (8000 - 46) for SNIFFER(TOKENRING) and the maximum 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:

Identifier
Type

0100
TCPTR

0200
UDPTR

0301
VSSCAN

0303
NORMSCAN

0401
MALFORMED

0402
RAW

0403
IPFRAGMENT

0404
ICMP

0405
IPOPT

0406
IPPROTO

0407
FLOOD

0408
PREPECHO

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 SYSTCPIS trace record filtering is done. If this exit routine returns a nonzero return code, then the record is skipped by the SYSTCPIS 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(SYSTCPIS) SUB((TCPCS)) SUMMARY DSN('IBMUSER.CTRACE1')
```

Examples:

```
  COMPONENT TRACE SUMMARY FORMAT

  SYSNAME(MVS118)

  COMP(SYSTCPIS)SUBNAME((TCPCS))

  DSNAME('IBMUSER.CTRACE1')

 **** 2002/11/20
 I - Inbound packet

 O - Outbound packet

 DP   Nr hh:mm:ss.mmmmmm IpId Group    Probe Id Corelatr JobName  Cid      DatLn Data Source/Destination
 II 4521 17:38:32.175560 0000 SCAN     03030000       10 TCPCS    00000000    12 ICMP  9.42.105.71
                                                                                       9.42.104.38
 IT 4522 17:38:45.130339 163F SCAN     03030026       11 FTPD1    00000020     0 TCP   9.2.197.34-46911
                                                                                       9.42.104.38-21
 IT 4523 17:38:45.153474 173F SCAN     03030026       12 FTPD1    00000020     0 TCP   9.224.157.220-47167
                                                                                       9.42.104.38-21
 IT 4524 17:38:45.170441 183F SCAN     03030026       13 FTPD1    00000020     0 TCP   9.74.208.131-47423
                                                                                       9.42.104.38-21
 IT 4525 17:38:45.190606 193F SCAN     03030026       14 FTPD1    00000020     0 TCP   9.79.235.253-47679
                                                                                       9.42.104.38-21
 IT 4526 17:38:45.213117 1A3F SCAN     03030026       15 FTPD1    00000020     0 TCP   9.40.107.43-47935
                                                                                       9.42.104.38-21
IT 5671 17:59:32.787165 0B3B ATTACK   04070002      277 FTPD1    00000020     0 TCP    9.42.104.38-21
                                                                                       9.84.160.95-47938
IT 5672 17:59:32.806700 0B1A ATTACK   04070002      277 FTPD1    00000020     0 TCP    9.42.104.38-21
                                                                                       9.156.214.250-44610
IT 5673 17:59:32.827193 0B1B ATTACK   04070002      277 FTPD1    00000020     0 TCP    9.42.104.38-21
                                                                                       9.150.148.96-44866
IT 5674 17:59:32.847730 0B1C ATTACK   04070002      277 FTPD1    00000020     0 TCP    9.42.104.38-21
                                                                                       9.48.42.177-45122
.
.
.

================================================================================
SYSTCPIS 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 TCPTR.

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)
  ))

1  **** 2002/11/20
RcdNr Sysname  Mnemonic Entry Id   Time Stamp     Description
----- -------- -------- -------- --------------- --------------------------------

-------------------------------------------------------------------------------
2   4521 MVS118   SCAN     03030000 17:38:32.175560 Scan-Normal packet
3   From Link         : ETH1             Device: LCS Ethernet    Full=40
  Tod Clock        : 2002/11/20 17:38:32.175559             Module: EZBIPICM
  Job Name         : TCPCS            Asid: 01F7             Tcb: 00000000
  Cid              : 00000000         Correlator: 10
  Policy           : ScanEventIcmp-rule
4   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

5   ICMP
  Type/Code        : ECHO             CheckSum: 5592 FFFF
  Id               : 0B3F             Seq: 0
6    Echo Data        : 12
000000 AEBCDB3D 03340A00 00000000                                  |...=.4......|

-------------------------------------------------------------------------------
 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
```

```
  Source Port      : 46911 ()       Destination Port: 21    (ftp)
  Sequence Number  : 2397868413     Ack Number: 0
  Header Length    : 20             Flags: Syn
  Window Size      : 242            CheckSum: 4E53 B695 Urgent Data Pointer: 0000
.
.
.
 ===============================================================================
 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 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.

Port	Keyword
67, 68	BOOTP
67, 68	DHCP
53	Domain
79	Finger
70	Gopher
520	Rip
520	Router
111	RFC
25	SMTP
23	TELNET
69	TFTP
37	TIME

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

-------------------------------------------------------------------------------
 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: EZBIPICM
  Job Name         : TCPCS            Asid: 01F7              Tcb: 00000000
  Cid              : 00000000         Correlator: 10
  Policy           : ScanEventIcmp-rule

1 IP Header          : 20
 000000 45000028 00004000 3E015914 092A6947  092A6826

2 Protocol Header    : 8
 000000 08005592 0B3F0000

3 Data               : 12     Data Length: 12
 000000 AEBCDB3D 03340A00 00000000          |............     ...=.4......    |

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

 IP Header         : 20
 000000 45000028 163F0000 FD06681C 0902C522  092A6826

 Protocol Header    : 20
 000000 B73F0015 8EEC917D 00000000 500200F2  4E530000

-------------------------------------------------------------------------------
 4523 MVS118   SCAN     03030026 17:38:45.153474 Scan Normal-TCP SYN  dropped
 From Link         : UNKNOWN          Device: Unknown:0       Full=40
  Tod Clock        : 2002/11/20 17:38:45.153473               Module: EZBTCPCN
  Job Name         : FTPD1            Asid: 01F7              Tcb: 00000000
  Cid              : 00000020         Correlator: 12
  Policy           : ScanEventHigh-rule
 IP Header         : 20
 000000 45000028 173F0000 FD068D84 09E09DDC  092A6826

 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.

SNIFFER

Purpose: This report shows information written to the SNIFFER 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 NOREASSEMBLY))
  DSNAME('MWS.PQ33208.PTRACE4')
 PTRPT04I SNIFFER(TOKENRING) option selected

 OPTIONS((
   Statistics PacketTrace X25 Sniffer Option Tokenring
   Skip(0) Limit(999999999) Dump(65535) Both Interval(5) MaxRsm(32767)
   Stream(131072)
   Bootp(67,68) Domain(53) Finger(79) Ftp(20,21)
   Gopher(70) Router(520) Rpc(111) SMTP(25)
   SNMP(161,162) Telnet(23) Tftp(69) Time(37) WWW(80)
   User(        )
   ))
     108 records written to SNIFFER
  46,000 bytes written to SNIFFER
      22 records were truncated to 1600 bytes
```

The following describes areas of the example.

108 records written to SNIFFER
This record count includes the two header records and one trailer record that were written to the SNIFFER data set.

46 000 bytes written to SNIFFER
The number of data bytes written to the SNIFFER 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.

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
 SYSNAME(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) 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) Statistics(Detail) Telnet(23) Tftp(69) Time(37)
 Userexit() Www(80)
  ))

**** 2002/11/20

===============================================================================
1  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
===============================================================================
2  Probe Report

   Total  Input    Data Output     Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Probe
    1526   1526     67144      0        0  4893 2002/11/20 17:56:00  7143 2002/11/20 18:09:17 03010021
       1      1        40      0        0  5652 2002/11/20 17:57:36  5652 2002/11/20 17:57:36 03010028
     859    859     34360      0        0  4553 2002/11/20 17:38:46  6376 2002/11/20 18:06:04 03020020
       6      6       724      0        0  4521 2002/11/20 17:38:32  5654 2002/11/20

.
.
.
    2623   2623    112084      0          0 Total

      9 Probe(s) found

3  Device Type Report

   Total  Input       Data Output   Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Device Type
     966    966      39300      0      0  4521 2002/11/20 17:38:32  6376 2002/11/20 18:06:04 1(LCS Ethernet)
     966    966      39300      0      0 Total

      1 Device Type(s) found

4  Interface Report

   Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Interface
     966    966      39300      0          0  4521 2002/11/20 17:38:32  6376 2002/11/20 18:06:04 ETH1
    1657   1657      72784      0          0  4522 2002/11/20 17:38:45  7143 2002/11/20 18:09:17 UNKNOWN
    2623   2623     112084      0          0 Total

      2 Interface(s) found

5  Interface Address Report

   Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Interface
     966    966      39300      0          0  4521 2002/11/20 17:38:32  6376 2002/11/20 18:06:04 ETH1
                                                   Addr: 9.42.104.38
    1557   1557      68384      0          0  4522 2002/11/20 17:38:45  7143 2002/11/20 18:09:17 UNKNOWN
```

```
                                                  Addr: 9.42.104.38
.
.
.
  2623   2623     112084      0          0 Total

   64 Interface Address(s) found

6 JobName Report
  Total  Input       Data Output   Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss JobName
   2610   2610     110984      0      0  4522 2002/11/20 17:38:45  7143 2002/11/20 18:09:17 FTPD1
      1      1         40      0      0  4587 2002/11/20 17:39:14  4587 2002/11/20 17:39:14 INETDCS1
      1      1        144      0      0  4591 2002/11/20 17:39:16  4591 2002/11/20 17:39:16 INETDCS3
      8      8        416      0      0  4521 2002/11/20 17:38:32  5892 2002/11/20 18:00:07 TCPCS
      1      1        123      0      0  4623 2002/11/20 17:40:48  4623 2002/11/20 17:40:48 TRMD
      2      2        377      0      0  5653 2002/11/20 17:57:37  5654 2002/11/20 17:57:37 USER17
   2623   2623     112084      0      0 Total
     6 JobName(s) found

7  Asid Report

  Total  Input       Data Output   Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Asid
   2623   2623     112084      0      0  4521 2002/11/20 17:38:32  7143 2002/11/20 18:09:17 01F7
   2623   2623     112084      0      0 Total
     1 Asid(s) found

8  Protocol Report

  Total  Input       Data Output    Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Protocol
     12     12        656      0       0  4521 2002/11/20 17:38:32  5892 2002/11/20 18:00:07 1(ICMP)
   2607   2607     110784      0       0  4522 2002/11/20 17:38:45  7143 2002/11/20 18:09:17 6(TCP)
      4      4        644      0       0  4591 2002/11/20 17:39:16  5654 2002/11/20 17:57:37 17(UDP)
   2623   2623     112084      0       0 Total   $

     3 Protocol(s) found

9  IP Address Report

  Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss
     11     11        484      0          0  6430 2002/11/20 18:09:02  7088 2002/11/20 18:09:16
                                                  Addr: 9.0.12.8
      1      1         40      0          0  4537 2002/11/20 17:38:45  4537 2002/11/20 17:38:45
                                                  Addr: 9.0.12.225
      1      1         56      0          0  5866 2002/11/20 18:00:06  5866 2002/11/20 18:00:06
                                                  Addr: 9.0.32.254
.
.
.

   5246   5246     224168      0          0 Total
   518 IP Address(s) found
10  Qos Report
  Total  Input       Data Output   Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Qos
      7      7        392      0      0  5830 2002/11/20 18:00:06  5892 2002/11/20 18:00:07 6(Internetwork)
      7      7        392      0      0 Total
     1 Qos(s) found

11  Tcp Port Report

  Total  Input       Data Output   Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Tcp Port
   2605   2605     110704      0      0  4522 2002/11/20 17:38:45  7143 2002/11/20 18:09:17 21(ftp)
      1      1         40      0      0  4743 2002/11/20 17:45:56  4743 2002/11/20 17:45:56 73(netrjs-3)
      1      1         40      0      0  5922 2002/11/20 18:00:10  5922 2002/11/20 18:00:10 74(netrjs-4)

.
.
.

  5214   5214     221568      0          0 Total

 1742 Tcp Port(s) found

12  Udp Port Report

  Total  Input       Data Output     Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Udp Port
      4      4        644      0        0  4591 2002/11/20 17:39:16  5654 2002/11/20 17:57:37 53(domain)
      1      1        144      0        0  4591 2002/11/20 17:39:16  4591 2002/11/20 17:39:16 1032()
      1      1        123      0        0  4623 2002/11/20 17:40:48  4623 2002/11/20 17:40:48 1033()
      1      1        144      0        0  5653 2002/11/20 17:57:37  5653 2002/11/20 17:57:37 1034()
      1      1        233      0        0  5654 2002/11/20 17:57:37  5654 2002/11/20 17:57:37 1035()
      8      8       1288      0        0 Total
     5 Udp Port(s) found

13 CID Report

   Total  Input       Data Output       Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss CID
     220    220       9200      0          0  4522 2002/11/20 17:38:45  5919 2002/11/20 18:00:07 00000020
       1      1         40      0          0  4553 2002/11/20 17:38:46  4553 2002/11/20 17:38:46 00000067
       1      1         40      0          0  4554 2002/11/20 17:38:46  4554 2002/11/20 17:38:46 00000096
.
.
.
    2615   2615     111668      0          0 Total

   2396 CID(s) found

14 Group Report
   Total  Input       Data Output    Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Group
    2423   2423     103508      0      0  4521 2002/11/20 17:38:32  7143 2002/11/20 18:09:17 3(SCAN)
     200    200       8576      0      0  5671 2002/11/20 17:59:32  5919 2002/11/20 18:00:07 4(ATTACK)
    2623   2623     112084      0      0 Total
      2 Group(s) found

15 Type Report
   Total  Input       Data Output    Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Type
    1527   1527      67184      0       0  4893 2002/11/20 17:56:00  7143 2002/11/20 18:09:17 0301(VSSCAN)
     859    859      34360      0       0  4553 2002/11/20 17:38:46  6376 2002/11/20 18:06:04 0302(PSSCAN)
      37     37       1964      0       0  4521 2002/11/20 17:38:32  5654 2002/11/20 17:57:37 0303(NORMSCAN)
     200    200       8576      0       0  5671 2002/11/20 17:59:32  5919 2002/11/20 18:00:07 0407(FLOOD)
    2623   2623     112084      0       0 Total
      4 Type(s) found

16 Correlator Report

   Total  Input       Data Output    Data First yyyy/mm/dd hh.mm.ss  Last yyyy/mm/dd hh.mm.ss Correlator
       4      4        644      0       0 4591 2002/11/20 17:39:16  5654 2002/11/20 17:57:37        2
       1      1         40      0       0  4521 2002/11/20 17:38:32  4521 2002/11/20 17:38:32       10
       1      1         40      0       0  4522 2002/11/20 17:38:45  4522 2002/11/20 17:38:45       11
.
.
.

    2623   2623     112084      0          0 Total

    467 Correlator(s) found

17 Protocol Summary Report
              Input             Output             Total
Protocol Packets      Bytes  Packets      Bytes  Packets      Bytes
Tcp         2607     110784        0          0     2607     110784
Udp            4        644        0          0        4        644
Icmp          12        656        0          0       12        656
Other          0          0        0          0        0          0

18 Session Summary Report
  Input Output  First yyyy/mm/dd hh.mm.ss   Last yyyy/mm/dd hh.mm.ss Protocol
      1      0   5738 2002/11/20 17:59:34   5738 2002/11/20 17:59:34 TCP      Lcl:      9.4.81.167-27970
                                                                             Rmt:     9.42.104.38-21
      1      0   5710 2002/11/20 17:59:33   5710 2002/11/20 17:59:33 TCP      Lcl:     9.5.101.147-47426
                                                                             Rmt:     9.42.104.38-21
      1      0   5748 2002/11/20 17:59:34   5748 2002/11/20 17:59:34 TCP      Lcl:      9.6.159.21-30530
                                                                             Rmt:     9.42.104.38-21
.
.
.
   2618 session(s) found

   2623 records processed for this report
 Recording ended at   2002/11/20 18:09:17.543000
 Recording started at 2002/11/20 17:38:32.175560
 The duration was                00:30:45.367440
      1 records with ABBREV=200
   2622 records with FULL=144
    233 is the maximum packet data length
 655360 bytes of storage used to create this report
   7841 requests for 652704 bytes of storage were made
```

The following describes numbered areas of the example.

1

The standard statistics shown with all executions of the SYSTCPIS packet trace formater.

- **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.

2

Probe report, which is the total by ProbeID.

3

Device type report, which is the totals by device type.

4

Interface report, which is the totals by interface.

5

Interface address report, which is the totals interface address.

6

Jobname report, which is the totals by jobname.

7

ASID report, which is the totals address space identifier.

8

Protocol report, which is the totals by protocol.

9

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.

10

QOS report, which is the totals by QOS.

11

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.

12
: 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.

13
: CID report, which is the totals by connection identifier.

14
: Group report, which is the totals by group, first byte PROBEID.

15
: Type report, which is the totals by type, first two bytes of PROBEID.

16
: Correlator report, which is the totals by correlator.

17
: Protocol summary report, which is the summary based on protocol.

18
: 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((TCPCS)) SHORT DSN('IBMUSER.CTRACE1') OPTIONS((OPT STREAM DUMP ASCII))
```

Examples:

```
COMPONENT TRACE SHORT FORMAT
 SYSNAME(MVS118)
 COMP(SYSTCPIS)SUBNAME((TCPCS))
 OPTIONS((OPT STREAM DUMP ASCII))
 DSNAME('IBMUSER.CTRACE1')

OPTIONS((Ascii 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) Streams(131072,Summary)
 Telnet(23) Tftp(69) Time(37) Userexit() Www(80)
  ))

**** 2002/11/20
RcdNr Sysname  Mnemonic 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 1F9B0000 0106EC56 092A6826  09A032EF 0015E644 7F6CBB58 |E..(.......V.*h&..2....D.l.X    |
 I  5892 18:00:07.149355 00:00:00.321697         28         28     28         56
 000000                                                                4500002C |                           E..,|
 000020 1FDC0000 0106EC11 092A6826 09A032EF  00154446 809241F5             |.........*h&..2...DF..A.       |


.
.
.
 ===============================================================================
 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
```

OMPROUTE trace (SYSTCPRT)

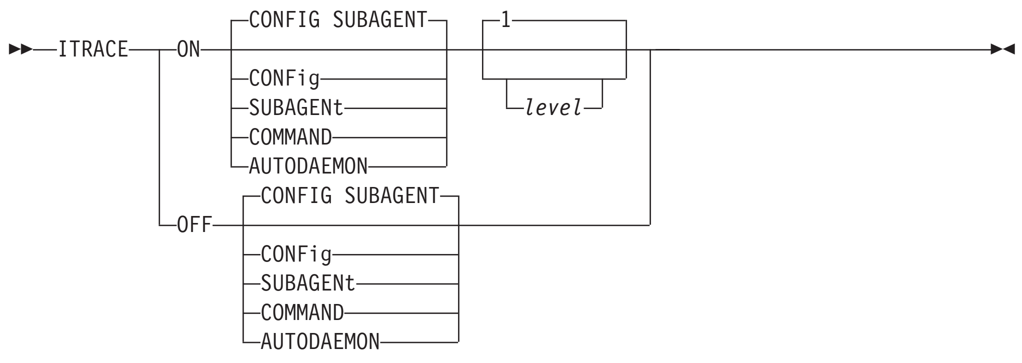
TCP/IP Services component trace is also available for use with the OMPROUTE application. See "TCP/IP services component trace for OMPROUTE" on page 684.

RESOLVER trace (SYSTCPRE)

TCP/IP Services component trace is also available for use with the RESOLVER application. See Chapter 37, "Diagnosing resolver problems," on page 761.

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 chapter describes the subcommands (including description, syntax, parameters, and sample output), installation, entering, and execution, and includes the following sections:

- "TCPIPCS command" on page 176
- "TCPIPCS subcommands" on page 179
- "ERRNO command" on page 262
- "IPHDR" on page 266
- "RESOLVER" on page 267
- "SETPRINT" on page 272
- "SKMSG" on page 273
- "TCPHDR" on page 275
- "TOD" on page 276
- "UDPHDR" on page 277
- "Installing TCP/IP IPCS subcommands" on page 278
- "Entering TCP/IP IPCS subcommands" on page 279

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 14.
- 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 14 lists all the IPCS subcommands. The TCPIPCS commands are shown first, followed by the general commands.

| *Table 14. TCP/IP IPCS commands*

Command	Description	Alias	See page
TCPIPCS ALL	Equivalent to TCPIPCS STATE TSEB TSDB TSDX DUAF CONFIG ROUTE SOCKET STREAM RAW TCB UDP LOCK TIMER STORAGE		NA
TCPIPCS API	Display control blocks for Sockets Extended Assembler Macro and Pascal APIs		"TCPIPCS API" on page 179
TCPIPCS CONFIG	Display device configuration information	TCPIPCS CNFG TCPIPCS CONF	"TCPIPCS CONFIG" on page 181

Table 14. TCP/IP IPCS commands (continued)

Command	Description	Alias	See page
TCPIPCS CONNECTION	Display active or all connections	TCPIPCS CONN	"TCPIPCS CONNECTION" on page 182
TCPIPCS COUNTERS	Display information about TCP/IP internal execution statistics		"TCPIPCS COUNTERS" on page 184
TCPIPCS DETAIL	Equivalent to TCPIPCS TSEB TSDB TSDX DUAF Available for Telnet.	TCPIPCS CBS	NA
TCPIPCS DU	Equivalent to TCPIPCS DUAF DUCB Available for Telnet.		NA
TCPIPCS DUAF	Summarize DUCBs Available for Telnet.	TCPIPCS DUCBS	"TCPIPCS DUAF" on page 186
TCPIPCS DUCB	Find and format DUCBs Available for Telnet.		"TCPIPCS DUCB" on page 188
TCPIPCS FIREWALL	Display information about Firewall filters and tunnels		"TCPIPCS FIREWALL" on page 192
TCPIPCS FRCA	Display state information about FRCA connections and objects		"TCPIPCS FRCA" on page 194
TCPIPCS HASH	Display TCP/IP data stored in hash tables		"TCPIPCS HASH" on page 196
TCPIPCS HEADER	Display dump Header info	TCPIPCS HDR	"TCPIPCS HEADER" on page 200
TCPIPCS HELP	Display syntax help for TCPIPCS command	TCPIPCS ?	"TCPIPCS HELP" on page 201
TCPIPCS IPSEC	Display information about IP security filters and tunnels		"TCPIPCS IPSEC" on page 203
TCPIPCS LOCK	Display locks Available for Telnet.	TCPIPCS LOCKSUM	"TCPIPCS LOCK" on page 205
TCPIPCS MAP	Display storage map		"TCPIPCS MAP" on page 207
TCPIPCS MTABLE	Display module table		"TCPIPCS MTABLE" on page 209
TCPIPCS POLICY	Display service policy data		"TCPIPCS POLICY" on page 211
TCPIPCS PROFILE	Display TCP/IP configuration data in the format of a profile dataset Available for Telnet.	TCPIPCS PROF	"TCPIPCS PROFILE" on page 212

Table 14. TCP/IP IPCS commands (continued)

Command	Description	Alias	See page
TCPIPCS PROTOCOL	Invokes RAW, TCB, UDP		"TCPIPCS PROTOCOL" on page 215
TCPIPCS RAW	Display RAW control blocks	TCPIPCS MRCB TCPIPCS RAWSUM TCPIPCS RCB	"TCPIPCS RAW" on page 218
TCPIPCS ROUTE	Display routing information	TCPIPCS RTE	"TCPIPCS ROUTE" on page 220
TCPIPCS SOCKET	Display socket information	TCPIPCS SCB TCPIPCS SOCKSUM	NA
TCPIPCS STATE	Display general stack information	TCPIPCS	"TCPIPCS STATE" on page 223
TCPIPCS STORAGE	Display TCP/IP storage usage	TCPIPCS STOR	"TCPIPCS STORAGE" on page 233
TCPIPCS STREAM	Display streams information	TCPIPCS SKSH TCPIPCS STREAMS	"TCPIPCS STREAM" on page 235
TCPIPCS SUMMARY	Equivalent to TCPIPCS DUAF CONFIG SOCKET		NA
TCPIPCS TCB	Display TCP protocol control blocks	TCPIPCS MTCB TCPIPCS TCBSUM	"TCPIPCS TCB" on page 237
TCPIPCS TELNET	Display Telnet information Available for Telnet.		"TCPIPCS TELNET" on page 239
TCPIPCS TIMER	Display information about timers Available for Telnet.	TCPIPCS TIMESUM	"TCPIPCS TIMER" on page 240
TCPIPCS TRACE	Display TCP/IP CTrace information Available for Telnet.	TCPIPCS TCA	Table 2 on page 8
TCPIPCS TREE	Display information about data stored in Patricia trees Available for Telnet.	TCPIPCS TREESUM	"TCPIPCS TREE" on page 246
TCPIPCS TSDB	Format TSDB Available for Telnet.		"TCPIPCS TSDB" on page 249
TCPIPCS TSDX	Format TSDX Available for Telnet.		"TCPIPCS TSDX" on page 250
TCPIPCS TSEB	Format TSEB Available for Telnet.		"TCPIPCS TSEB" on page 251

Table 14. TCP/IP IPCS commands (continued)

Command	Description	Alias	See page
TCPIPCS TTLS	Display state information about AT-TLS connections and groups		"TCPIPCS TTLS" on page 253
TCPIPCS UDP	Display UDP control blocks	TCPIPCS MUCB TCPIPCS UCB TCPIPCS UDPSUM	"TCPIPCS UDP" on page 256
TCPIPCS VMCF	Display information about VMCF and IUCV users		"TCPIPCS VMCF" on page 258
TCPIPCS XCF	Display information about XCF links and dynamic VIPA		"TCPIPCS XCF" on page 260
ERRNO	Interpret error numbers Available for Telnet.		"ERRNO command" on page 262
ICMPHDR	Format an ICMP header		"ICMPHDR" on page 264
IPHDR	Format an IP header		"IPHDR" on page 266
RESOLVER	Format and summarize resolver control blocks		NA
SETPRINT	Set destination so the IPCS subcommand output is sent to a user ID or the printer Available for Telnet.		"SETPRINT" on page 272
SKMSG	Format a stream message Available for Telnet.		"SKMSG" on page 273
TCPHDR	Format a TCP header		"TCPHDR" on page 275
TOD	Convert a S/390 64-bit time-of-day timestamp to a readable date and time Available for Telnet.		"TOD" on page 276
UDPHDR	Format UDP header		"UDPHDR" on page 277

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.

```
                 ┌─────────────────────────────────┐
                 ▼                                 │
►►──TCPIPCS──────subcommand──┬──────────────────┬──┴──────────────────────────►
                             └─(──parameters──)─┘
```

```
                                            ┌─TITLE───┐
►─┬─────────────────────────────────────┬──┼─────────┼──────────────────────────────►◄
  └─TCP──(──┬─tcp_proc_name─┬──)────────┘  └─NOTITLE─┘
            └─tcp_index─────┘
```

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 the TN3270E Server is running in its own address space). When issuing commands for Telnet that ran in its own address space, 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 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.

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:

```
Selected TCP/IP is not V1R7
```

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* 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───────────────────────────────────────────────────────────►

►──┬────────────────────────────────────────────────────────────────────┬──►
   │                                  ┌─MACRO──┐  ┌─SUMMARY─┐          │
   └─(──┬─*─────────────────────┬──┼────────┼──┼─────────┼──)─┘
        ├─variable_item─────────┤  ├─PASCAL─┤  └─DETAIL──┘
        │   ┌──────────────┐    │  └─ALL────┘
        │   ▼              │    │
        └─(───variable_list───)─┘

                                        ┌─TITLE───┐
►──┬──────────────────────────────────┬──┼─────────┼──────────────────────►◄
   └─TCP──(──┬─tcp_proc_name─┬──)─┘  └─NOTITLE─┘
             └─tcp_index─────┘
```

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

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 177 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 TCPIPCS API subcommand

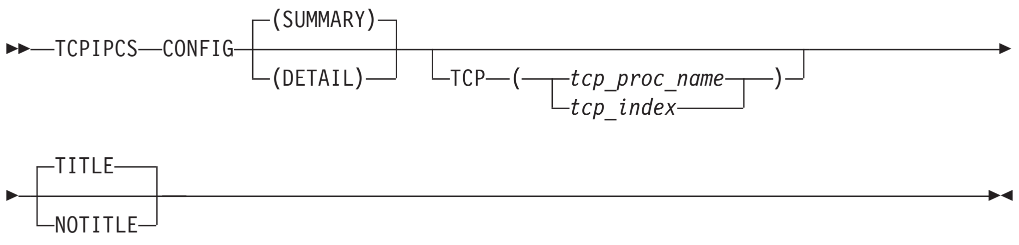
The following is sample output of the TCPIPCS API subcommand.

```
The contents of the SDST control blocks are formatted by the TCPIPCS 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 --
   ::
   +00B2  SDST_LOCAL_IPADDRLEN.  00
   +00B3  SDST_REMOTE_IPADDRLEN. 00
   +00B4  SDST_LOCAL_IPADDR.     00000000  00000000  00000000  00000000
   +00C4  SDST_REMOTE_IPADDR.    00000000  00000000  00000000  00000000
   ::
        -- End of array   --
```

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



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 177 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 Tsdb     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

Dif@      DeviceName       Next     Prev     DevR DevW Protocol
7F6AA408  LOOPBACK         7F1ED408 00000000 **** **** LOOPBACK
7F1ED408  OSAQDIO3         00000000 7F6AA408 **** **** MPCIPA

IPv4 Pif@ LinkName         Next     Prev     DeviceName       Protocol        Dif@     Lif@
7F679468  LOOPBACK         7F503B88 00000000 LOOPBACK         LOOPBACK        7F6AA408 7F6792E8
```

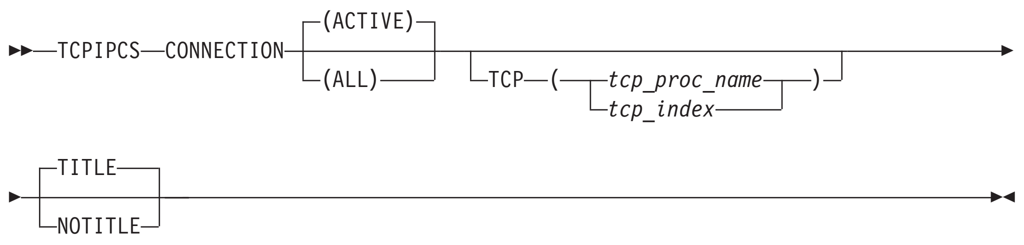
```
7F503B88  OSAQDIOL        00000000 7F679468 OSAQDIO3        IPAQENET        7F1ED408 7F1ED008

IPv6 Pif@ IntfName        Next     Prev     DeviceName      Protocol        Dif@     Lif@
7F503488  LOOPBACK6       7F503F08 00000000 LOOPBACK        LOOPBACK6       7F6AA408 7F3FDCE8
7F503F08  OSAQDI26        00000000 7F503488 OSAQDIO3        IPAQENET6       7F1ED408 7F6BF028

IPv4 Lif@ LinkName        Next     Prev     Pif@     IpAddr
7F1ED008  OSAQDIOL        7F6792E8 00000000 7F503B88 9.67.115.82
7F6792E8  LOOPBACK        00000000 7F1ED008 7F679468 127.0.0.1

IPv6 Lif@ IntfName        Next     Prev     Pif@     IpAddr
7F3083C8  OSAQDI26        7F3FDCE8 00000000 7F503F08 FEC9:C2D4:1::9:67:115:82
7F3FDCE8  LOOPBACK6       7F6BF028 7F3083C8 7F503488 ::1
7F6BF028  OSAQDI26        00000000 7F3FDCE8 7F503F08 FE80::2:559A:3F5F:1

Configuration Summary

IPv4 Lif@ LinkName        DeviceName      DevR DevW IpAddr
7F1ED008  OSAQDIOL        OSAQDIO3        **** **** 9.67.115.82
7F6792E8  LOOPBACK        LOOPBACK        **** **** 127.0.0.1

IPv6 Lif@ LinkName        DeviceName      DevR DevW IpAddr
7F3083C8  OSAQDI26        OSAQDIO3        **** **** FEC9:C2D4:1::9:67:115:82
7F3FDCE8  LOOPBACK6       LOOPBACK        **** **** ::1
7F6BF028  OSAQDI26        OSAQDIO3        **** **** FE80::2:559A:3F5F:1

Analysis of Tcp/Ip for TCPCS2 completed
```

TCPIPCS 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



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 177 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 10 total TCP connections, of which eight are active. There is one UDP connection, which is not active. Both raw connections are active.

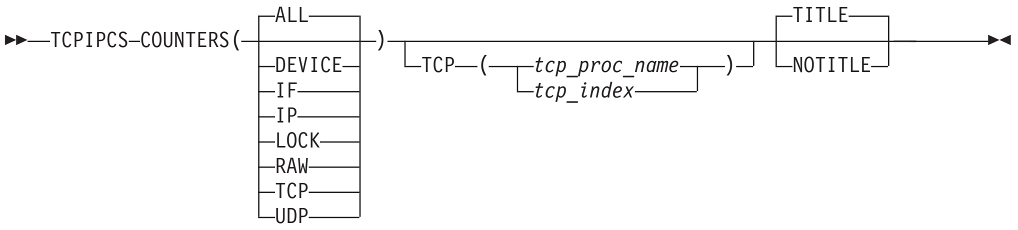
```
TCPIPCS CONNECTION
Dataset: IPCS.R8A0723.RASDUMP
Title:   EZRPE005
The address of the TSAB is: 098221F0
Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status
09822230  1 TCPCS     V1R5    08E85000 08E850C8 001E 9FFF7E7F  Active
098222B0  2 TCPCS2    V1R5    08937000 089370C8 01F6 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

TCP IPv4 Connections:
Userid   Conn       State
TCPCS    00000012   Listening   Local Socket  : 127.0.0.1..1024
                                Foreign Socket: 0.0.0.0..0
BPXOINIT 00000019   Listening   Local Socket  : 127.0.0.1..1024
                                Foreign Socket: 127.0.0.1..1025
TCPCS    00000016   Established Local Socket  : 127.0.0.1..1024
                                Foreign Socket: 127.0.0.1..1025
TCPCS    00000014   Established Local Socket  : 127.0.0.1..1024
                                Foreign Socket: 127.0.0.1..1025
4 TCP IPv4 connections

Active TCP IPv6 Connections:
Userid   Conn     State        Socket
FTPUNIX1 00000051 Listening    Local    ::0..21
                               Foreign  ::0..0
FTPMVS1  00000049 Listening    Local    ::0..1821
                               Foreign  ::0..0
2 TCP IPv6 connections

Active UDP Unicast IPv4  Connections:
Userid   Conn     Socket
PORTMAP  00000027 Local    0.0.0.0..111
                  Foreign  0.0.0.0..0
OSNMPD   00000030 Local    0.0.0.0..161
                  Foreign  127.0.0.1..162
MISCSRV  00000039 Local    198.11.98.124..7
                  Foreign  0.0.0.0..0
MISCSRV  0000003E Local    198.11.98.124..9
                  Foreign  0.0.0.0..0
4 UDP Unicast IPv4 connections

Active UDP Unicast IPv6  Connections:
Userid   Conn     Socket

0 UDP Unicast IPv6 connections

Active UDP Multicast IPv4  Connections:
Userid   Conn     Socket

0 UDP Multicast IPv4 connections

Active UDP Multicast IPv6  Connections:
Userid   Conn     Socket
```

```
0 UDP Multicast IPv6 connections

Active RAW Connections:
Userid   Conn     Protocol    Socket
TCPCS    00000006 IP          Local    0.0.0.0
                              Foreign  0.0.0.0
TCPCS    00000008 RAW         Local    0.0.0.0
                              Foreign  0.0.0.0
TCPCS    0000000E IP          Local    ::0
                              Foreign  ::0
3 RAW connections
Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS COUNTERS

Use this subcommand to display information about TCP/IP internal execution statistics.

Syntax



Parameters

ALL
Display all statistics.

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 177 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

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts          Status

15136040  1 TCPCS     V1R7    1511E000 1511E0C8 002F 9FFF767F 00000000  Active

   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

  nonbat.............. 3
  batch............... 0
  batnum.............. 0
  nonrsm.............. 0
  batrsm.............. 0
  rsmnum.............. 0
  rteadd.............. 28
  rtedel.............. 0
  rteinc.............. 11
  rtedec.............. 8
  rtpadd.............. 14
  rtpdel.............. 0
  rtechg.............. 86
  trredr.............. 0
  trsusp.............. 0
  trsust.............. 0
  dupfrg.............. 0
  dataadj1............ 0
  dataadj2............ 0

IP6 Statistics

  nonbat.............. 32
  batch............... 0
  batnum.............. 0
  nonrsm.............. 0
  batrsm.............. 0
  rsmnum.............. 0
  rteadd.............. 0
  rtedel.............. 0
  rteinc.............. 0
  rtedec.............. 0
  rtpadd.............. 11
  rtpdel.............. 0
  rtechg.............. 43
  trredr.............. 0
  trsusp.............. 0
  trsust.............. 0
  dupfrg.............. 0
  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.

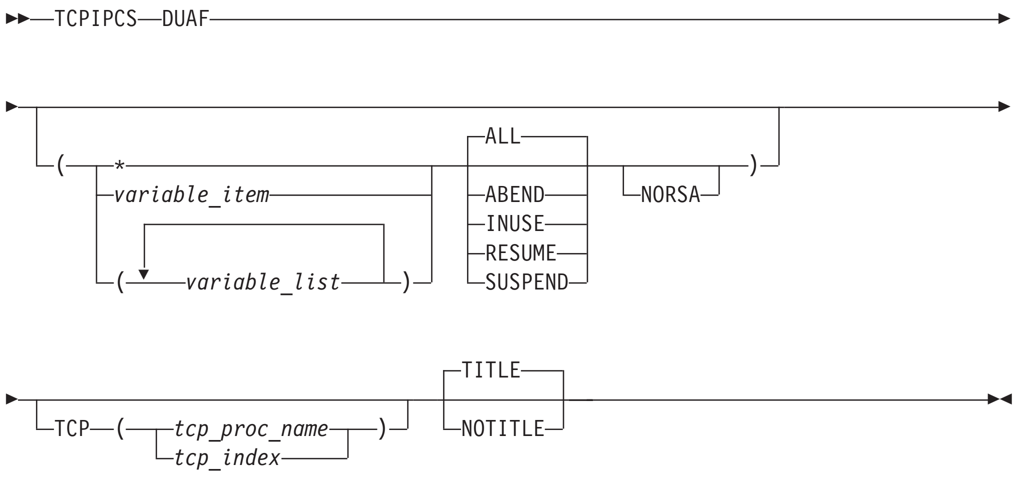
The status of each DUCB is abbreviated as follows:

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



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. 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.

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 177 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 S0C4/00000004 TCB P=0029,S=000E,H=0019

The address of the TSAB is: 08D138C0

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

08D13900  1 TCPCS     V2R10   0885A000 0885A0C8 0029 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 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 EZBIEOER
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 09BB9798  Next 087846B8  Mod EZBPFSOC
RSA  087846C0  Prev 087843F8  Next 08784988  Mod EZBPFOPN
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

```
►►──TCPIPCS──DUCB──┬──────────────────────────────────┬──────────────────────────────►
                   └─(──┬─*──────────────────────┬──)─┘
                        ├─variable_item──────────┤
                        │  ┌──────────────────┐  │
                        │  ▼                  │  │
                        └─────variable_list───┴──┘

                                               ┌─TITLE───┐
►──┬──────────────────────────────────────┬────┼─────────┼──────────────────────────►◄
   └─TCP──(──┬─tcp_proc_name─┬──)─────────┘    └─NOTITLE─┘
             └─tcp_index─────┘
```

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 177 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:

```
TCPIPCS DUCB
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
DUCB Detail Analysis
Dispatchable Unit Allocation Table: 08E83010
+0000  DUAT0      EYE...... DUAT      NEXT..... 00000000
+0018  DUAE0      DUCB..... 08D8D010  FLAGS.... 0014001E
+0020  DUAE1      DUCB..... 08D90000  FLAGS.... 0034001E
+0028  DUAE2      DUCB..... 08D93000  FLAGS.... 0050001E
+0030  DUAE3      DUCB..... 08D96000  FLAGS.... 0014001E
+0038  DUAE4      DUCB..... 08D99000  FLAGS.... 5490001E
.
.
.
+0280  DUAE77     DUCB..... 08E74000  FLAGS.... 00000000
+0288  DUAE78     DUCB..... 08E77000  FLAGS.... 00000000
+0290  DUAE79     DUCB..... 08E7A000  FLAGS.... 00000000
+0298  DUAE80     DUCB..... 08E7D000  FLAGS.... 00000000
+02A0  DUAE81     DUCB..... 08E80000  FLAGS.... 00000000
Dispatchable Unit Control Block: DUCB0
 EZBDUCB: 08D8D010
 +0000  DUCB_EYE................. DUCB
 +0004  DUCB_LENGTH.............. 0100
 +0006  DUCB_VERSION............. 0002
 +0008  DUCB_TOKEN............... 08D8D010  0014001E  10000000  00000000
 +0018  DUCB_DUSA................ 08D8D110
 +001C  DUCB_AVAIL_CHAIN......... 00000000
 +0020  DUCB_DUAEP............... 08E83028
 +0026  DUCB_ASID................ 001E
 +0028  DUCB_ASCB................ 00FA4400
 +002C  DUCB_ATCB................ 007EC920
 +0030  DUCB_ITCVT............... 08E853C8
 +0034  DUCB_LOCKSHELDCOUNT...... 00000000
 +0038  DUCB_LOCKS_TABLE......... 08D8D194
 +003C  DUCB_LOCKS_SUSPENDED..... 00000000
 +0040  DUCB_LOCKS_SUSPENDED_NEXT. 7FFAFAF1
 +0044  DUCB_SUSPENDTOKEN........ 00000000  40000000
 +004C  DUCB_JOBNAME............. TCPCS
 +0054  DUCB_TSAB................ 098221F0
 +0058  DUCB_TSEB................ 09822230
 +005C  DUCB_TSDB................ 08E85000
 +0060  DUCB_TSDX................ 08E850C8
 +0064  DUCB_TCP_ASCB............ 00FA4400
 +0068  DUCB_STREAMHEAD.......... 00000000
 +006C  DUCB_OSI................. 00000000
 +0070  DUCB_CREATE_FLAGS........ 00000000
 +0074  DUCB_CID................. 00000000
 +0078  DUCB_PORT................ 0000
 +007A  DUCB_IPADDR_LEN.......... 00
 +007C  DUCB_IPADDR.............. 00010002  00030004  00050006  00070008
 +008C  DUCB_TRR_PTR............. 08D8D128
 +0090  DUCB_ESTAEX_TOKEN........ 00000004
 +0094  DUCB_ERRORCODE........... 00000000
 +0098  DUCB_REASONCODE.......... 00000000
 +009C  DUCB_ABEND_FLAGS......... 000F0000
 +00A0  DUCB_DUMP_ECB............ 00000000
        DUCB_EXP_SAVE............
 +00B8  08D8D110  00000000  00000000  88E8865C  08F671E8  7F4FC488  08D8F75C  08D8D010  08D90118  08D8F3A0  000001A8  08D8D010
 +00E8  08D8F214  08E850C8  08D8FDA8  08D8FA30  0923E21C  88E8862C
Dispatchable Unit Stack Area: DUSA0@1
 EZBDUSA: 08D8D110
 +0000  DUSA_EYE....... DUSA
 +0004  DUSA_NEXTDUSA.. 08CFF010
 +0008  DUSA_DUCB...... 08D8D010
 +000C  DUSA_START..... 08D8D450
 +0010  DUSA_LAST...... 08D90000
 +0014  DUSA_NEXTAVAIL. 08D8D540
```

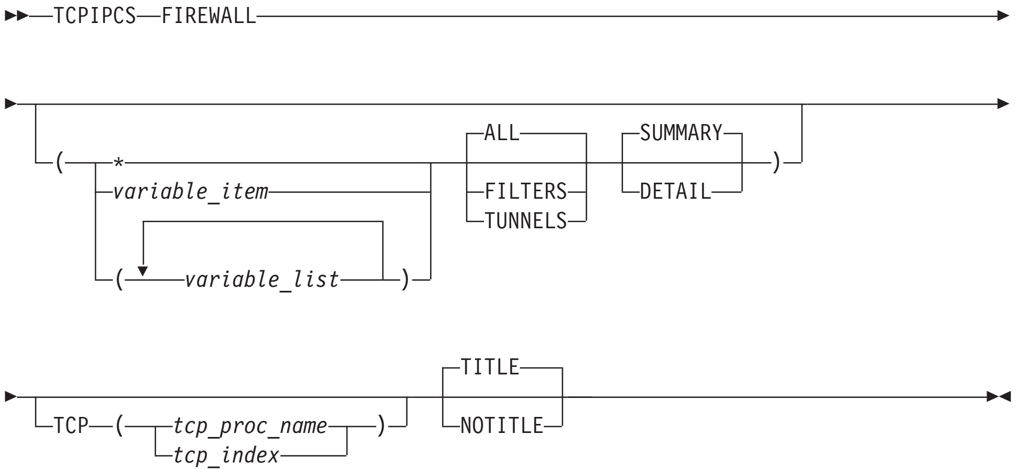
```
Register Save Area: RSA0@1
Module: EZBTIWAT
 EZBRSA: 08D8D458
 +0000  RSA_DUSA. 08D8D110  RSA_PREV. 0A301890  RSA_NEXT. 00000000  RSA_R14.. FEFEFEFE  RSA_R15.. FEFEFEFE  RSA_R0... FEFEFEFE
 +0018  RSA_R1... FEFEFEFE  RSA_R2... FEFEFEFE  RSA_R3... FEFEFEFE  RSA_R4... FEFEFEFE  RSA_R5... FEFEFEFE  RSA_R6... FEFEFEFE
 +0030  RSA_R7... FEFEFEFE  RSA_R8... FEFEFEFE  RSA_R9... FEFEFEFE  RSA_R10.. FEFEFEFE  RSA_R11.. FEFEFEFE  RSA_R12.. FEFEFEFE
 +0050  RSA_AR13. FEFEFEFE  RSA_AR14. FEFEFEFE  RSA_AR15. FEFEFEFE  RSA_AR0.. FEFEFEFE  RSA_AR1.. FEFEFEFE  RSA_AR2.. FEFEFEFE
 +0068  RSA_AR3.. FEFEFEFE  RSA_AR4.. FEFEFEFE  RSA_AR5.. FEFEFEFE  RSA_AR6.. 08D773D0  RSA_AR7.. 7EFEFEFE  RSA_AR8.. 08E85084
 +0080  RSA_AR9.. 08E85238  RSA_AR10. 08E8523C  RSA_AR11. 08E85350  RSA_AR12. 08E85358
Dynamic Area of RSA0@1
Module: EZBTIWAT
 08D8D458  08D8D110  0A301890  00000000  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  | .QJ............................. |
    +0020  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  | ................................ |
    +0040  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  | ................................ |
    +0060  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  08D773D0  7EFEFEFE  08E85084  | .....................P.}=....Y&d |
.
.
.
    +0FC0  FEFEFEFE  FEFEFEFE  08D8E620  08D8E620  FEFEFEFE  FEFEFEFE  FEFEFEFE  7F207498  | .........QW..QW.............."..q |
    +0FE0  09822230  FEFEFEFE  08E850C8  08D8D010  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  | .b.......Y&H.Q}................. |
Dispatchable Unit Stack Area: DUSA0@2
 EZBDUSA: 08CFF010
 +0000  DUSA_EYE....... DUSA
 +0004  DUSA_NEXTDUSA.. 00000000
 +0008  DUSA_DUCB...... 08D8D010
 +000C  DUSA_START..... 08CFF028
 +0010  DUSA_LAST...... 08D04000
 +0014  DUSA_NEXTAVAIL. 08CFF028
Register Save Area: RSA0@2
Module: EZBCTRCD
 EZBRSA: 08CFF030
 +0000  RSA_DUSA. 08CFF010  RSA_PREV. 08D8F3A0  RSA_NEXT. 08CFF1D0  RSA_R14.. FEFEFEFE  RSA_R15.. FEFEFEFE  RSA_R0... FEFEFEFE
 +0018  RSA_R1... FEFEFEFE  RSA_R2... FEFEFEFE  RSA_R3... FEFEFEFE  RSA_R4... FEFEFEFE  RSA_R5... FEFEFEFE  RSA_R6... FEFEFEFE
 +0030  RSA_R7... FEFEFEFE  RSA_R8... FEFEFEFE  RSA_R9... FEFEFEFE  RSA_R10.. FEFEFEFE  RSA_R11.. FEFEFEFE  RSA_R12.. FEFEFEFE
 +0050  RSA_AR13. 00000000  RSA_AR14. FEFEFEFE  RSA_AR15. 08D8FDD4  RSA_AR0.. 08D8FDA8  RSA_AR1.. 08D8D010  RSA_AR2.. FEFEFEFE
 +0068  RSA_AR3.. 000263D8  RSA_AR4.. 01FF000C  RSA_AR5.. 000003C4  RSA_AR6.. 00000048  RSA_AR7.. 00000004  RSA_AR8.. 00026795
 +0080  RSA_AR9.. 08D8FA30  RSA_AR10. 08D8F75C  RSA_AR11. FEFEFEFE  RSA_AR12. FEFEFEFE
Dynamic Area of RSA0@2
Module: EZBCTRCD
 08CFF030  08CFF010  08D8F3A0  08CFF1D0  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  | ..0..Q3...1}.................... |
    +0020  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  | ................................ |
    +0040  FEFEFEFE  FEFEFEFE  88E88744  88E88AB8  00000000  FEFEFEFE  08D8FDD4  08D8FDA8  | ........hYg.hY...........Q.M.Q.y |
    +0060  08D8D010  FEFEFEFE  000263D8  01FF000C  000003C4  00000048  00000004  00026795  | .Q}........Q.......D...........n |
    +0080  08D8FA30  08D8F75C  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  03C40010  3001012D  | .Q...Q7*.................D...... |
    +00A0  B64C8AC7  14F03740  001E001E  001E02F0  007EC920  892336FA  E3C3D7C3  E2404040  | .<.G.0. .......0.=I.i...TCPCS    |
    +00C0  00000000  0000020A  00000000  00000000  00000000  00000000  00000000  00000000  | ................................ |
    +00E0  00000000  00000002  00000000  00000000  00000000  00000000  00000000  00000000  | ................................ |
    +0100  00000000  00000000  00000000  00000000  00000000  00000000  012892A0  00000000  | ..........................k..... |
    +0120  04FEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  | ................................ |
    +0140  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  FEFEFEFE  | ................................ |
    +0160  00000000  08D8F75C  892336FA  08CFF190  08D8F3A0  08D8D010  08D8D128  08D8F214  | .....Q7*i.....1..Q3..Q}..QJ..Q2. |
    +0180  08CFF010  08D8FDA8  08CFF1D0  0923E21C  88E8862C  08CFF030  00000008  08CFF010  | ..0..Q.y..1}..S.hYf...0.......0. |
TCPIP Recovery Routine
 DTRR: 08D8D128
 +0000  TRR_CURRENT_INDEX. 00000000
 +0004  TRR_MAX_INDEX..... 00000005
         -- Array elements --
 +0008  TRR_ROUTINE....... 08E8875E
 +000C  TRR_PARM.......... 00000000
 +0010  TRR_DATA.......... 80000000
 +0014  TRR_REGS.......... 08D8D5B8
 +0018  TRR_DUMPPARM...... 00000000
 +001C  TRR_ROUTINE....... 08E8875E
 +0020  TRR_PARM.......... 00000000
 +0024  TRR_DATA.......... 80000000
 +0028  TRR_REGS.......... 08D8E358
 +002C  TRR_DUMPPARM...... 00000000
 +0030  TRR_ROUTINE....... 08E8875E
 +0034  TRR_PARM.......... 00000000
 +0038  TRR_DATA.......... 80000000
 +003C  TRR_REGS.......... 08CFF190
 +0040  TRR_DUMPPARM...... 00000000
 +0044  TRR_ROUTINE....... 00000000
 +0048  TRR_PARM.......... 00000000
 +004C  TRR_DATA.......... 00000000
 +0050  TRR_REGS.......... 00000000
 +0054  TRR_DUMPPARM...... 00000000
 +0058  TRR_ROUTINE....... 00000000
 +005C  TRR_PARM.......... 00000000
 +0060  TRR_DATA.......... 00000000
 +0064  TRR_REGS.......... 00000000
 +0068  TRR_DUMPPARM...... 00000000
         -- End of array   --
```

```
82 DUCBs were found
 2 DUCBs were formatted
Analysis of Tcp/Ip for TCPCS completed
```

TCPIPCS FIREWALL

Use this subcommand to display information about firewall filters or tunnels.

Syntax



Parameters

If no parameters are specified, all Firewall filters and tunnels are summarized.

* 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:

filter_address
Displays the Firewall filter 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.

tunnel_address
Displays the Firewall tunnel 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 described above, you can specify the following keyword parameters:

ALL
Display information for all Firewall filters and tunnels. ALL is the default.

FILTERS
Display only information for Firewall filters.

TUNNELS
Display only information for Firewall tunnels.

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 177 for a description of these parameters.

Restrictions: Be aware of the following keyword restrictions:

- If you specify multiple keywords from the set {ALL, FILTERS, TUNNELS}, only the last one is used.
- If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.
- | The TCPIPCS FIREWALL subcommand does not work on stacks configured for IPSECURITY. See the "TCPIPCS IPSEC" on page 203 for information about displaying IP security information on stacks configured for IPSECURITY.

Sample output of the TCPIPCS FIREWALL subcommand

The following is a sample output of the TCPIPCS FIREWALL subcommand:

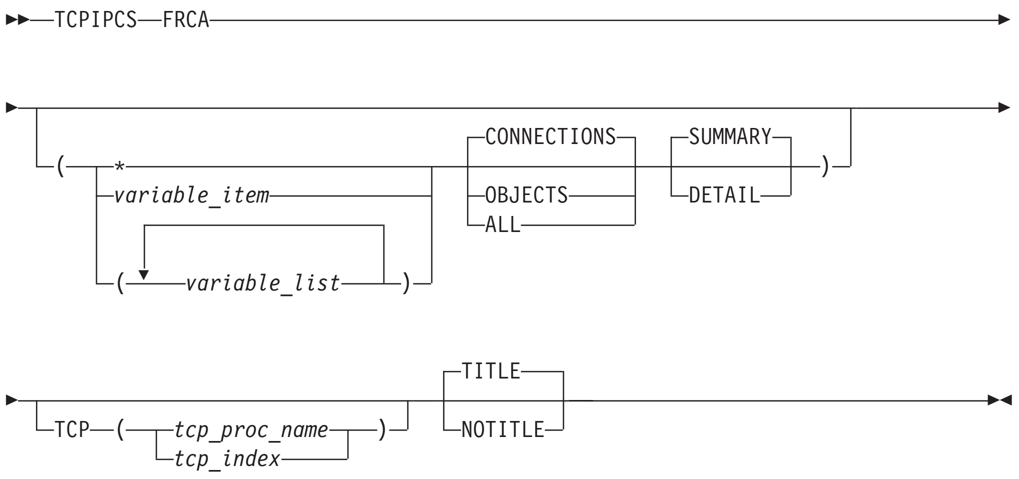
```
TCPIPCS FIREWALL ((* )  SUMMARY ALL )
Dataset: IPCS.A594094.DUMPN
Title:   FIREWALL DUMP


The address of the TSAB is: 08DE56F8

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

08DE5738  1 TCPCS     V2R10   0854B000 0854B0C8 01F6 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 TCPCS.  Index: 1

TCPIP Firewall Analysis

Secure Adapters:
  9.67.116.125
  162.33.33.33
  9.67.11.1
  9.67.11.2
  9.67.114.1

  Pre-decap filtering : No

Filter Summary:

Action  Src1@        Src2@           Dst1@         Dst2@                           SPort   DPort    Protocol
Permit  0.0.0.0      0.0.0.0         0.0.0.0       0.0.0.0                         =500    =500     17 (UDP)
Permit  0.0.0.0      0.0.0.0         0.0.0.0       0.0.0.0                         0       0        51 (SIPP-AH)
Permit  0.0.0.0      0.0.0.0         0.0.0.0       0.0.0.0                         0       0        50 (SIPP-ESP)
Permit  0.0.0.0      0.0.0.0         9.67.113.36   255.255.255.255                 0       =1014     6 (TCP)
Permit  9.67.113.36  255.255.255.255 0.0.0.0       0.0.0.0                         =1014   0         6 (TCP)
Unknown 9.67.116.36  255.255.255.255 9.67.116.47   255.255.255.255                 0       0         1 (ICMP)
Permit  9.67.116.36  255.255.255.255 9.67.116.47   255.255.255.255                 0       0         1 (ICMP)
...
Permit  9.67.116.47  255.255.255.255 9.67.113.4    255.255.255.255                 0       0       254 (254)
```

```
Deny    0.0.0.0      0.0.0.0        0.0.0.0       0.0.0.0                        0       0       254 (254)

Tunnel Summary:

Name       Src@            Dst@            Policy   Format
0000000510:0000000508:0000000507:0000000516:0000000517:0000000506:0000000003
           9.67.116.36     9.67.116.47     00000021 00000033
0000000504:0000000508:0000000507:0000000504:0000000503:0000000506:0000000004
           9.67.116.36     9.67.116.47     00000021 00000033
0000000510:0000000508:0000000507:0000000516:0000000518:0000000502:0000000005
           9.67.116.36     9.67.116.47     00000042 000000CC
0000000510:0000000508:0000000507:0000000516:0000000518:0000000502:0000000006
           9.67.116.36     9.67.116.47     00000042 000000CC

Analysis of Tcp/Ip for TCPCS completed
```

TCPIPCS FRCA

Use this subcommand to display information about the Fast Response Cache Accelerator (FRCA) connections or about cached objects.

Syntax



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:

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.

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 zero to avoid the address being 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. CONNECTIONS is the default.

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 177 for a description of these parameters.

Restrictions: Be aware of the following keyword restrictions:
- If you specify multiple keywords from the set {CONNECTIONS, 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

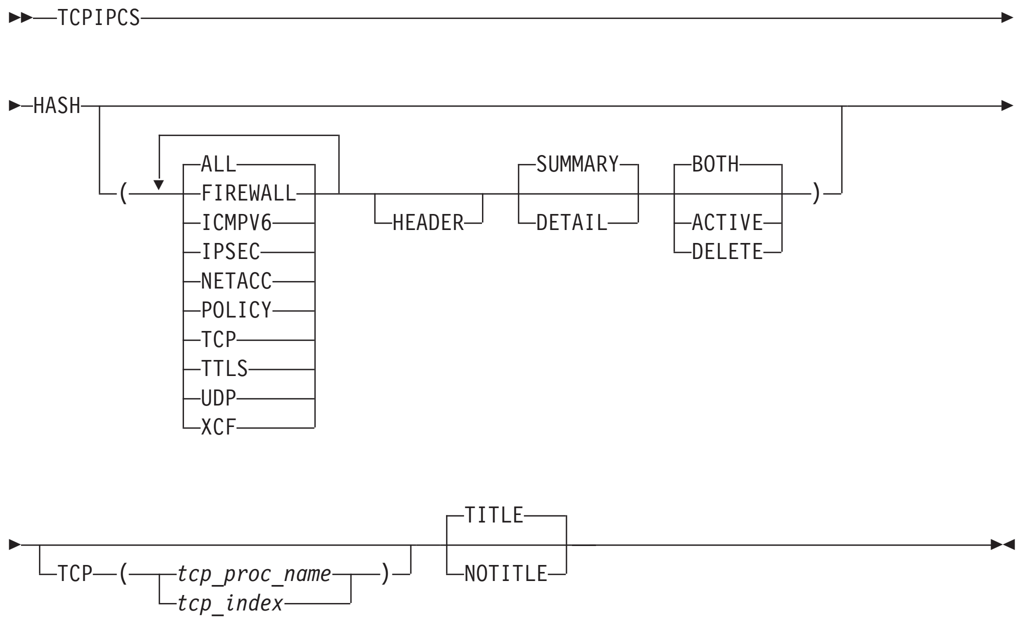
The following is sample output of the TCPIPCS FRCA subcommand:

```
TCPIPCS FRCA
Dataset: IPCS.MV20372.DUMPA
Title:   TCPSVT   V2R10: Job(TCPSVT  ) EZBITSTO(HTCP50A 99.281)+
         00077A S4C5/74BE2500 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

FRCA Server Connections

Uwsx@    Tcb@     Cache@      References Flags
12E6BA90 7F272D08 12E6AE98            10 60

FRCA Client Connections

Uwcx@    Tcb@     Server@  Object@  Flags
7F20E060 7F20DD08 12E6BA90 1299CB08 08
7F14D460 7F14D108 12E6BA90 1299BA88 48
7F1FAC60 7F1FA908 12E6BA90 12434488 48
7F4DD460 7F4DD108 12E6BA90 00000000 28
7F0A9060 7F0A8D08 12E6BA90 00000000 28
7F08F460 7F08F108 12E6BA90 00000000 28
7F066860 7F066508 12E6BA90 00000000 00

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS HASH

Use this subcommand to display information about the structure of TCP/IP hash tables.

Syntax



Parameters

ALL
: Display structure of all TCP/IP hash tables. ALL is the default.

FIREWALL
: Display only the structure of Firewall hash tables.

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 177 for a description of these parameters.

Restrictions: Be aware of the following keyword restrictions:

| • If you specify multiple keywords from the set
| (ALL,FIREWALL,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.

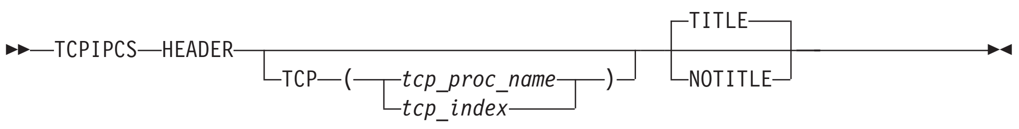
```
TCPIPCS HASH ( DETAIL ALL )
 Dataset: D74L.KWDEV03A.DUMP
 Title:   ICMP HASHTAB
 The address of the TSAB is: 0999D6F8
 Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status
 0999D738  1 TCPCS2    V1R5    08FE9000 08FE90C8 0013 00000000  Active
 0999D7B8  2 TCPCS     V1R5    08FB2000 08FB20C8 002D 00000000  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: 1
 TCPIP 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 7F2F8E80 Active  0AA6BFD8 FEC00000 00000000 00000000 00000000
                                            Clock Ticks...... 00000003
                                            Tokens........... 00
                                            Token Tenths..... 00
       5 7F699C58 7F2F9100 Active  0AA6BF98 00000000 00000000 00000000 00000000
                                            Clock Ticks...... 00000008
                                            Tokens........... 00
                                            Token Tenths..... 00
       6 7F699C68 7F2F9080 Active  0AA6BFA8 00000000 00000000 00000000 00000000
                                            Clock Ticks...... 00000011
                                            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
   Element queue      : 08FE9E48
 Bucket# Bucket@  Element@ Status  User@    KeyValue
       0 7F528B88 7F507FE0 Active  7F510108 00000000 00000000 00000000
     530 7F52ACA8 7F5080C0 Active  7F51B988 00000000 00000000 00150000
   30479 7F59FC78 7F508020 Active  7F510A08 7F000001 00000000 04000000
   35181 7F5B2258 7F5080A0 Active  7F51A788 00000000 00000000 27170000
   37771 7F5BC438 7F508040 Active  7F511308 7F000001 7F000001 04000401
   40773 7F5C7FD8 7F508060 Active  7F510E88 7F000001 7F000001 04010400
 6 elements in TCB V4 Index Table
 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
 Bucket# Bucket@  Element@ Status  User@    KeyValue
       0 7F2FDB88 7F2F8C80 Active  7F510588 00000000 00000000 00000000 00000000
                                            00000000 00000000 00000000 00000000
                                            00000000
     530 7F2FFCA8 7F2F8F00 Active  7F51B988 00000000 00000000 00000000 00000000
                                            00000000 00000000 00000000 00000000
                                            00000000
 2 elements in TCB V6 Index Table
 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
 Bucket# Bucket@  Element@ Status  User@    KeyValue
       0 7F403B88 7F508000 Active  7F4F8108 00000000 0000
     529 7F405C98 7F508080 Active  7F500608 00000000 0211

 2 elements in UDP DMUX V4 Table
 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 V6 Table
 Hash Table Header at 7F203B88
   Instance            : 6
   Active entries      : 1
   Hash buckets        : 62,533
   User free routine   : 88DB0E3C
   Element queue       : 08FE9E50
 Bucket# Bucket@  Element@ Status  User@    KeyValue
       0 7F203B88 7F2F8D00 Active  7F4F8808 00000000 00000000 00000000 00000000
                                            0000
 1 elements in UDP DMUX V6 Table
 Table Summary:
   Active buckets      : 1
   Inactive buckets    : 0
   Unused buckets      : 62,532
   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        : 1

 UDP MULTICAST V6 Table
 Hash Table Header at 7F10EB88
   Instance            : 7
   Active entries      : 0
   Hash buckets        : 62,533
   User free routine   : 88DB0E3C
   Element queue       : 08FE9E50
 0 elements in UDP MULTICAST V6 Table
 Table Summary:
   Active buckets      : 0
   Inactive buckets    : 0
   Unused buckets      : 62,533
   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
 Analysis of Tcp/Ip for TCPCS2 completed
```

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



Parameters

TCP, TITLE, NOTITLE
See “Parameters” on page 177 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

 Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

 13391C00  1 TCPSVT    V2R10   1323B000 1323B0C8 07DE 04041405  Active
 13391C80  2 TCPSVT2   V2R10   00000000 00000000 07E8 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

                                   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 FBD700, JOB(WLM), for the home ASID
  ASXB11 at 7FDFA0 and TCB11M at 7FB440 for the home ASID
  HOME ASID: 000B PRIMARY ASID: 000B SECONDARY ASID: 000B

  GPR VALUES
      0-3  00000001  0288E01C  00000C38  00000008
      4-7  007FB440  007FFC10  007F6A68  00FBD700
      8-11 00000000  01584B00  015AD820  007FEE48
     12-15 00000000  00000000  80FDE336  81584B18
...
```

TCPIPCS HELP

Use this subcommand to display TCPIPCS usage and syntax information.

Syntax

```
►►──TCPIPCS──HELP──┬───────────────────────────────────────────┬──────────►◄
                   │                      ┌─────────────┐      │
                   │                      ▼ ┌─ALL──────┐ │      │
                   └─(──┬─*─────────────┬───┼──────────┼─┴──)───┘
                        └─variable_item─┘   ├─FUNCTION─┤
                                            ├─OPERANDS─┤
                                            └─SYNTAX───┘
```

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:

```
tcpipcs 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

   Syntax:

     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.

Syntax

```
►►──TCPIPCS──IPSEC──────────────────────────────────────────────────────────►

►─┬─────────────────────────────────────────────────────────────────────┬───►
  │                                 ┌─ALL─────┐  ┌─SUMMARY─┐           │
  └─(──┬─*─────────────────────────┬┼─────────┼──┼─────────┼──)────────┘
       ├─variable_item─────────────┤├─FILTERS─┤  └─DETAIL──┘
       │    ┌──────────────────┐   │├─TUNNELS─┤
       │    ▼                  │   │└─XLPORTS─┘
       └─(──────variable_list──┴─)─┘

                                        ┌─TITLE───┐
►─┬───────────────────────────────────┬─┼─────────┼──────────────────────────►◄
  └─TCP──(──┬─tcp_proc_name─┬──)──────┘ └─NOTITLE─┘
            └─tcp_index─────┘
```

Parameters

If no parameters are specified, all IP security filters, tunnels, and NAT translated ports 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
The following variable parameters can be repeated up to 32 times, separated by a blank space, within parentheses:

filter_address
Displays the IP security 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 zero to avoid the address being 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 zero to avoid the address being 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, and NAT Traversal remote port translations. 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.

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 177 for a description of these parameters.

Tips:
- If you specify multiple keywords from the set {ALL, FILTERS, TUNNELS}, only the last one is used.
- If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

Restriction: To display FIREWALL information, see "TCPIPCS FIREWALL" on page 192. 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:

```
TCPIPCS IPSEC((*) SUMMARY ALL)
Dataset: IPCS.D133971.DUMPA
Title:   SLIP DUMP ID=AIKE


The address of the TSAB is: 3B8E6000

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

3B8E6040  1 TCPSVT    V1R7    3B86C000 3B86C0C8 006C 97BF141F  Active

   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 TCPSVT.  Index: 1

TCPIP Ipsecurity Analysis

FWE_GDA   at 7F3B9D10
FILTER_DA at 7F3B7C10
TUNNEL_DA at 7F3B7690
ENCRYP_DA at 7F3B9610

  Filter set active   : Policy
  Filter logging      : No
  Pre-decap filtering : Yes

Filter Summary:
Filter@  Action  Src@1       Src@2       Dst@1         Dst@2     SPrt1 SPrt2 DPrt1 DPrt2 Protocol
7E84B870 Permit  0.0.0.0     0.0.0.0     0.0.0.0       0.0.0.0     500     0   500     0  17 (UDP)
7EB33310 Permit  0.0.0.0     0.0.0.0     0.0.0.0       0.0.0.0     500     0   500     0  17 (UDP)
7EB33090 Permit  0.0.0.0     0.0.0.0     0.0.0.0       0.0.0.0       0     0     0     0  50 (SIPP-ESP)
7EB34A10 Permit  0.0.0.0     0.0.0.0     0.0.0.0       0.0.0.0       0     0     0     0  50 (SIPP-ESP)
7EB34790 Permit  0.0.0.0     0.0.0.0     0.0.0.0       0.0.0.0       0     0     0     0  51 (SIPP-AH)
7EB34510 Permit  0.0.0.0     0.0.0.0     0.0.0.0       0.0.0.0       0     0     0     0  51 (SIPP-AH)
...
7EB5B510 Permit  0.0.0.0     0.0.0.0     0.0.0.0       0.0.0.0       0     0     0     0   0 (Any)
7EB5B290 Permit  0.0.0.0     0.0.0.0     0.0.0.0       0.0.0.0       0     0     0     0   0 (Any)
7EB5CA10 Deny    0.0.0.0     0.0.0.0     0.0.0.0       0.0.0.0       0     0     0     0   0 (Any)
7EB5C790 Deny    0.0.0.0     0.0.0.0     0.0.0.0       0.0.0.0       0     0     0     0   0 (Any)

Tunnel Summary:

Tunnel@  Src@            Dst@            Policy   Format   Name
7EB5F090 197.11.107.1    197.11.235.24   0000014A 000000CC Y 240 AH240m
7EC15250 197.11.107.1    197.11.235.109  000001EF 000000FF Y 153 AHESP10080m
7EC36410 197.11.107.1    197.11.235.109  000001EF 000000FF Y 153 AHESP10080m
7EC16410 197.11.107.1    197.11.235.109  000001EF 000000FF Y 152 AHESP10080m
...
7EB60090 197.11.107.1    197.11.235.25   000000A5 00000033 Y 242 ESP480m
7EB5D090 197.11.107.1    197.11.235.23   0000014A 000000CC Y 241 AH240m

Analysis of Tcp/Ip for TCPSVT completed
```

Figure 24. 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

```
                   ┌─(SUMMARY)─┐                                           ┌─TITLE───┐
►►──TCPIPCS──LOCK──┼───────────┼──┬─────────────────────────────────┬──────┼─────────┼──►◄
                   └─(DETAIL)──┘  └─TCP──(──┬─tcp_proc_name─┬──)──┘      └─NOTITLE─┘
                                            └─tcp_index─────┘
```

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 177 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   Ducb@    Lktb@    Susp@    Next@    DucbIx   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
            MCGRP
   ...
   0C    06 SKITSSL
            TCFG_CLEANUP

Analysis of Tcp/Ip for TCPSVT completed
```

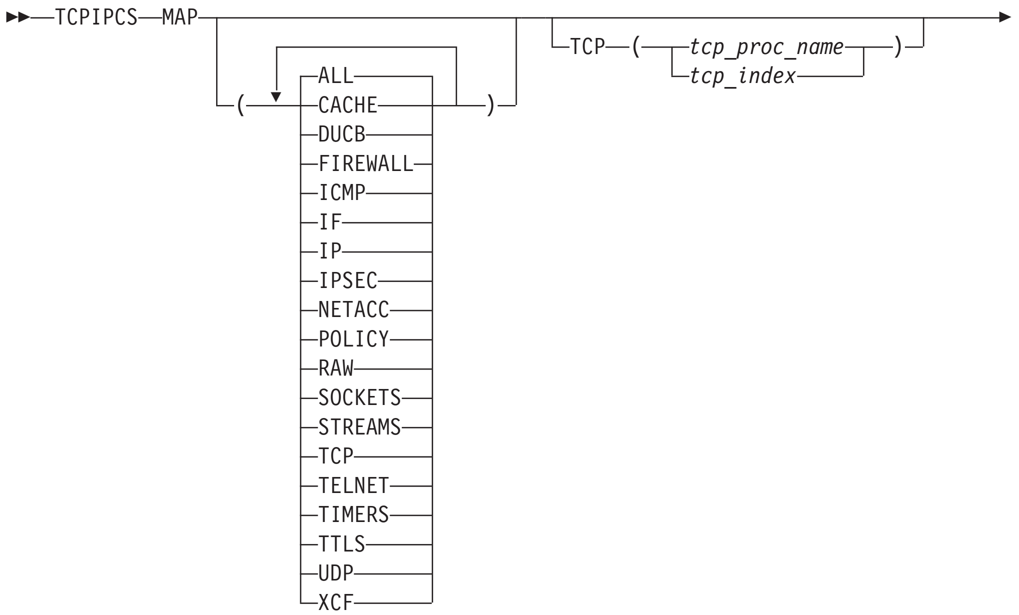
TCPIPCS MAP

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



```
   ┌─TITLE───┐
►──┤         ├──────────────────────────────────────────────────►◄
   └─NOTITLE─┘
```

Parameters

ALL
Display storage usage information for all components.

CACHE
Display only CACHE storage usage information.

DUCB
Display only DUCB storage usage information.

FIREWALL
Display only FIREWALL/IPSEC 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/FIREWALL 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 177 for a description of these parameters.

| **Restriction:** If you specify multiple keywords from the set (ALL, CACHE, DUCB,
| FIREWALL, 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+008C
0876B000 "ALCCSA"            96 6      08DD9328+0004
                                       08DD9368+0000
0876B388 "CACSMM"           120 6      0876B408+0004
0876B408 "CACSMM"           120 6      0876E5C8+0560
0876B488 "CACSMM"           120 6      0876B688+0004
0876B500 "CACSA "           120 6      0876B600+000C
0876B580 "CACSA "           120 6      0876B500+000C
0876B600 "CACSA "           120 6      0876E5C8+0218
0876B688 "CACSMM"           120 6      0876E5C8+0568
0876B700 "CACSA "           120 6      0876D700+000C
0876B780 "CACSA "           120 6      0876B700+000C
...
7F6E8B78 SKQU                64 6      7F6E8748+00E8
7F6E8BB8 SKQU                64 6      7F6E8AA8+0004
7F6E8BF8 SKQP                16 6      7F6E8B78+0018
                                       7F6E8BB8+0018
7F6E8C08 SKBD                32 6      7F6E8B78+002C
7F6E8C28 SKBD                32 6      7F6E8C08+0004
7F6E8C48 SKBD                32 6      7F6E8BB8+002C
7F6E8C68 SKBD                32 6      7F6E8C48+0004
7F6E8CC8 SKSC               176 6      7F6E8008+0004
                                       7F6E8748+0060
7F6E8D88 SKRT               128 6      0876E0C8+0130

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

```
►►──TCPIPCS──MTABLE──┬──────────────────────────────────────────────────────────┬──────────────►
                     │                                    ┌─BOTH───┐            │
                     └─(──┬─*─────────────────────────┬───┼─BYADDR─┼──)─────────┘
                          ├─variable_item─────────────┤   └─BYNAME─┘
                          │   ┌───────────────────┐   │
                          │   ▼                   │   │
                          └───(──variable_list──)─┴───┘

                                           ┌─TITLE───┐
►──┬────────────────────────────────────┬──┴─NOTITLE─┴─────────────────────────────────────►◄
   └─TCP──(──┬─tcp_proc_name─┬──)───────┘
             └─tcp_index─────┘
```

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.

BOTH
: Display modules sorted by address and by name.

BYADDR
: Display only modules sorted by address.

BYNAME
: Display only modules sorted by name.

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.

TCP, TITLE, NOTITLE
: See "Parameters" on page 177 for a description of these parameters.

Sample output of the TCPIPCS MTABLE subcommand

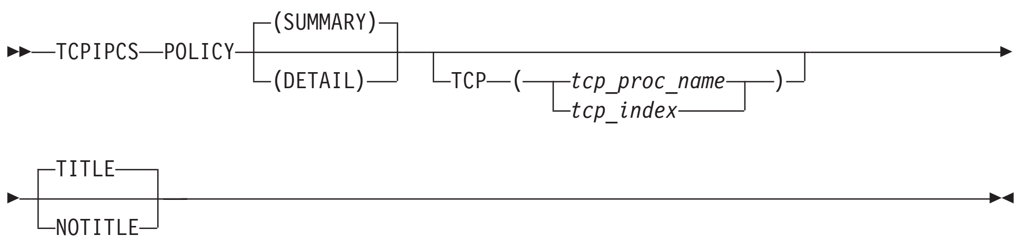
The following is a sample output of the TCPIPCS MTABLE subcommand:

```
TCPIPCS MTABLE (12DE3800 12D9B858)
Dataset: IPCS.A594094.DUMPM
Title:   TCPSVT   V2R10: Job(TCPSVT  ) EZBITSTO(HTCP50A 99.281)+
         00077A S4C5/74BE2500 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 Module Table Analysis

TCMT 12B590E8 EZBITCOM Size: 00D8 Cnt: 47
MTBL 12C23F28 EZBTIINI Size: 0CD4 Cnt: 272
MTBL 948ACA50 EZBTZMST Size: 0134 Cnt: 24
MTBL 94FE8470 EZBTTMST Size: 0704 Cnt: 148
MTBL 94AA0B00 EZBTMCTL Size: 0380 Cnt: 73

Module   Epa      Date       Time     PTF      Lmod     Asid

EZBIFARP 12DE35D8 1999/10/15 07:01:58 HTCP50A  EZBTIINI 0051
EZBXFINI 12D9B808 1999/10/08 00:37:29 HTCP50A  EZBTIINI 0051

Address 12DE3800 is EZBIFARP+0228
Address 12D9B858 is EZBXFINI+0050

Analysis of Tcp/Ip for TCPSVT completed
```

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 177 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: 12EFD818

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

12EFD858  1 TCPSVT    V2R10   12EAB000 12EAB0C8 0058 9CFF755F  Active
12EFD8D8  2 TCPSVT1   V2R10   12A0F000 12A0F0C8 0069 9CFF755F  Active
12EFD958  3 TCPSVT2   V2R10   127C9000 127C90C8 07DE 9CFF755F  Active
12EFD9D8  4 TCPSVT3   V2R10   126FB000 126FB0C8 0054 9CFF755F  Active
12EFDA58  5 TCPSVT4   V2R10   12646000 126460C8 004C 9CFF755F  Active
12EFDAD8  6 TCPSVT5   V2R10   1260E000 1260E0C8 07DD 9CFF755F  Active
12EFDB58  7 TCPSVT6   V2R10   12383000 123830C8 007A 9CFF755F  Active
12EFDBD8  8 TCPSVT7   V2R10   11ECE000 11ECE0C8 07DC 9CFF755F  Active

   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    paPRD-GenImp5
129454F0 Both  00  00  Allowed    padefault
129453F0 Both  E0  00  Allowed    paOSPF-1
129452F0 Both  E0  00  Allowed    paTST-1-GenImp1
129451F0 Both  C0  00  Allowed    paTST-1-GenImp2
12942B10 Both  A0  00  Allowed    paTST-1-GenImp3
12942A10 Both  80  00  Allowed    paTST-1-GenImp4
12942910 Both  60  00  Allowed    paTST-1-GenImp5
12942810 Both  40  00  Allowed    paTST-1-GenImp6
12942710 Both  20  00  Allowed    paTST-1-GenImp7
...

Policy Rules:

Prentry@ Permission Cond Level@   Cond@    Name
126C04F0 Allowed    DNF  00000000 00000000 prPRD-CBS-30001
128F2A90 Allowed    CNF  126EB590 00000000 prTST-WEB-4-80-BO
                         126C0B10 00000000
                         126C0790 126C0950
...
```

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──┬───────────────────────┬──┬──────────────────────────────────┬──►
                      │       ┌─ALL────┐      │  └─TCP──(──┬─tcp_proc_name──┬──)─┘
                      └─(─────┼─TCPIP──┼──)───┘            └─tcp_index──────┘
                              └─TELNET─┘

   ┌─TITLE───┐
►──┼─────────┼───────────────────────────────────────────────────────────────────►◄
   └─NOTITLE─┘
```

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 177 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.R450697.V6TCBD1
Title:   TCPCS2 CLIENT SIDE


The address of the TSAB is: 09DBE1A0

Tseb     SI Procedure Version Tsdb     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

;
; Profile generated on 2002/01/24 at 19:40:20
;
; Dump Dataset  : IPCS.R450697.V6TCBD1
; Dump Time     : 2001/11/07 17:41:23.649345
; TCP/IP Jobname: TCPCS2
;

;
```

```
; For informational purposes, only BEGINRoutes will
; be generated in this reconstructed profile.
;
; Either GATEWAY or BEGINRoutes statements may be
; specified in a real profile/obeyfile dataset.
; BEGINRoutes statement is the recommended way
; to define static routes.
;

ARPAGE 20

BEGINRoutes
 ROUTE 9.67.115.83 HOST = OSAQDIOL MTU 8992 MAXImumretransmittime 120
       MINImumretransmittime 0.5 ROUNDTRIPGain 0.125 VARIANCEGain 0.25
       VARIANCEMultiplier 2 DELAYAcks NOREPLaceable
 ROUTE 9.67.115.79 HOST = OSAQDIOL MTU 1500 MAXImumretransmittime 120
       MINImumretransmittime 0.5 ROUNDTRIPGain 0.125 VARIANCEGain 0.25
       VARIANCEMultiplier 2 DELAYAcks NOREPLaceable
 ROUTE 9.67.115.0 255.255.255.192 = OSAQDIOL MTU 8992
       MAXImumretransmittime 120 MINImumretransmittime 0.5
       ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
       DELAYAcks NOREPLaceable
 ROUTE DEFAULT 9.67.115.1 OSAQDIOL MTU 8992 MAXImumretransmittime 120
       MINImumretransmittime 0.5 ROUNDTRIPGain 0.125 VARIANCEGain 0.25
       VARIANCEMultiplier 2 DELAYAcks NOREPLaceable
 ROUTE FEC9:C2D4:1::9:67:115:83/128 FEC9:C2D4:1::206:2AFF:FE66:C81C
       OSAQDI26 MTU 8992 MAXImumretransmittime 120
       MINImumretransmittime 0.5 ROUNDTRIPGain 0.125 VARIANCEGain 0.25
       VARIANCEMultiplier 2 DELAYAcks NOREPLaceable
 ROUTE FEC9:C2D4:1::9:67:115:79/128 = OSAQDI26 MTU 1500
       MAXImumretransmittime 120 MINImumretransmittime 0.5
       ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
       DELAYAcks NOREPLaceable
 ROUTE FEC9:C2D4:1::206:2AFF:FE66:C81C/128 = OSAQDI26 MTU 8992
       MAXImumretransmittime 120 MINImumretransmittime 0.5
       ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
       DELAYAcks NOREPLaceable
 ROUTE 50C9:C2D4:1::206:2AFF:FE66:C81C/128 = OSAQDI26 MTU 8992
       MAXImumretransmittime 120 MINImumretransmittime 0.5
       ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
       DELAYAcks NOREPLaceable
 ROUTE 50C9:C2D4:1::0/112 50C9:C2D4:1::206:2AFF:FE66:C81C OSAQDI26 MTU
       8992 MAXImumretransmittime 120 MINImumretransmittime 0.5
       ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
       DELAYAcks NOREPLaceable
 ROUTE FEC9:C2D4:1::0/112 FEC9:C2D4:1::206:2AFF:FE66:C81C OSAQDI26 MTU
       8992 MAXImumretransmittime 120 MINImumretransmittime 0.5
       ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
       DELAYAcks NOREPLaceable
 ROUTE DEFAULT FEC9:C2D4:1::206:2AFF:FE66:C81C OSAQDI26 MTU 8992
       MAXImumretransmittime 120 MINImumretransmittime 0.5
       ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
       DELAYAcks NOREPLaceable
ENDRoutes

DEVice OSAQDIO3 MPCIPA SECROUTER NOAUTORESTART
LINK OSAQDIOL IPAQENET OSAQDIO3 IFSPEED 1000000000

GLOBALCONFig NOTCPStatistics ECSALIMIT 0K POOLLIMIT 0K NOMLSCHKTERMinate
HOME
  9.67.115.82 OSAQDIOL

INTERFace OSAQDI26 DEFINE IPAQENET6 PORTNAME OSAQDIO3 SECROUTER
          DUPADDRDET 1
  IPADDR FEC9:C2D4:1::9:67:115:82

IPCONFIG6 DATAGRamfwd(NOFWDMULTipath) NOSOURCEVIPA NOMULTIPATH HOPLimit
```

```
          255 ICMPErrorlimit 1000 NOIGNOREROUTERHOPLIMIT

IPCONFig ARPTO 1200 DATAGRamfwd(NOFWDMULTipath) NOSOURCEVIPA
         NOTCPSTACKSOURCEVIPA NOVARSUBNETTING NOSYSPLEXRouting
         REASSEMBLytimeout 60 TTL 64 NOPATHMTUDISCovery NOMULTIPATH
         NODYNAMICXCF NOIQDIORouting FORMAT(LONG)

ITRACE OFF AUTODAEMON
ITRACE OFF COMMAND
ITRACE OFF CONFig
ITRACE OFF SUBAGENt

PKTTRACE FULL LINKNAME=LOOPBACK PROT=* IP=* SRCPort=* DESTport=*
PKTTRACE ON LINKNAME=LOOPBACK
PKTTRACE FULL LINKNAME=OSAQDIOL PROT=* IP=* SRCPort=* DESTport=*
PKTTRACE ON LINKNAME=OSAQDIOL

PRImaryinterface OSAQDIOL

SACONFig COMMUNity public AGENT 161 ENABLED SETSDISAbled

SMFCONFIG TYPE118 NOTCPINIT NOTCPTERM NOFTPCLIENT NOTN3270CLIENT
          NOTCPIPStatistics TYPE119 NOTCPINIT NOTCPTERM NOFTPCLIENT
          NOTN3270CLIENT NOTCPIPSTATISTICS NOIFSTATISTICS
          NOPORTSTATISTICS NOTCPSTACK NOUDPTERM

SMFPARMS 0 0 0

SOMAXCONN 10

START OSAQDIO3
START OSAQDI26

TCPCONFIG INTerval 120 RESTRICTLowports TCPRCVBufrsize 16384
          TCPSENDBufrsize 16384 TCPMAXRCVBufrsize 262144 FINWAIT2TIME
          600 SENDGarbage FALSE TCPTIMESTAMP

UDPCONFIG RESTRICTLowports UDPCHKsum UDPRCVBufrsize 65535
          UDPSENDBufrsize 65535 UDPQueuelimit

Analysis of Tcp/Ip for TCPCS2 completed
```

TCPIPCS PROTOCOL

Use this subcommand to display information from TCP, UDP, and RAW protocol control blocks.

Syntax

```
                         ┌─(SUMMARY)─┐
►►──TCPIPCS──PROTOCOL──┼───────────┼──┬────────────────────────────────┬──────────►
                         └─(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 177 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:

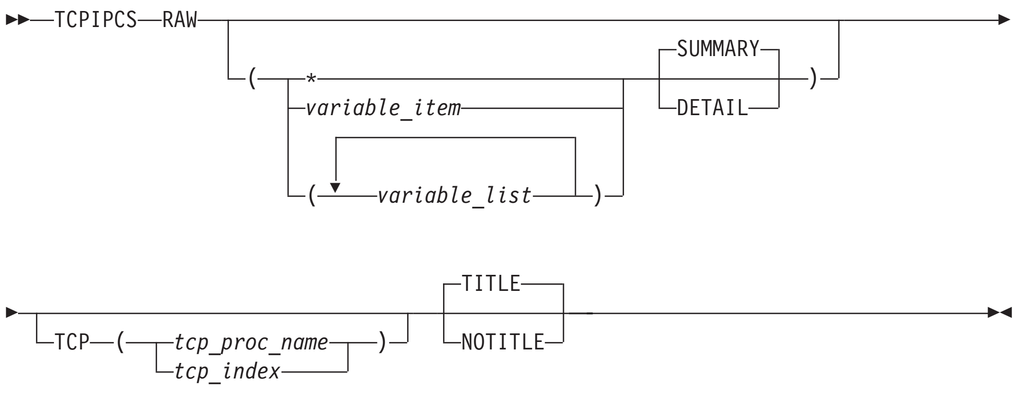
```
 TCPIPCS PROTOCOL
 Dataset: IPCS.MV21381.DUMPA
 Title:   SLIP DUMP ID=TC


 The address of the TSAB is: 13391BC0

 Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

 13391C00  1 TCPSVT    V2R10   1323B000 1323B0C8 07DE 04041405  Active
 13391C80  2 TCPSVT2   V2R10   00000000 00000000 07E8 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            RSTKDOWN. 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
TCPIP Main TCP Control Block (MTCB)
 MTCB: 1338E350
 +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  MTCPMTX............ 00000000  00000000  00000000  D7D60601
 +0028  MTCPAQMX........... 00000000  00000000  00000000  D7D60604
 +0038  MTCB_LIST_LOCK..... 00000000  00000000  00000000  D7D60604
 +0048  M_PORT_CEILING..... 00000FFF
 +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   ApplName UserID
 7F607108 00000002 TCPSVT   Closed      WLOUNBND 00040000 00000001 0.0.0.0
        0
 7F60A908 000083D7 FTPUNIX1 Listening   WLOIDLE  00200080 00000001 0.0.0.0
        0
 7F608D08 00000013 TCPSVT   Listening   WLOIDLE  00000080 00000001 0.0.0.0
        0
 7F617508 0000019B CICSRU   Listening   WLOIDLE  08200080 00000001 0.0.0.0
        0
 7F615108 00000144 INETD5   Listening   WLOIDLE  00200080 00000001 0.0.0.0
        0
...
 7F610108 0000878F NAMED4   TimeWait    WLOWIORL 80800C00 00000002 198.11.22.103
      53
 7F60C508 0000005C DHCP1    Established WLOXFER  01800000 00000001 198.11.25.104
    6000
 7F609D08 00000049 MISCSRV  Listening   WLOIDLE  00200000 00000001 0.0.0.0
       0
 7F608908 00000012 TCPSVT   Listening   WLOIDLE  00000080 00000001 0.0.0.0
       0
 7F60E108 00000063 TCPSVT   Established WLOXFER  80800000 00000001 127.0.0.1
    1030
 30 TCB(s) FOUND
 30 TCB(s) FORMATTED
 ----------------------------------------------------------------------------------

 User Datagram Protocol Control Block Summary
  MUCB: 7F7812A8
  +0000  UMUCBEYE. MUCB      USTKDOWN. 00        USTKLNKD. 01
  UAPAR.... 00        UDRVSTAT. 00
  +0008  UOPENPRT. 00000000  UFREEPRT. 0408      MCBMUTEX. 00000000
  00000000  00000000  D7D60402
  +0020  UDPCFG... 00000001  0000FFFF  0000FFFF  00000001  80000000
  00000000
  +0038  UDPCFG2.. 00000001  0000FFFF  0000FFFF  00000001  80000000
  00000000
  +0050  UDPMIB... 00001D1F  0000531F  00000000  0000166B
  USBCAST.. 00000000  USLPBACK. 00000000
  +0068  USDNTRTE. 00000000  USRCVBUF. 0000FFFF  USSNDBUF. 0000FFFF
  UFGPRC... 00        USERIALV. 0000065F
  +007C  USERIAL1. 0000065F  ULASTADR. 810B2068  ULASTPRT. 0043
  ULASTUCB. 7F5FD508  USERIAL2. 0000065F
   ...

 UCB      ResrcID  ResrcNm  TpiState IPAddr          Port
 7F5F6108 00000004 TCPSVT   WLOUNBND 0.0.0.0
 7F5FCD08 00000086 OSNMPD   WLOIDLE  127.0.0.1        161
 7F5FD508 0000005E DHCP1    WLOIDLE  129.11.32.1       67
 7F5FCF08 00000055 DHCP1    WLOIDLE  198.11.25.104   1027
 7F5FD308 0000005B NAMED    WLOIDLE  129.11.176.87     53
 7F5FD108 00000059 DHCP3    WLOIDLE  0.0.0.0         6001
 7F5FC908 00000048 MISCSRV  WLOIDLE  0.0.0.0           19
 7F5FC708 00000047 MISCSRV  WLOIDLE  0.0.0.0           19
....
 7F5F6B08 00000017 MISCSRV  WLOIDLE  0.0.0.0            7
 7F5F6908 00000014 PORTMAP  WLOIDLE  0.0.0.0          111
 56 UCB(s) FOUND
 56 UCB(s) FORMATTED

 Analysis of Tcp/Ip for TCPSVT completed
```

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



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 API control blocks 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 hexadecimal digit.

In addition to the variable parameters described above, you can specify the following keyword parameters:

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 177 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 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
TCPIP Raw Control Block Analysis
Master Raw Control Block (MRCB)
 MRAWCB: 7F407208
 +0000  RMRCBEYE. MRCB      RSTKLNKD. 01        RDRVSTAT. 01
 +000C  R6STKLNKD.                    01
 +000D  R6DRVSTAT.                    01
 +0010  MRCMUTEX. 00000000  00000000  00000000  D7D60501
        RSBCAST.. 00000000  RSDNTRTE. 00000000  RSRC
 +002C  RSSNDBUF. 0000FFFF  RDIPTOS.. 00        RDIPTTL.. 00
        RIPWRQ@.. 7F621DA8  RIPRDQ@.. 7F621D68  RHAS
 +0040  RIP6WRQ@. 7F686468  RIP6RDQ@. 7F686428  R6HASH@.. 7F407374
 +004C  R6DFFLTR. 7F781FFF  FFFFFFFF  FFFFFFFF  FFFFFFFF  003FFFFF
        FFFFFFFF  FFFFFFFF  FFFFFFFF

IPv4 Raw Hash Table Entries
ID  First    Last
0   7F52C390 7F52C390
15  7F52C110 7F52C110

IPv6 Raw Hash Table Entries
ID  First    Last
0   7F2073BC 7F2073BC

IPv4 RAW Connections
RCB      ResrcID  ResrcNm  TpiState   ProtocolId  DestAddr
7F52C388 00000006 TCPCS    WLOIDLE    0           0.0.0.0
7F52C108 00000008 TCPCS    WLOIDLE    255         0.0.0.0

IPv6 RAW Connections
RCB      ResrcID  ResrcNm  TpiState   ProtocolId  DestAddr
7F207208 0000000E TCPCS    WLOIDLE    0           ::0
```

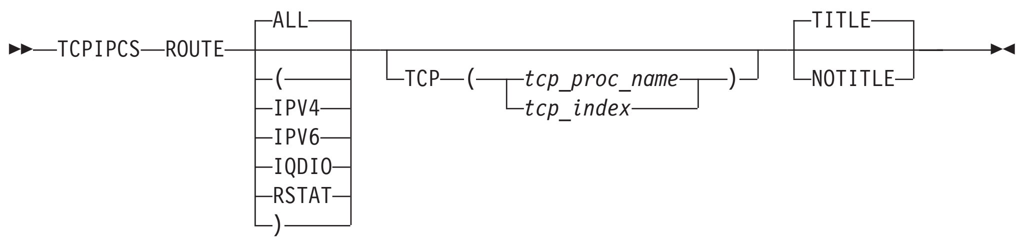
```
3 RCB(s) FOUND
3 RCB(s) FORMATTED

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:

- Address device name
- Type
- Protocol
- Destination IP address
- Gateway IP address
- Physical interface control block address

Syntax



Parameters

ALL
Display structure of all route table information. ALL is the default.

IPV4
All IPv4 search tree and update tree routes.

IPV6
All IPv6 search tree and update tree routes.

IQDIO
All 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 177 for a description of these parameters.

Restriction: If you specify multiple keywords from the set {ALL, IPV4, IPV6, IQDIO, RSTAT}, all of them are used.

Sample output of the TCPIPCS ROUTE subcommand

The following is sample output of the TCPIPCS ROUTE subcommand:

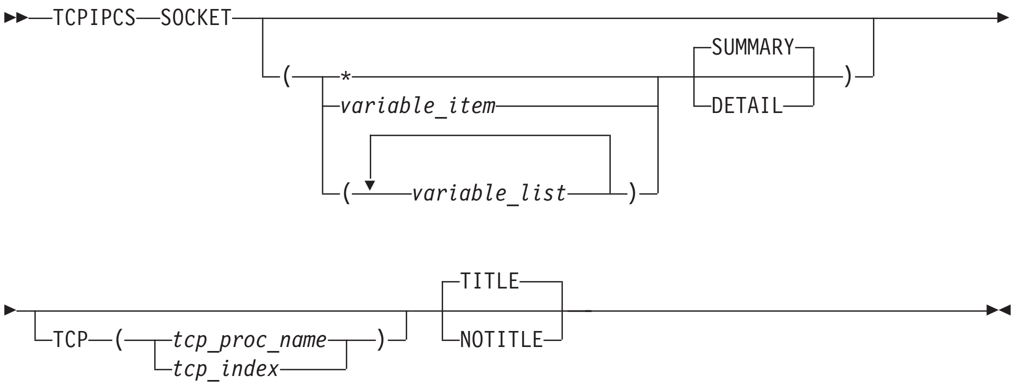
```
TCPIPCS ROUTE ( IPV4 IPV6 )  TCP ( 1 )
Dataset: IPCS.MV25332.DUMPC
Title:   SLIP DUMP ID=TKO2
The address of the TSAB is: 1EF11570
Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status
1EF115B0  1 TCPSVT    V1R5    1DE47000 1DE470C8 0057 C5BF755F  Active
```

```
1EF11630  2 TCPSVT3   V1R5    1DD4D000 1DD4D0C8 0074 C5BF755F  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 TCPSVT.  Index: 1
TCPIP Route Analysis
IPv4 Routes in Search Table
Rte@     DeviceName Type    Protocol Pif@      IP Addresses
7EED5708 LCGE2SP32  Host    OSPF     7F539088  Destination: 216.51.55.202
                                               Gateway    : 198.11.32.110
7EEB7888 LMGE2RU33  Host    OSPF     7F538608  Destination: 216.51.55.202
                                               Gateway    : 198.11.33.104
7EF0DBE8 LCGE2SP32  Host    OSPF     7F539088  Destination: 216.51.55.155
                                               Gateway    : 198.11.32.110
IPv6 Routes in Search Table
Rte@     DeviceName Type    Protocol Pif@      IP Addresses
7EE718E8 LV6OGETH2  Direct  ICMP     7F53C888  Destination: FEC0:176:11:48::0
                                               Gateway    : ::0
7EE46068 LV6OGETH2  Default ICMP     7F53C888  Destination: ::0
                                               Gateway    : FE80::230:71FF:FED3:5160
IPv4 Routes in Update Table
Rte@     DeviceName Type    Protocol Pif@      IP Addresses
7EED5708 LCGE2SP32  Host    OSPF     7F539088  Destination: 216.51.55.202
                                               Gateway    : 198.11.32.110
7EEB7888 LMGE2RU33  Host    OSPF     7F538608  Destination: 216.51.55.202
                                               Gateway    : 198.11.33.104
7EF0DBE8 LCGE2SP32  Host    OSPF     7F539088  Destination: 216.51.55.155
                                               Gateway    : 198.11.32.110
7EEB7A08 LMGE2RU33  Host    OSPF     7F538608  Destination: 216.51.55.155
                                               Gateway    : 198.11.33.104
7EED5A08 LCGE2SP32  Host    OSPF     7F539088  Destination: 216.22.1.202
                                               Gateway    : 198.11.32.110
7EEB7B88 LMGE2RU33  Host    OSPF     7F538608  Destination: 216.22.1.202
                                               Gateway    : 198.11.33.104
IPv6 Routes in Update Table
Rte@     DeviceName Type    Protocol Pif@      IP Addresses
7EE718E8 LV6OGETH2  Direct  ICMP     7F53C888  Destination: FEC0:176:11:48::0
                                               Gateway    : ::0
7EE46068 LV6OGETH2  Default ICMP     7F53C888  Destination: ::0
                                               Gateway    : FE80::230:71FF:FED3:5160
Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS SOCKET

Use this subcommand to display information from TCP/IP socket control blocks.

Syntax



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 1-8 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 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 177 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

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

13391C00  1 TCPSVT    V2R10   1323B000 1323B0C8 07DE 04041405  Active
13391C80  2 TCPSVT2   V2R10   00000000 00000000 07E8 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
```

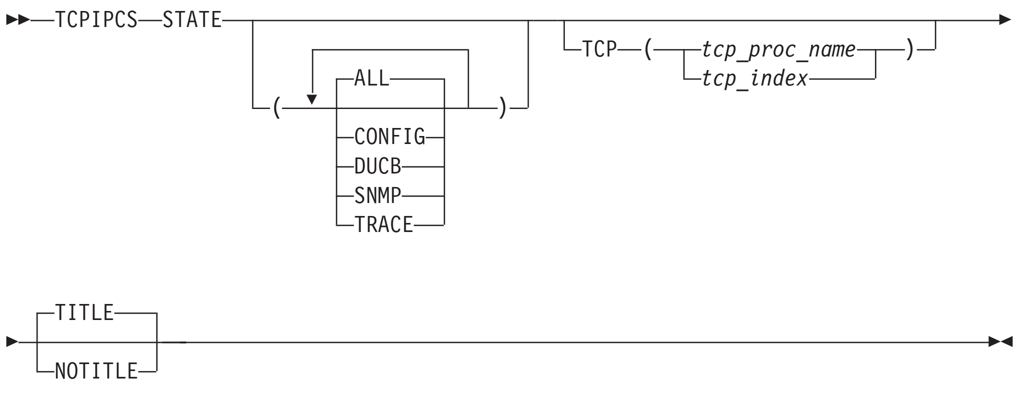
```
TCPIP Socket Analysis

SCB      CID      Protocol    SockOpts ScbFlags ResrcNm
12D40108 00000008 RAW         00000000 00280000 TCPSVT
12D40208 0000000B UDP         00000000 00280000 TCPSVT
12D40308 0000000C TCP         00020000 C0280000 TCPSVT
12D40408 0000000E UDP         00000000 00280000 TCPSVT
12D40508 0000000F TCP         00000000 B0280000 TCPSVT
12D40608 00000010 TCP         00020000 90280000 TCPSVT
12D40708 00000067 TCP         08000000 90280000 OMPROUTE
12D40808 00000012 TCP         00400000 C0000000 TCPSVT
12D40908 00000013 TCP         00400000 C0000000 TCPSVT
12D40A08 00000014 UDP         00000000 80280000 PORTMAP
12D40B08 00000015 TCP         00000000 C0280000 PORTMAP
 ...
12D44C08 00000058 TCP         00000000 C0000000 DHCP3
12D44D08 00000059 UDP         00000000 80280000 DHCP3
12D44E08 0000005A TCP         00400000 C0280000 NAMED
12D44F08 0000005B UDP         00400000 80280000 NAMED

79 Socket control blocks were found
79 Socket control blocks were formatted


Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS 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



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 be displayed.)

TRACE
Display only trace state information.

TCP, TITLE, NOTITLE
See "Parameters" on page 177 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:

```
TCPIPCS STATE
Dataset: IPCS.R450697.V6TCBD1
Title:   TCPCS2 CLIENT SIDE


The address of the TSAB is: 09DBE1A0

Tseb     SI Procedure Version Tsdb     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

TCPIP State

TCPIP Status:
 Procedure:     TCPCS2
 Version:       V1R5
 Status:        Active
 Asid:          0034
 Started:       2001/11/07 17:34:16
 Ended:         2001/11/07 17:41:23
 Active:        00:07:06.980607 hours

Major Control Blocks
 TSEB:          9DBE260       TSDB:          96C9000
 TSDX:          96C90C8       TCA:           97E9610
 ITCVT:         96C93C8       ITSTOR:        96C95E0
 DUAF:          92F0010       MRCB:          7F40E208
 MTCB:          96CE910       MUCB:          7F512248
 IPMAIN:        95A83D0       Streams_root:  7F6BFD08
 TosMains:      96CE770       MIB2:          97E9010
 CdCb:          97E9390       User:          9EDD060
 Conf:          96CC110       Stks:          9D098B0
 IPMAIN6:       96CE470
================================================================================
TCPIP Subtasks
Task      Tcb       FirstRB   EotECB    StopEcb   CmpCode   RsnCode   RTWA
```

```
EZBTCPIP  007EC920  007ECF98  807FD178            00000000  00000000  00000000
EPWPITSK  007EC690  007EC608  00000000            00000000  00000000  00000000
EZBITTUB  007EC400  007E2AB8  00000000  807EC0E8  00000000  00000000  00000000
EZBIPSUB  007EC170  007EC060  00000000  807EC060  00000000  00000000  00000000
EZBIEOER  007E3D90  007E3D08  00000000  807E3D08  00000000  00000000  00000000
EZACDMSM  007E3B70  007E3A88  00000000  807E3A88  00000000  00000000  00000000
EZACFMMN  007E38F0  007E3838  807ECF98  807E3838  00000000  00000000  00000000
EZBTZMST  007E34F0  007E33D0  007ECF98            00000000  00000000  00000000
EZBTTSSL  007E3140  007E30B8  807E33D0            00000000  00000000  00000000
EZBTMCTL  007E2D90  007E32F8  807E33D0            00000000  00000000  00000000
EZACFALG  007E2BF8  007E2B60  807ECF98            00000000  00000000  00000000
EZASASUB  007E28D0  007E2848  807ECF98  807E2848  00000000  00000000  00000000
================================================================================
Storage Cache Information
  Total CSA Allocated:               4,959,664
  Tcp/ip CSA Limit:                     2,147M
  Total CSA Elements:                       52
  Cache Delay:                             180 seconds
  Scan Delay:                               75 seconds
  Total cache allocated:                22,040
  Total cache elements:                      3
  Total freed elements:                      0
  Last cache scan time:     2001/11/07 21:39:45

CSM Status
  ECSA Storage:           OK
  Data Space Storage:     OK
  Fixed Storage:          OK
Alet: 01FF0009               Dspname: CSM31001

================================================================================
Dispatchable Unit Status
  DUCB Initializations:                 7,091
  DUCB Expansions:                         63
  Percent DUCB expansions:                  0 %
  Last DUCB scan time:     2001/11/07 21:40:48

   1 DUAT control block(s) were found in the DUAF at 092F0010
  82 Dispatchable units were found.
  No DUs indicate abend.
================================================================================

CTrace Status:
   Member Name  : CTIEZBN0
   Buffer Size  : 20,972K
   Options      : Socket PFS TCP Internet VtamData
   Asid List    : ()
   JobNameList  : ()
   PortList     : ()
   IpAddrList   : ()
   Xwriter      : Disconnected
   Dwriter      : Disconnected
   Trace Count  : 313,343
   Lost Count   : 0
   Lost Time    : 1900/01/01 00:00:00
   Wrap Count   : 10
   Wrap Time    : 2001/11/07 21:40:51
================================================================================

Device Interface: 7F6AA408
  Device: LOOPBACK        Devtype: LOOPBACK        State: Active
  Address: **** ****

Physical Interface: 7F679468
  Name: LOOPBACK         Protocol: LOOPBACK        State: Active
    NetNum: 0  QueSize: 0  Bytein: 3,861      Byteout: 3,861
     Index: 2
  Bsd Routing Parameters:
   MtuSize: 0                  Metric: 0
   SubnetMask: 0.0.0.0         DestAddr: 0.0.0.0
  SNMP Input Counters:
         Octets:               3,861   Unicast:                 61
     NonUnicast:                   0  Discarded:                 0
          Error:                   0  Unkn Type:                 0
      Broadcast:                   0  Multicast:                 0
  SNMP Output Counters:
         Octets:               3,861   Unicast:                 61
     NonUnicast:                   0  Discarded:                 0
          Error:                   0  Queue Len:                 0
      Broadcast:                   0  Multicast:                 0

  IPv4 Search Patricia tree     Address: 7F679F68
    Search Ptree Reader Count: 0

     Route: 7F6AA288
      Name: LOOPBACK          Type: Direct           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: 2          Tos: 0
      Metric1:  0             Metric2: -1
      Metric3:  -1            Metric4: -1
      Metric5:  -1            Age: 2001/11/07 21:34:25
  IPv6 Search Patricia tree     Address: 7F3FDF08
    Search Ptree Reader Count: 0

  IPv4 Update Patricia tree     Address: 7F679F08
    No update to Route Ptree is pending.

     Route: 7F6AA288
      Name: LOOPBACK          Type: Direct           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: 2          Tos: 0
      Metric1:  0             Metric2: -1
      Metric3:  -1            Metric4: -1
      Metric5:  -1            Age: 2001/11/07 21:34:25
  IPv6 Update Patricia tree     Address: 7F3FDF08
    No update to Route Ptree is pending.


Logical Interface: 7F6792E8
  Name: LOOPBACK               Protocol: LOOPBACK         State: Active
  Subnet Mask: 255.255.255.255  Addr: 127.0.0.1
  Mtu Size: 65535
  Packet Trace Parameters: 7F1AFA48
    Protocol: 0    TrRecCnt: 48  PckLength: 65535
    SrcPort: 0     DestPort: 0
    SubnetMask: 255.255.255.255  DestAddr: 0.0.0.0
Physical Interface: 7F503488
  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
     NonUnicast:                   0  Discarded:                 0
          Error:                   0  Queue Len:                 0
      Broadcast:                    0  Multicast:                    0

  IPv4 Search Patricia tree     Address: 7F679F68
    Search Ptree Reader Count: 0
  IPv6 Search Patricia tree     Address: 7F3FDF08
    Search Ptree Reader Count: 0

     Route: 7F3FD9C8
      Name: LOOPBACK6         Type: Direct           State: Active
      Subnet Prefix: 128            Addr: ::1
      Protocol   : Configuration    Gate: ::0
      Mtu Size: 0             Ref Cnt: 0          Tos: 28
      Metric1:  0             Metric2: -1
      Metric3:  -1            Metric4: -1
      Metric5:  -1            Age: 2001/11/07 21:34:28

  IPv4 Update Patricia tree     Address: 7F679F08
    No update to Route Ptree is pending.
  IPv6 Update Patricia tree     Address: 7F3FDF08
    No update to Route Ptree is pending.

     Route: 7F3FD9C8
      Name: LOOPBACK6         Type: Direct           State: Active
      Subnet Prefix: 128            Addr: ::1
      Protocol   : Configuration    Gate: ::0
      Mtu Size: 0             Ref Cnt: 0          Tos: 28
      Metric1:  0             Metric2: -1
      Metric3:  -1            Metric4: -1
      Metric5:  -1            Age: 2001/11/07 21:34:28


Logical Interface: 7F3FDCE8
  Name: LOOPBACK6               Protocol: LOOPBACK6       State: Active
  Subnet Prefix: 128            Addr: ::1
  Mtu Size: 65535
  Packet Trace Parameters: 7F1AFB88
    Protocol: 0    TrRecCnt: 0   PckLength: 65535
    SrcPort: 0     DestPort: 0
    SubnetMask: 255.255.255.255  DestAddr: ::0
================================================================================
Device Interface: 7F1ED408
  Device: OSAQDIO3        Devtype: MPCIPA          State: Active
  Address: **** ****
SAP:
  UserID: 10010000     TransId: 0001012F  ProviderId: 00010131
  Data@: 8928E0CC   ReqSignal@: 85A2F810  RspSignal@: 85A2F810
  State: Active         Retry:    0  Restart:    0  Xstatus: 0
Connection 2:
  UserID: 00000000 ProviderId: 90010001
  Data@: 00000000  ReqSignal@: 00000000 RspSignal@: 00000000
  State: Reset        linknum: 00 flags 00

Physical Interface: 7F503B88
  Name: OSAQDIOL         Protocol: IPAQENET         State: Active
    NetNum: 0  QueSize: 0  Bytein: 0          Byteout: 0
     Index: 5
  Bsd Routing Parameters:
   MtuSize: 0                   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  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: 7F679F68
   Search Ptree Reader Count: 0

    Route: 7F3FD188
     Name: OSAQDIOL          Type: Direct           State: Active
     Subnet Mask: 255.255.255.255  Addr: 9.67.115.83
     Protocol   : Configuration    Gate: 0.0.0.0
     Mtu Size: 8992          Ref Cnt: 0          Tos: 0
     Metric1:  0             Metric2: -1
     Metric3:  -1            Metric4: -1
     Metric5:  -1            Age: 2001/11/07 21:34:31

    Route: 7F3081E8
     Name: OSAQDIOL          Type: Direct           State: Active
     Subnet Mask: 255.255.255.255  Addr: 9.67.115.79
     Protocol   : Configuration    Gate: 0.0.0.0
     Mtu Size: 1500          Ref Cnt: 0          Tos: 0
     Metric1:  0             Metric2: -1
     Metric3:  -1            Metric4: -1
     Metric5:  -1            Age: 2001/11/07 21:34:31

    Route: 7F1EB128
     Name: OSAQDIOL          Type: Direct           State: Active
     Subnet Mask: 255.255.255.255  Addr: 9.67.115.82
     Protocol   : Configuration    Gate: 0.0.0.0
     Mtu Size: 8992          Ref Cnt: 0          Tos: 0
     Metric1:  0             Metric2: -1
     Metric3:  -1            Metric4: -1
     Metric5:  -1            Age: 2001/11/07 21:34:31

    Route: 7F628348
     Name: OSAQDIOL          Type: Direct           State: Active
     Subnet Mask: 255.255.255.192  Addr: 9.67.115.0
     Protocol   : Configuration    Gate: 0.0.0.0
     Mtu Size: 8992          Ref Cnt: 0          Tos: 0
     Metric1:  0             Metric2: -1
     Metric3:  -1            Metric4: -1
     Metric5:  -1            Age: 2001/11/07 21:34:31

    Route: 7F6AA108
     Name: OSAQDIOL          Type: Default          State: Active
     Subnet Mask: 0.0.0.0          Addr: 0.0.0.0
     Protocol   : Configuration    Gate: 9.67.115.1
     Mtu Size: 8992          Ref Cnt: 0          Tos: 0
     Metric1:  1             Metric2: -1
     Metric3:  -1            Metric4: -1
     Metric5:  -1            Age: 2001/11/07 21:34:31
 IPv6 Search Patricia tree     Address: 7F3FDF08
   Search Ptree Reader Count: 0

 IPv4 Update Patricia tree     Address: 7F679F08
   No update to Route Ptree is pending.

    Route: 7F3FD188
     Name: OSAQDIOL          Type: Direct           State: Active
     Subnet Mask: 255.255.255.255  Addr: 9.67.115.83
     Protocol   : Configuration    Gate: 0.0.0.0
     Mtu Size: 8992          Ref Cnt: 0          Tos: 0
     Metric1:  0             Metric2: -1
     Metric3:  -1            Metric4: -1
     Metric5:  -1            Age: 2001/11/07 21:34:31

    Route: 7F3081E8
     Name: OSAQDIOL          Type: Direct           State: Active
```

```
        Subnet Mask: 255.255.255.255  Addr: 9.67.115.79
        Protocol   : Configuration    Gate: 0.0.0.0
        Mtu Size: 1500          Ref Cnt: 0          Tos: 0
        Metric1:  0             Metric2: -1
        Metric3:  -1            Metric4: -1
        Metric5:  -1            Age: 2001/11/07 21:34:31

       Route: 7F1EB128
        Name: OSAQDIOL          Type: Direct           State: Active
        Subnet Mask: 255.255.255.255  Addr: 9.67.115.82
        Protocol   : Configuration    Gate: 0.0.0.0
        Mtu Size: 8992          Ref Cnt: 0          Tos: 0
        Metric1:  0             Metric2: -1
        Metric3:  -1            Metric4: -1
        Metric5:  -1            Age: 2001/11/07 21:34:31

       Route: 7F628348
        Name: OSAQDIOL          Type: Direct           State: Active
        Subnet Mask: 255.255.255.192  Addr: 9.67.115.0
        Protocol   : Configuration    Gate: 0.0.0.0
        Mtu Size: 8992          Ref Cnt: 0          Tos: 0
        Metric1:  0             Metric2: -1
        Metric3:  -1            Metric4: -1
        Metric5:  -1            Age: 2001/11/07 21:34:31

       Route: 7F6AA108
        Name: OSAQDIOL          Type: Default          State: Active
        Subnet Mask: 0.0.0.0          Addr: 0.0.0.0
        Protocol   : Configuration    Gate: 9.67.115.1
        Mtu Size: 8992          Ref Cnt: 0          Tos: 0
        Metric1:  1             Metric2: -1
        Metric3:  -1            Metric4: -1
        Metric5:  -1            Age: 2001/11/07 21:34:31
    IPv6 Update Patricia tree     Address: 7F3FDF08
      No update to Route Ptree is pending.

     Address Translate Entry: 7F1AF5E8
       addr: 9.67.115.79      flags:  C0  ttldlt:  0
       retries: 0
     Address Translate Entry: 7F1AF548
       addr: 9.67.115.82      flags:  C0  ttldlt:  0
       retries: 0
     Address Translate Entry: 7F1AF4A8
       addr: 9.67.115.1       flags:  C0  ttldlt:  0
       retries: 0

  Logical Interface: 7F1ED008
    Name: OSAQDIOL                Protocol: IPAQENET         State: Active
    Subnet Mask: 255.255.255.255  Addr: 9.67.115.82
    Mtu Size: 1492
    Packet Trace Parameters: 7F1AFAE8
      Protocol: 0    TrRecCnt: 0   PckLength: 65535
      SrcPort: 0     DestPort: 0
      SubnetMask: 255.255.255.255  DestAddr: 0.0.0.0
  Physical Interface: 7F503F08
    Name: OSAQDI26          Protocol: IPAQENET6        State: Active
      NetNum: 0  QueSize: 0  Bytein: 54,570K    Byteout: 285,212
        Index: 6
    Bsd Routing Parameters:
     MtuSize: 0                   Metric: 0
     SubnetMask: 0.0.0.0          DestAddr: 0.0.0.0
    SNMP Input Counters:
          Octets:              54,570K    Unicast:              29,240
       NonUnicast:                   0  Discarded:                   0
            Error:                   0  Unkn Type:                   0
        Broadcast:                   0  Multicast:                   7
    SNMP Output Counters:
           Octets:              285,212   Unicast:                2,401
       NonUnicast:                   0  Discarded:                   0
            Error:                   0  Queue Len:                   0
        Broadcast:                   0  Multicast:                  12

    IPv4 Search Patricia tree     Address: 7F679F68
      Search Ptree Reader Count: 0
    IPv6 Search Patricia tree     Address: 7F3FDF08
      Search Ptree Reader Count: 0

       Route: 7F1EBE88
        Name: OSAQDI26          Type: Host            State: Active
        Subnet Prefix: 128            Addr: FEC9:C2D4:1::9:67:115:83
        Protocol   : Configuration    Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
        Mtu Size: 0             Ref Cnt: 0          Tos: 28
        Metric1:  1             Metric2: -1
        Metric3:  -1            Metric4: -1
        Metric5:  -1            Age: 2001/11/07 21:34:31

       Route: 7F1EC3E8
        Name: OSAQDI26          Type: Direct          State: Active
        Subnet Prefix: 128            Addr: FEC9:C2D4:1::9:67:115:79
        Protocol   : Configuration    Gate: ::0
        Mtu Size: 5             Ref Cnt: 0          Tos: 0
        Metric1:  0             Metric2: -1
        Metric3:  -1            Metric4: -1
        Metric5:  -1            Age: 2001/11/07 21:34:31

       Route: 7F1C7C48
        Name: OSAQDI26          Type: Direct          State: Active
        Subnet Prefix: 128            Addr: FEC9:C2D4:1::9:67:115:82
        Protocol   : Configuration    Gate: ::0
        Mtu Size: 0             Ref Cnt: 0          Tos: 28
        Metric1:  0             Metric2: -1
        Metric3:  -1            Metric4: -1
        Metric5:  -1            Age: 2001/11/07 21:34:36

       Route: 7F1ECE88
        Name: OSAQDI26          Type: Direct          State: Active
        Subnet Prefix: 128            Addr: FEC9:C2D4:1::206:2AFF:FE66:C81C
        Protocol   : Configuration    Gate: ::0
        Mtu Size: 0             Ref Cnt: 0          Tos: 0
        Metric1:  0             Metric2: -1
        Metric3:  -1            Metric4: -1
        Metric5:  -1            Age: 2001/11/07 21:34:31

       Route: 7F1C7E88
        Name: OSAQDI26          Type: Direct          State: Active
        Subnet Prefix: 128            Addr: FE80::2:559A:3F5F:1
        Protocol   : Configuration    Gate: ::0
        Mtu Size: 2             Ref Cnt: 0          Tos: 28
        Metric1:  0             Metric2: -1
        Metric3:  -1            Metric4: -1
        Metric5:  -1            Age: 2001/11/07 21:34:32

       Route: 7F1ECAE8
        Name: OSAQDI26          Type: Direct          State: Active
        Subnet Prefix: 128            Addr: 50C9:C2D4:1::206:2AFF:FE66:C81C
        Protocol   : Configuration    Gate: ::0
        Mtu Size: 0             Ref Cnt: 0          Tos: 0
        Metric1:  0             Metric2: -1
        Metric3:  -1            Metric4: -1
        Metric5:  -1            Age: 2001/11/07 21:34:31

       Route: 7F1EB588
        Name: OSAQDI26          Type: Prefix          State: Active
        Subnet Prefix: 112            Addr: 50C9:C2D4:1::0
        Protocol   : Configuration    Gate: 50C9:C2D4:1::206:2AFF:FE66:C81C
        Mtu Size: 0            Ref Cnt: 0          Tos: 28
        Metric1:  1            Metric2: -1
        Metric3:  -1           Metric4: -1
        Metric5:  -1           Age: 2001/11/07 21:34:31

       Route: 7F1C7768
        Name: OSAQDI26         Type: Direct          State: Active
        Subnet Prefix: 64            Addr: 50C9:C2D4:1::0
        Protocol   : ICMP            Gate: ::0
        Mtu Size: 0            Ref Cnt: 0          Tos: 28
        Metric1:  0            Metric2: -1
        Metric3:  -1           Metric4: -1
        Metric5:  -1           Age: 2001/11/07 21:34:57

       Route: 7F1C74C8
        Name: OSAQDI26         Type: Direct          State: Active
        Subnet Prefix: 64            Addr: 50C9:FFFF:1::0
        Protocol   : ICMP            Gate: ::0
        Mtu Size: 0            Ref Cnt: 0          Tos: 28
        Metric1:  0            Metric2: -1
        Metric3:  -1           Metric4: -1
        Metric5:  -1           Age: 2001/11/07 21:34:57

       Route: 7F1EBA28
        Name: OSAQDI26         Type: Prefix          State: Active
        Subnet Prefix: 112           Addr: FEC9:C2D4:1::0
        Protocol   : Configuration    Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
        Mtu Size: 0            Ref Cnt: 0          Tos: 28
        Metric1:  1            Metric2: -1
        Metric3:  -1           Metric4: -1
        Metric5:  -1           Age: 2001/11/07 21:34:31

       Route: 7F1C7A08
        Name: OSAQDI26         Type: Direct          State: Active
        Subnet Prefix: 64            Addr: FEC9:C2D4:1::0
        Protocol   : ICMP            Gate: ::0
        Mtu Size: 0            Ref Cnt: 0          Tos: 28
        Metric1:  0            Metric2: -1
        Metric3:  -1           Metric4: -1
        Metric5:  -1           Age: 2001/11/07 21:34:57

       Route: 7F1EC748
        Name: OSAQDI26         Type: Default         State: Active
        Subnet Prefix: 0             Addr: ::0
        Protocol   : Configuration    Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
        Mtu Size: 0            Ref Cnt: 0          Tos: 28
        Metric1:  1            Metric2: -1
        Metric3:  -1           Metric4: -1
        Metric5:  -1           Age: 2001/11/07 21:34:31

    IPv4 Update Patricia tree     Address: 7F679F08
      No update to Route Ptree is pending.
    IPv6 Update Patricia tree     Address: 7F3FDF08
      No update to Route Ptree is pending.

       Route: 7F1EBE88
        Name: OSAQDI26         Type: Host            State: Active
        Subnet Prefix: 128           Addr: FEC9:C2D4:1::9:67:115:83
        Protocol   : Configuration    Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
        Mtu Size: 0            Ref Cnt: 0          Tos: 28
        Metric1:  1            Metric2: -1
        Metric3:  -1           Metric4: -1
        Metric5:  -1           Age: 2001/11/07 21:34:31

       Route: 7F1EC3E8
        Name: OSAQDI26         Type: Direct          State: Active
```

```
 Subnet Prefix: 128           Addr: FEC9:C2D4:1::9:67:115:79
 Protocol   : Configuration    Gate: ::0
 Mtu Size: 5            Ref Cnt: 0          Tos: 0
 Metric1:  0            Metric2: -1
 Metric3:  -1           Metric4: -1
 Metric5:  -1           Age: 2001/11/07 21:34:31

Route: 7F1C7C48
 Name: OSAQDI26         Type: Direct            State: Active
 Subnet Prefix: 128           Addr: FEC9:C2D4:1::9:67:115:82
 Protocol   : Configuration    Gate: ::0
 Mtu Size: 0            Ref Cnt: 0          Tos: 28
 Metric1:  0            Metric2: -1
 Metric3:  -1           Metric4: -1
 Metric5:  -1           Age: 2001/11/07 21:34:36

Route: 7F1ECE88
 Name: OSAQDI26         Type: Direct            State: Active
 Subnet Prefix: 128           Addr: FEC9:C2D4:1::206:2AFF:FE66:C81C
 Protocol   : Configuration    Gate: ::0
 Mtu Size: 0            Ref Cnt: 0          Tos: 0
 Metric1:  0            Metric2: -1
 Metric3:  -1           Metric4: -1
 Metric5:  -1           Age: 2001/11/07 21:34:31

Route: 7F1C7E88
 Name: OSAQDI26         Type: Direct            State: Active
 Subnet Prefix: 128           Addr: FE80::2:559A:3F5F:1
 Protocol   : Configuration    Gate: ::0
 Mtu Size: 2            Ref Cnt: 0          Tos: 28
 Metric1:  0            Metric2: -1
 Metric3:  -1           Metric4: -1
 Metric5:  -1           Age: 2001/11/07 21:34:32

Route: 7F1ECAE8
 Name: OSAQDI26         Type: Direct            State: Active
 Subnet Prefix: 128           Addr: 50C9:C2D4:1::206:2AFF:FE66:C81C
 Protocol   : Configuration    Gate: ::0
 Mtu Size: 0            Ref Cnt: 0          Tos: 0
 Metric1:  0            Metric2: -1
 Metric3:  -1           Metric4: -1
 Metric5:  -1           Age: 2001/11/07 21:34:31

Route: 7F1EB588
 Name: OSAQDI26         Type: Prefix            State: Active
 Subnet Prefix: 112           Addr: 50C9:C2D4:1::0
 Protocol   : Configuration    Gate: 50C9:C2D4:1::206:2AFF:FE66:C81C
 Mtu Size: 0            Ref Cnt: 0          Tos: 28
 Metric1:  1            Metric2: -1
 Metric3:  -1           Metric4: -1
 Metric5:  -1           Age: 2001/11/07 21:34:31

Route: 7F1C7768
 Name: OSAQDI26         Type: Direct            State: Active
 Subnet Prefix: 64            Addr: 50C9:C2D4:1::0
 Protocol   : ICMP             Gate: ::0
 Mtu Size: 0            Ref Cnt: 0          Tos: 28
 Metric1:  0            Metric2: -1
 Metric3:  -1           Metric4: -1
 Metric5:  -1           Age: 2001/11/07 21:34:57

Route: 7F1C74C8
 Name: OSAQDI26         Type: Direct            State: Active
 Subnet Prefix: 64            Addr: 50C9:FFFF:1::0
 Protocol   : ICMP             Gate: ::0
 Mtu Size: 0            Ref Cnt: 0          Tos: 28
 Metric1:  0            Metric2: -1
```

```
       Metric3:  -1           Metric4: -1
       Metric5:  -1           Age: 2001/11/07 21:34:57

      Route: 7F1EBA28
       Name: OSAQDI26         Type: Prefix          State: Active
       Subnet Prefix: 112           Addr: FEC9:C2D4:1::0
       Protocol   : Configuration    Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
       Mtu Size: 0            Ref Cnt: 0          Tos: 28
       Metric1:  1            Metric2: -1
       Metric3:  -1           Metric4: -1
       Metric5:  -1           Age: 2001/11/07 21:34:31

      Route: 7F1C7A08
       Name: OSAQDI26         Type: Direct          State: Active
       Subnet Prefix: 64            Addr: FEC9:C2D4:1::0
       Protocol   : ICMP            Gate: ::0
       Mtu Size: 0            Ref Cnt: 0          Tos: 28
       Metric1:  0            Metric2: -1
       Metric3:  -1           Metric4: -1
       Metric5:  -1           Age: 2001/11/07 21:34:57

      Route: 7F1EC748
       Name: OSAQDI26         Type: Default         State: Active
       Subnet Prefix: 0             Addr: ::0
       Protocol   : Configuration    Gate: FEC9:C2D4:1::206:2AFF:FE66:C81C
       Mtu Size: 0            Ref Cnt: 0          Tos: 28
       Metric1:  1            Metric2: -1
       Metric3:  -1           Metric4: -1
       Metric5:  -1           Age: 2001/11/07 21:34:31


Logical Interface: 7F6BF028
  Name: OSAQDI26              Protocol: IPAQENET6       State: Active
  Subnet Prefix: 128          Addr: FE80::2:559A:3F5F:1
  Mtu Size: 8992
  Packet Trace Parameters: 7F1AFC28
    Protocol: 0    TrRecCnt: 31646  PckLength: 65535
    SrcPort: 0     DestPort: 0
    SubnetMask: 255.255.255.255  DestAddr: ::0
Logical Interface: 7F3083C8
  Name: OSAQDI26              Protocol: IPAQENET6       State: Active
  Subnet Prefix: 0            Addr: FEC9:C2D4:1::9:67:115:82
  Mtu Size: 1492
================================================================================

Analysis of Tcp/Ip for TCPCS2 completed
```

TCPIPCS STORAGE

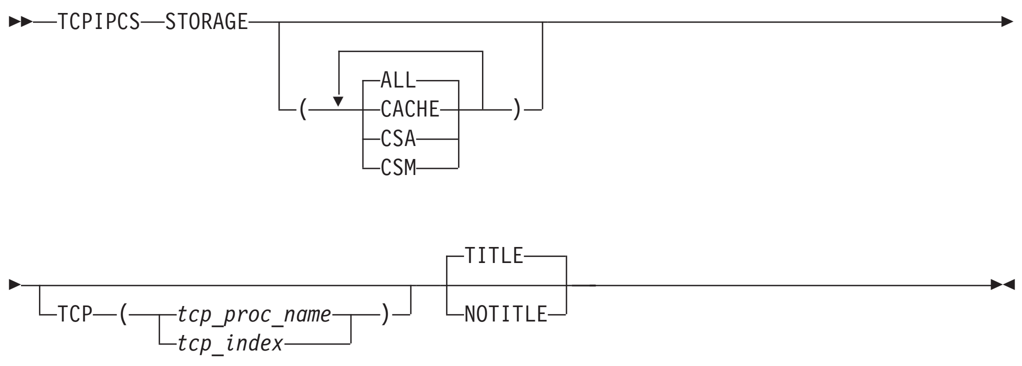
Use this subcommand to display the TCP/IP storage summary referenced in common cached storage.

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



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 177 for a description of these parameters.

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:

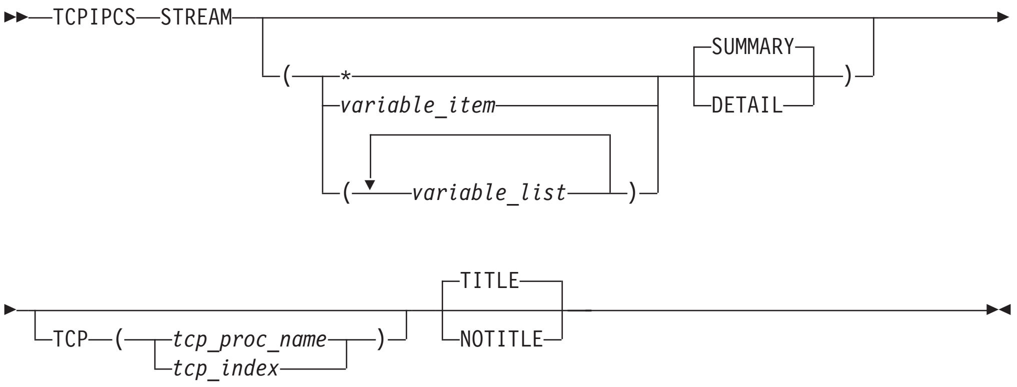
```
TCPIPCS STORAGE
Dataset: IPCS.A594094.DUMPM
Title:   TCPSVT   V2R10: Job(TCPSVT  ) EZBITSTO(HTCP50A 99.281)+
         00077A S4C5/74BE2500 SRB P=0051,S=0051,H=0051
...
TCPIP Storage Analysis

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 008AC010 (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
Addr       Size Key Sp Cblk Time Stamp         Index

Common non-fetch protected storage
12E6DCB0    304  6 241 CFGM B30A8EDF19BD18C3  10
12774310   3056  6 241 CFGM B30A8E3DDBBB1943  10 Index was 29
The control block at 0E289010 (prev: 12B57650) 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: 12B57730) 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
...
Storage Summary Statistics
                                       All               Cache
Type                           Count      Size     Count      Size
Common Non-fetch protected     21460   177489K         2      3392
Common Fetch protected           369     68488       141     36936
Common persistent                  3       192         3       192
Common SCB pool                   80     21128        32      8448
Private Non-fetch protected      492    395848       156     65192
Total                          22571   178149K       334    114160

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



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:

SKCB	Stream context control block address.
SKQI	Stream queue initialization control block address.
SKQP	Stream queue pair control block address.
SKQU	Stream Queue control block address.
SKSC	Stream access control control block address.
SKSH	Stream header control block address.

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 177 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:

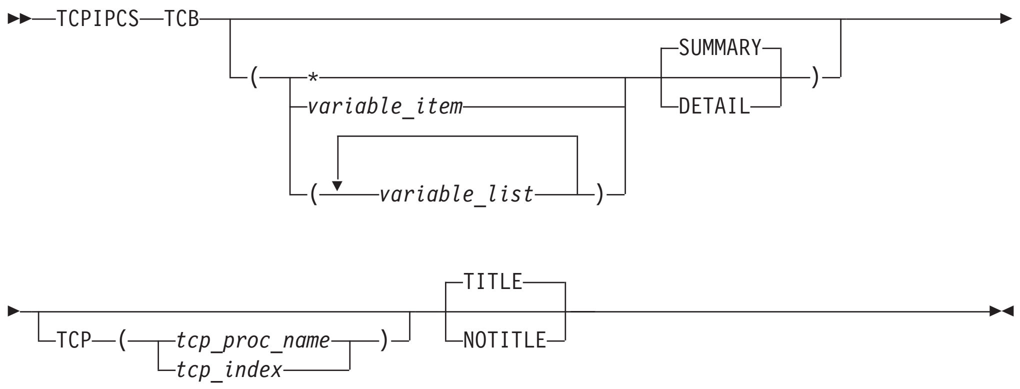
```
TCPIPCS STREAM
Dataset: IPCS.A594094.DUMPM
Title:   TCPSVT   V2R10: Job(TCPSVT  ) EZBITSTO(HTCP50A 99.281)+
         00077A S4C5/74BE2500 SRB P=0051,S=0051,H=0051

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@
7F77E6C8 7F77E7C8 00000007 IP/NAM 00000000 00000000 00000000 00000000
7F70F088 7F61A088 00000006 RAW    00000000 00000000 00000000 00000000
7F70F148 7F61A608 00000005 IP/NAM 00000000 00000000 00000000 00000000
7F70F8C8 7F70F348 00000004 UDP    00000000 00000000 00000000 00000000
7F70F988 7F70FA48 00000003 IP/NAM 00000000 00000000 00000000 00000000
7F78B008 7F7580E8 00000002 TCP    00000000 00000000 00000000 00000000
7F78BCC8 7F78B748 00000001 IP/NAM 00000000 00000000 00000000 00000000

7 Stream(s) found
7 Stream(s) formatted

Analysis of Tcp/Ip for TCPSVT completed
```

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



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:

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 TCB(s).

TCP, TITLE, NOTITLE
See "Parameters" on page 177 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 13C9F0C8 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  MTCPMTX............ 00000000  00000000  00000000  D7D60601
 +0028  MTCPAQMX........... 00000000  00000000  00000000  D7D60604
 +0038  MTCB_LIST_LOCK..... 00000000  00000000  00000000  D7D60604
 +0048  M_PORT_CEILING..... 00000FFF
 +004C  M_TPI_SEQ#......... 00000008
 +0050  M_PORT_ARRAY....... 7F711FC8
 +0054  M_LAST_PORT_NUM.... 0000040C
 ...
```

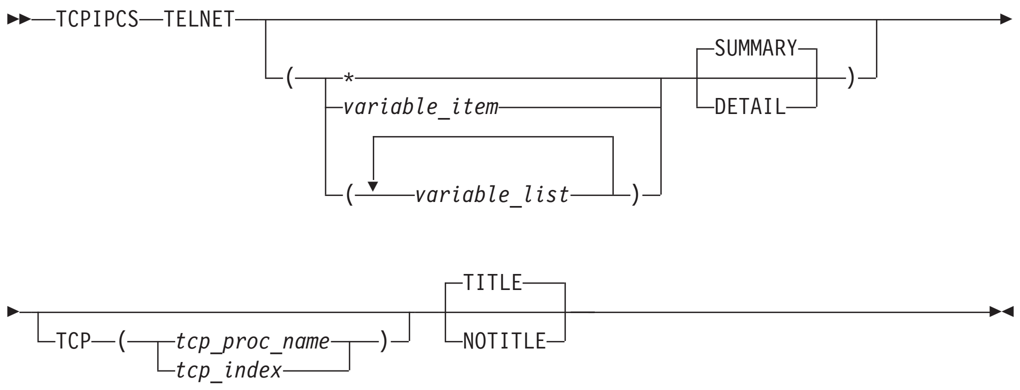
```
TCB      ResrcID  ResrcNm  TcpState    TpiState Local IPAddr/Port      Remote IPAddr/Port    LuName   ApplName UserID
7F603108 00000002 TCPSVT   Closed      WLOUNBND 0.0.0.0..0             0.0.0.0..0
7F605D08 00000017 FTPUNIX1 Listening   WLOIDLE  0.0.0.0..21            0.0.0.0..0
7F605108 00000013 TCPSVT   Listening   WLOIDLE  0.0.0.0..625           0.0.0.0..0
7F603508 0000000A TCPSVT   Listening   WLOIDLE  0.0.0.0..1025          0.0.0.0..0
7F604508 000000EA TCPSVT   Established WLOXFER  197.66.103.1..23       197.11.108.1..1032
...
7F607108 0000003E TCPSVT   Established WLOXFER  127.0.0.1..1029        127.0.0.1..1028
7F60A508 000000E8 TCPSVT   Listening   WLOIDLE  0.0.0.0..623           0.0.0.0..0
25 TCB(s) FOUND
25 TCB(s) FORMATTED

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS 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



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.

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 right with zeros.

In addition to the variable parameters described above, you can specify the following keyword parameters:

SUMMARY
Displays the address of the control blocks. SUMMARY is the default.

DETAIL
Displays the contents of the control blocks.

TCP, TITLE, NOTITLE
See "Parameters" on page 177 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 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 Tsdb     Tsdx     Asid TraceOpts Status

13391C00  1 TCPSVT    V2R10   1323B000 1323B0C8 07DE 04041405  Active
13391C80  2 TCPSVT2   V2R10   00000000 00000000 07E8 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
7F59D8A0   623 7F5A6068 CURR 00000000 00000000 00000000 00000000
7F59D4E0   625 7F59D620 CURR 00000000 00000000 00000000 00000000

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS TIMER

Use this subcommand to display the timer control blocks.

Syntax

```
                      ┌─(SUMMARY)─┐
►►──TCPIPCS──TIMER────┼───────────┼──┬───────────────────────────────────┬──────────────────►◄
                      └─(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.

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 177 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 086D8F80

ItTmr        Pass     Slot      Delta        Max   PopCount Array@
086D8F80       64       62        100      12800       8253 086D9000

  Global TQE Queue for Slot 63:

    Tqe      Tid      Ecb      Mod      Parm           Msec TqeFlag  TidFLag

    08EDDD58 08EDDD44 00000000 EZBIFIU2 08EDDD40        100 00       20

  1 TQE(s) for slot 63 with 0 msec timer offset

ItTmr        Pass     Slot      Delta        Max   PopCount Array@
086D8FA0        6       58       1000     128000        825 086D9400

ItTmr        Pass     Slot      Delta        Max   PopCount Array@
086D8FC0        0       83      10000    1280000         82 086D9800

  Global TQE Queue for Slot 122:
      Tqe      Tid      Ecb      Mod      Parm           Msec TqeFlag  TidFLag

      086C9020 7F4CEBD0 7F4CEBCC 00000000 00000000    1200000 40       20

    1 TQE(s) for slot 122 with 128000 msec timer offset

  ItTmr        Pass     Slot      Delta        Max   PopCount Array@
  086D8FE0        0        9     100000 4294967295          8 086D9C00

 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

```
                        ┌─(SUMMARY)─┐
►►──TCPIPCS──TRACE──┼───────────┼──┬──────────────────────────────────┬──────────►
                        └─(DETAIL)──┘  └─TCP──(──┬─tcp_proc_name─┬──)─┘
                                                 └─tcp_index─────┘

   ┌─TITLE───┐
►──┼─────────┼───────────────────────────────────────────────────────────────►◄
   └─NOTITLE─┘
```

Parameters

SUMMARY
Displays a summary of the CTrace status.

DETAIL
In addition to the SUMMARY information, lists the individual trace buffer entries.

TCP, TITLE, NOTITLE
See "Parameters" on page 177 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

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

09C44610  1 TCPCS     V1R5    093C1000 093C10C8 0029 9FFF7E7F  Active
09C44690  2 TCPCS2    V1R5    00000000 00000000 002A 00000000  Down Stopping

   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 SYSTCPIS Trace: CTIIDS00

Trace Control Area
 TCA: 092BD410
 +0000  TCAACRONYM.......... TCA
 +0006  TCAVERSION.......... 0006
 +0008  TCASIZE............. 0000CBD0
 +000C  TCAFTBE............. 092BD7E0
 +0010  TCACURTBE........... 092BE5C8
 +0014  TCACURENT........... 0059C4C0
 +0018  TCATABSZ............ 01000000
 +001C  TCANUMBF............ 00000100
 +0020  TCABUFSZ............ 00010000
 +0024  TCAMXDAT............ 00003800
 +0028  TCAALET............. 01FF000C
 +002C  TCARCNT............. 00004103
 +0030  TCAECNT............. 00004103
 +0034  TCALCNT............. 00000000
 +0038  TCALTOD............. 00000000  00000000
 +0040  TCACOMP............. 00000000
 +0044  TCAFLAG............. 03200A80
 +0048  TCAXWRTSEQ.......... 00000059
 +004C  TCACTSSWTKN......... 00000000  00000000
 +0054  TCAACNT............. 0000

         -- Array elements --
 +0058  TCAFILTER_ASID...... 0000
 +005A  TCAFILTER_ASID...... 0000
 +005C  TCAFILTER_ASID...... 0000
 +005E  TCAFILTER_ASID...... 0000
 +0060  TCAFILTER_ASID...... 0000
 +0062  TCAFILTER_ASID...... 0000
 +0064  TCAFILTER_ASID...... 0000
 +0066  TCAFILTER_ASID...... 0000
 +0068  TCAFILTER_ASID...... 0000
 +006A  TCAFILTER_ASID...... 0000
 +006C  TCAFILTER_ASID...... 0000
 +006E  TCAFILTER_ASID...... 0000
 +0070  TCAFILTER_ASID...... 0000
 +0072  TCAFILTER_ASID...... 0000
 +0074  TCAFILTER_ASID...... 0000
 +0076  TCAFILTER_ASID...... 0000
         -- End of array   --

 +0078  TCAUCNT............. 0000

         -- Array elements --
 +007C  TCAFILTER_USERID.... ........
 +0084  TCAFILTER_USERID.... ........
 +008C  TCAFILTER_USERID.... ........
 +0094  TCAFILTER_USERID.... ........
 +009C  TCAFILTER_USERID.... ........
 +00A4  TCAFILTER_USERID.... ........
 +00AC  TCAFILTER_USERID.... ........
 +00B4  TCAFILTER_USERID.... ........
 +00BC  TCAFILTER_USERID.... ........
 +00C4  TCAFILTER_USERID.... ........
 +00CC  TCAFILTER_USERID.... ........
 +00D4  TCAFILTER_USERID.... ........
 +00DC  TCAFILTER_USERID.... ........
```

```
+00E4  TCAFILTER_USERID.... ........
+00EC  TCAFILTER_USERID.... ........
+00F4  TCAFILTER_USERID.... ........
        -- End of array   --

+0100  TCAPCNT............. 00000000

        -- Array elements --
+0104  TCAFILTER_PORT...... 0000
+0106  TCAFILTER_PORT...... 0000
+0108  TCAFILTER_PORT...... 0000
+010A  TCAFILTER_PORT...... 0000
+010C  TCAFILTER_PORT...... 0000
+010E  TCAFILTER_PORT...... 0000
+0110  TCAFILTER_PORT...... 0000
+0112  TCAFILTER_PORT...... 0000
+0114  TCAFILTER_PORT...... 0000
+0116  TCAFILTER_PORT...... 0000
+0118  TCAFILTER_PORT...... 0000
+011A  TCAFILTER_PORT...... 0000
+011C  TCAFILTER_PORT...... 0000
+011E  TCAFILTER_PORT...... 0000
+0120  TCAFILTER_PORT...... 0000
+0122  TCAFILTER_PORT...... 0000
        -- End of array   --

+0124  TCAICNT............. 00000000

        -- Array elements --
+0128  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+0138  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+0148  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+0158  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+0168  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+0178  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+0188  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+0198  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+01A8  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+01B8  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+01C8  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+01D8  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+01E8  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+01F8  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+0208  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+0218  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+0228  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+0238  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+0248  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+0258  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+0268  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+0278  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+0288  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+0298  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+02A8  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+02B8  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+02C8  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+02D8  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+02E8  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+02F8  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
+0308  TCAFILTER_IPADDRESS. 00000000  00000000  00000000  00000000
+0318  TCAFILTER_IPSUBMASK. 00000000  00000000  00000000  00000000
        -- End of array   --

+0330  TCAWRAPTIME......... B66479D0  DE1B3402
+0338  TCAWRAPCOUNT........ 00000001
+033C  TCAXWTRCNT.......... 00000000
+0340  TCACURCUR........... 092BFFE0
```

```
 +0344  TCANXTCUR........... 01004E80
 +0348  TCADATSZ............ 02000000
 +034C  TCADATBF............ 00000200
 +0350  TCAWCONT............ 00000068
 +0354  TCATRCNT............ 00000068
 +0358  TCALSCNT............ 00000000
 +035C  TCASEQXWRT.......... 00000001
 +0360  TCAPTSSWTKN......... 02895060  00000001
 +0368  TCAISTBE............ 092C4FE0
 +036C  TCAISNRTBE.......... 00000200
 +0370  TCAISBUFSZ.......... 00010000
 +0374  TCAISTBLSZ.......... 02000000
 +0378  TCAISTRCNT.......... 00000000
 +037C  TCAISWRCNT.......... 00000000
 +0380  TCAISLSCNT.......... 00000000
 +0384  TCAISRQCNT.......... 00000000
 +0388  TCAISXWSEQ.......... 00000001
 +0390  TCAISCDS............ 092C4FE0  03001000
 +0390  TCAISCDTBE.......... 092C4FE0
 +0394  TCAISCDBUF.......... 03001000
 +0398  TCAISXWTKN.......... 00000000  00000000
 +03A0  TCAISWRTIM.......... 00000000  00000000
 +03A8  TCAISLSTIM.......... 00000000  00000000
 +03B0  TCADUMPSZ........... 00000000
 +03B4  TCADUMPDS........... 00000000
 +03B8  TCADUMPOF........... 00000000
 +03C0  TCADUMPTOD.......... 00000000  00000000

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 SYSTCPIS
  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

Tseb_Trace_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
         VtamData 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 SYSTCPIS Trace Buffer Elements were found
   0 SYSTCPIS 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──────────────────────────────────────────────────────────────────►

►─TREE──┬─────────────────────────────────────────────────────────────────┬──►
        │      ┌─────────────────┐                                        │
        │      ▼  ┌─ALL──────┐   │           ┌─SUMMARY─┐  ┌─BOTH───┐     │
        └─(───────┼─ARP──────┼───┴─┬────────┬┼─────────┼──┼────────┼──)──┘
                  ├─FIREWALL─┤     └─HEADER─┘└─DETAIL──┘  ├─ACTIVE─┤
                  ├─IPSEC────┤                            └─DELETE─┘
                  ├─IQDIO────┤
                  ├─ND───────┤
                  ├─NETACC───┤
                  ├─NETACCV4─┤
                  ├─NETACCV6─┤
                  ├─POLICY───┤
                  ├─ROUTE────┤
                  ├─ROUTEV4──┤
                  ├─ROUTEV6──┤
                  ├─TCP──────┤
                  ├─TELNET───┤
                  └─XCF──────┘

                                       ┌─TITLE───┐
►──┬────────────────────────────────┬──┼─────────┼───────────────────────────►◄
   └─TCP──(──┬─tcp_proc_name─┬──)───┘  └─NOTITLE─┘
             └─tcp_index─────┘
```

Parameters

ALL
Display structure of all TCP/IP trees. ALL is the default.

ARP
Display only structure of ARP trees.

FIREWALL
Display only structure of Firewall trees.

IPSEC
Display only structure of IP security trees.

IQDIO
Display only structure of iQDIO 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.

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 177 for a description of these parameters.

Restrictions:

- If you specify multiple keywords from the set (ALL,ARP,FIREWALL,IPSEC,IQDIO,ND,NETACC,NETACCV4,NETACCV6, POLICY,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 TCPIPCS TREE subcommand

The following is sample output of the TCPIPCS TREE subcommand:

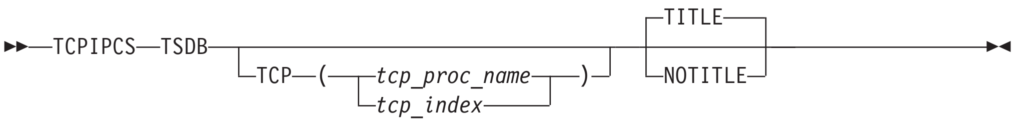
```
TCPIP Tree Analysis

IPv4 NetAccess Search Tree

Node@     Bit Parent   LChild   RChild   Key      Element
2B42D678  255 00000000 7F042D90 7F04D6F0
7F04D230    2 7F0D1010 7F0D1010 7F04D230 7F04D490 2B1F4898
7F0D1010    3 7F04D570 7F04D6F0 7F04D230 7F04D190 2B1F48F8
7F04D570    4 7F04CD90 7F0D1010 7F04D570 7F04D610 2B1F4838
7F04CC50    9 7F04C950 7F04C950 7F04CC50 7F04CCF0 2B1F47D8
7F04C810    9 7F04C510 7F04C510 7F04C810 7F04C8B0 2B2064B8
7F04C250    9 7F04C510 7F0D4010 7F04C250 7F04C470 2B206578
7F04C510   10 7F0D4010 7F04C250 7F04C810 7F04C770 2B206518
7F04BD90    9 7F04BB10 7F04BB10 7F04BD90 7F04BFB0 2B206638
7F04B850    9 7F04BB10 7F04B5D0 7F04B850 7F04BA70 2B2066F8
7F04BB10   10 7F0D4010 7F04B850 7F04BD90 7F04BCF0 2B206698
7F042D90   32 2B42D678 2B42D678 7F042D90 7F042CF0 2B1F4658

11 elements in IPv4 NetAccess Search Tree

IPv4 NetAccess Update Tree

Node@     Bit Parent   LChild   RChild   Key      Element
2B21E818  255 00000000 7F042E10 7F04D670
7F04D2B0    2 7F04D030 7F04D030 7F04D2B0 7F04D430 2B1F4898
7F04D030    3 7F04D4F0 7F04D670 7F04D2B0 7F04D130 2B1F48F8
7F04D4F0    4 7F04CE10 7F04D030 7F04D4F0 7F04D5B0 2B1F4838
7F04CE90    9 7F04C9D0 7F04C9D0 7F04CE90 7F04CC90 2B1F47D8
7F04CA50    9 7F04C590 7F04C590 7F04CA50 7F04C850 2B2064B8
7F04C2D0    9 7F04C590 7F04C050 7F04C2D0 7F04C410 2B206578
7F04C590   10 7F04C050 7F04C2D0 7F04CA50 7F04C710 2B206518
7F04BE10    9 7F04BB90 7F04BB90 7F04BE10 7F04BF50 2B206638
7F04B8D0    9 7F04BB90 7F04B650 7F04B8D0 7F04BA10 2B2066F8
7F04BB90   10 7F04C050 7F04B8D0 7F04BE10 7F04BC90 2B206698
7F042E10   32 2B21E818 2B21E818 7F042E10 7F042C90 2B1F4658

11 elements in IPv4 NetAccess Update Tree

IPv6 NetAccess Search Tree

Node@     Bit Parent   LChild   RChild   Key      Element
2B2180B8  255 00000000 7F04D830 7F0A0010
7F085010    2 7F083010 7F083010 7F085010 7F050B70 2B1F1658
7F079350    1 7F080010 7F0A0010 7F079350 7F050930 2B1F1598
7F080010    2 7F083010 7F079350 7F080010 7F0509F0 2B1F1538
7F083010    3 7F0C4010 7F080010 7F085010 7F050AB0 2B1F16B8
7F0C4010    4 7F0506D0 7F083010 7F0C4010 7F050C30 2B1F15F8
7F050510    2 7F0506D0 7F0506D0 7F050510 7F050630 2B1F1478
7F0506D0  114 7F0A3010 7F050510 7F0C4010 7F050470 2B1F14D8
```

```
7F0A3010  115 7F050290 7F0506D0 7F0A3010 7F050CF0 2B1F1418
7F0503D0   65 7F0664D0 7F0664D0 7F0503D0 7F050170 2B1F1898
7F0661D0   65 7F0664D0 7F0C9010 7F0661D0 7F0663B0 2B1F1958
7F0664D0   66 7F0C9010 7F0661D0 7F0503D0 7F0666B0 2B1F18F8
7F04FDD0   65 7F04FB10 7F04FB10 7F04FDD0 7F04FFB0 2B1F40B8
7F04F790   65 7F04FB10 7F04F510 7F04F790 7F04F9F0 2B1F4178
7F04FB10   66 7F0C9010 7F04F790 7F04FDD0 7F04FCB0 2B1F4118
7F04D830  128 2B2180B8 2B2180B8 7F04D830 7F04D790 2B1F12F8

15 elements in IPv6 NetAccess Search Tree

IPv6 NetAccess Update Tree

Node@     Bit Parent   LChild   RChild   Key      Element
2B42D6D8  255 00000000 7F04D7F0 7F0B2010
7F0685D0    2 7F08D010 7F08D010 7F0685D0 7F050B10 2B1F1658
7F079250    1 7F079310 7F0B2010 7F079250 7F0508D0 2B1F1598
7F079310    2 7F08D010 7F079250 7F079310 7F050990 2B1F1538
7F08D010    3 7F0C2010 7F079310 7F0685D0 7F050A50 2B1F16B8
7F0C2010    4 7F050690 7F08D010 7F0C2010 7F050BD0 2B1F15F8
7F050590    2 7F050690 7F050690 7F050590 7F0505D0 2B1F1478
7F050690  114 7F0A8010 7F050590 7F0C2010 7F050410 2B1F14D8
7F0A8010  115 7F050310 7F050690 7F0A8010 7F050C90 2B1F1418
7F050010   65 7F066550 7F066550 7F050010 7F050110 2B1F1898
7F066250   65 7F066550 7F0C7010 7F066250 7F066350 2B1F1958
7F066550   66 7F0C7010 7F066250 7F050010 7F066650 2B1F18F8
7F04FE50   65 7F04FB90 7F04FB90 7F04FE50 7F04FF50 2B1F40B8
7F04F810   65 7F04FB90 7F04F590 7F04F810 7F04F990 2B1F4178
7F04FB90   66 7F0C7010 7F04F810 7F04FE50 7F04FC50 2B1F4118
7F04D7F0  128 2B42D6D8 2B42D6D8 7F04D7F0 7F04D730 2B1F12F8

15 elements in IPv6 NetAccess Update Tree
```

TCPIPCS TSDB

Use this subcommand to display the TSDB server data block.

Syntax



Parameters

TCP, TITLE, NOTITLE
See “Parameters” on page 177 for a description of these parameters.

Sample output of the TCPIPCS TSDB subcommand

The following is sample output of the TCPIPCS TSDB subcommand:

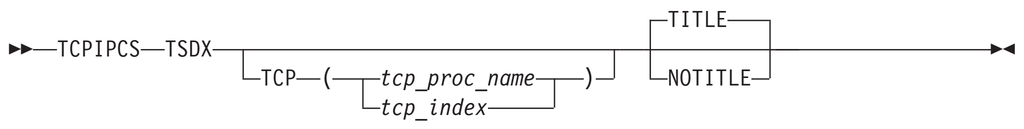
```
TCPIPCS TSDB
Dataset: IPCS.MV21381.DUMPA
Title:   SLIP DUMP ID=TC


The address of the TSAB is: 13391BC0

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

13391C00  1 TCPSVT    V2R10   1323B000 1323B0C8 07DE 04041405  Active
13391C80  2 TCPSVT2   V2R10   00000000 00000000 07E8 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

TSDB control block summary

 TSDB: 1323B000
 +0000  TSDB_ACRONYM............ TSDB
 +0004  TSDB_LENGTH............. 00C8
 +0006  TSDB_VERSION............ 0003
 +0008  TSDB_STATE.............. 0015
 +000A  TSDB_ASID............... 07DE

         -- Array elements --
 +0010  TSDB_MT................. 11A7E870
 +0014  TSDB_MT................. 962F5E00
 ....
 +0060  TSDB_CTRACE_PARMLIB_NAME. CTIEZB02
 +006C  TSDB_SMCA............... 00000000
 +0070  TSDB_TSRMT.............. 00000000
 +0074  TSDB_FLAGS.............. 00000000
 +0078  TSDB_CONFIG_PORT........ 00000401
 +007C  TSDB_OSASF_PORT......... FFFFFFFF
 +0080  TSDB_EZBITMSN@.......... 91A8BF90
 +0084  TSDB_TERMINATING_ECB.... 807EC758
 +0088  TSDB_DUAF............... 00000000
 +008C  TSDB_TSCA............... 13236A58
 +0090  TSDB_SOCIFPTR........... 91BC3E78
 +0094  TSDB_SOMIFPTR........... 91BCA050
 +0098  TSDB_RXGLUPTR........... 91BF6308
 +009C  TSDB_FFSTADDR........... 80B46E18
 +00A0  TSDB_FFST_PHMSGTIME..... 00000000  00000000
 +00A8  TSDB_LEPARMS............ 14B01BBA
 +00AC  TSDB_OE_AS_STOKEN....... 00000038  00000001
 +00B4  TSDB_SOMT2.............. 91C3FE60

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS TSDX

Use this subcommand to display the TSDX server data extension.

Syntax



Parameters

TCP, TITLE, NOTITLE
See "Parameters" on page 177 for a description of these parameters.

Sample output of the TCPIPCS TSDX subcommand

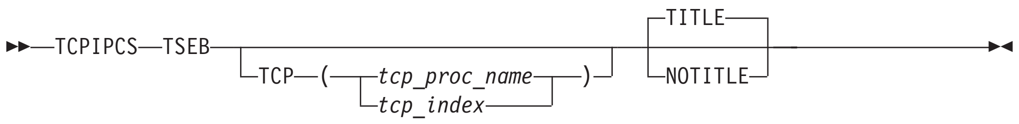
The following is sample output of the TCPIPCS TSDX subcommand:

```
TCPIPCS TSDX
Dataset: IPCS.MV21381.DUMPA
Title:   SLIP DUMP ID=TC


The address of the TSAB is: 13391BC0

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

13391C00  1 TCPSVT    V2R10   1323B000 1323B0C8 07DE 04041405  Active
13391C80  2 TCPSVT2   V2R10   00000000 00000000 07E8 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

TSDX control block summary

 TSDX: 1323B0C8
 +0000  TSDX_ACRONYM............... TSDX
 +0004  TSDX_LENGTH................ 0300
 +0006  TSDX_VERSION............... 0003
 +0008  TSDX_FLAGS................. 60000001
 +000C  TSDX_ASCB.................. 00F7C280
 +0010  TSDX_PROCNAME.............. TCPSVT
 +0018  TSDX_CART.................. 00000000  00000000
 +0020  TSDX_CONSID................ 00000001
 +0024  TSDX_TCB................... 007EC9A8
 +0028  TSDX_TCB_TOKEN............. 00001F78  00000008  00000003  007EC9A8
 +0038  TSDX_TCPIP_DS_ALET......... 01FF0011
 +003C  TSDX_TCPIP_DS_ADDR......... 00001000
 +0040  TSDX_TCPIP_DS_END.......... 19001000
 +0044  TSDX_ET_TOKEN.............. 7FFD9D10
...
 +026C  TSDX_CSMSTATAREA........... 141C7A88
 +0270  TSDX_CSMDUMPINFO........... 141C7A90
 +0288  TSDX_AUTOLOG_TASK_ECB...... 807EC758
 +028C  TSDX_AUTOLOG_CB............ 1333C0A8
 +0290  TSDX_SASTRT_ECB............ 807EC758
 +0294  TSDX_XFCVT................. 13096410
 +0298  TSDX_XCFLOCK............... 00000000  00000000  00000000  D7D60901

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS TSEB

Use this subcommand to display the TSEB server anchor block.

Syntax



Parameters

TCP, TITLE, NOTITLE
See "Parameters" on page 177 for a description of these parameters.

Sample output of the TCPIPCS TSEB subcommand

The following is sample output of the TCPIPCS TSEB subcommand:

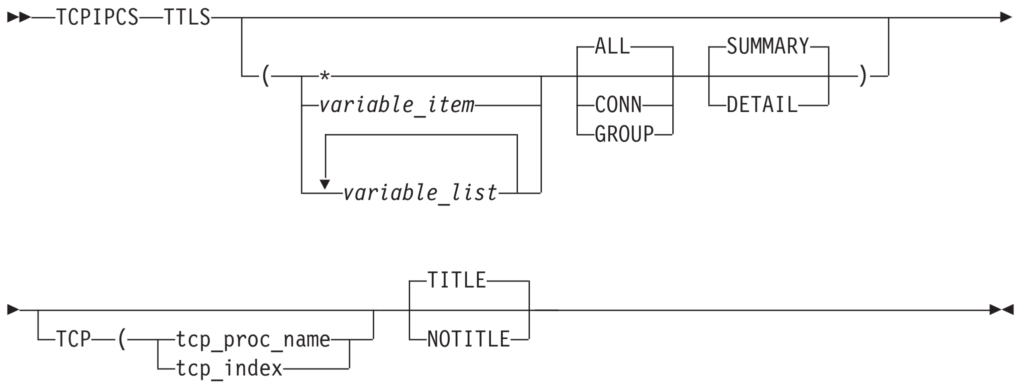
```
TCPIPCS TSEB
Dataset: IPCS.MV21381.DUMPA
Title:   SLIP DUMP ID=TC


The address of the TSAB is: 13391BC0

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

13391C00  1 TCPSVT    V2R10   1323B000 1323B0C8 07DE 04041405  Active
13391C80  2 TCPSVT2   V2R10   00000000 00000000 07E8 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
 +0008  TSEB_STATUS........... 82
 +000C  TSEB_REQUESTORS....... 00000000
 +0010  TSEB_TCPIP_NAME....... TCPSVT
 +0018  TSEB_SI............... 01
 +0019  TSEB_IID.............. 04
 +001A  TSEB_TCPIP_VERSION.... 0510
 +001C  TSEB_TSDB............. 1323B000
 +0020  TSEB_LX............... 00002E00
 +0024  TSEB_TCA.............. 11469E50
 +0028  TSEB_TRACE_OPTS....... 04041405
 +002C  TSEB_TRACE_OPT2....... 00000000
 +0034  TSEB_SCHEDULED_EVENTS. 00000000
 +0038  TSEB_ASID............. 07DE
 +003C  TSEB_LPA_SADDR........ 11A719E0
 +0040  TSEB_LPA_EADDR........ 11C62FFF
 +0044  TSEB_QDIO_BGRP_Q@..... 1320A648
 +0048  TSEB_EZBITDCR......... 9320ACF8
 +004C  TSEB_ITCVT............ 1323B3C8
 +0050  TSEB_BGRP_Q@.......... 1320A608
 +0054  TSEB_DUAF............. 130A4010
 +0059  TSEB_TOKENID.......... 000015
 +005C  TSEB_TCMTPTR.......... 132072F0
 +0060  TSEB_EZBITCOM_LEN..... 00007D28
 +0064  TSEB_CS390_VERSION.... 020A
 +0068  TSEB_CSMFREE.......... 1320A548
 +006C  TSEB_TCPIP_STOKEN..... 00001F78  00000008
 +0074  TSEB_CSMPACK.......... 1320A588
```

```
  +0078  TSEB_SOCA............. 140BAEB8
  +007C  TSEB_CSMPACKQDIO...... 1320A5C8

 Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS TTLS

Display information about Application Transparent Transport Layer Security (AT-TLS), AT-TLS groups, and AT-TLS connections.

Syntax



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:

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 177 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:

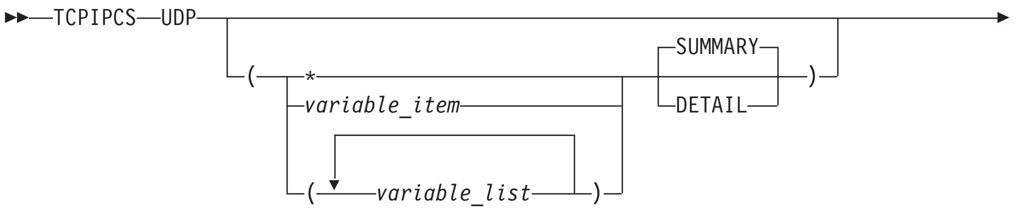
```
TCPIPCS TTLS
Dataset: IPCS.MV31738.DUMPA
...

===============================================================================

Analysis of Tcp/Ip for TCPSVT.  Index: 1


TCP/IP Analysis
TCPIP Main TTLS Control Block (EZBZTTLS)
 EZBZTTLS: 7F722150
 +0000  TTLS_ACRONYM.......... EZBZTTLS
 +0008  TTLS_PART_LOCK........ 00000000  00000000  00000000  D225030A
 +0008  LOCK_CDS.............. 00000000  00000000
 +0008  LOCK_SUSPENDED_GLOBAL. 00000000
 +000C  LOCK_HOLDER........... 00000000
 +0010  LOCK_SUSPENDED_LOCAL.. 00000000
 +0014  LOCK_INFO............. D225030A
 +0014  LOCK_INIT............. D225
 +0014  LOCK_INIT1............ D2
 +0015  LOCK_INIT2............ 25
 +0016  LOCK_CLASS............ 03
 +0017  LOCK_LEVEL............ 0A
 +0018  TTLS_FLAG1............ E8000000
 +001C  TTLS_PIPIADD_CNT...... 00
 +001D  TTLS_GRPCNT........... 13
 +001E  TTLS_TCB_CMPC_OFF..... 11
 +001F  TTLS_ABEND_COUNT...... 00
 +0020  TTLS_1STABEND......... 00000000
 +0024  TTLS_TCBPTR........... 006EB5C8
 +0028  TTLS_RESMGR_TOKEN..... 000077FE  00000000
 +0030  TTLS_INBNDPART@....... 7E8142B0
 +0034  TTLS_OUTBNDPART@...... 7E822170
 +0038  TTLS_PCT_STATE........ 00000003
 +003C  TTLS_PCT_INSTANCEID... 41E7D968
 +0040  TTLS_WORKQ............ 00000000  00000000
 +0040  ITLFPUBLIC............ 00000000
 +0040  ITLFHEAD.............. 00000000
 +0044  ITLFPRIVATE........... 00000000
 +0044  ITLFTAIL.............. 00000000
 +0048  TTLS_TERM_ECB......... 806EB520
 +004C  TTLS_INIT_ECB......... 00000000
 +0050  TTLS_EOT_ECB.......... 006FF278
 +0054  TTLS_WORKQ_ECB........ 806EB520
 +0058  TTLS_CLEANUP_TIMER.... 00000000  00000000  00000000  00000000
 +0058  TID_EYE............... 00000000
 +005C  TID_MSEC.............. 00000000
 +0060  TID_FLAGS............. 00000000
 +0064  TID_TQE............... 00000000
 +0068  TTLS_MODLIST.......... 7F722208
 +006C  TTLS_GROUPNUM......... 00000024
 +0070  TTLS_TGRP_HT_TOKEN.... 7F7858F0  00000004
 +0078  TTLS_CLEANUP_ECB...... 00000000
 +007C  TTLS_MAX_SRBS......... 00000005
 +0080  TTLS_CURR_SRBS........ 00000000
 +0084  TTLS_WE_CNT........... 00000000
 +0088  TTLS_PIPI_ECB......... 806EB520
 +008C  TTLS_PIPI_POOLPTR..... 7F5FE9B8
 +0090  TTLS_PIPI_SUSPQ....... 00000000  00000000
 +0090  ITLFPUBLIC............ 00000000
 +0090  ITLFHEAD.............. 00000000
 +0094  ITLFPRIVATE........... 00000000
 +0094  ITLFTAIL.............. 00000000
 +0098  TTLS_ENVNUM........... 00000004
 +009C  TTLS_GLBLTHD.......... 0000004C
 +00A0  TTLS_SECOND_HT_TOKEN.. 7F785750  00000005

 0 TLMST Work Requests Formatted

TTLS Secondary Map hashtable entries
Pri_TCB@ PID      Local_IP..Remote_IP

TCB@     ConnID   TLSX@    Proto Cipher Jobname  UserID   Cert@    CertId
7D83B110 0000016D 7D9ED0B0 ..... ..     WEBSTCP  SVTWSRV  00000000 ........

  LocalSocket:  197.11.203.11..1026
  RemoteSocket: 198.11.22.103..1033
  Tcb_tcp_state: Established
  Tcb_TtlsFlags:Ttls_Gate
  TLSX_Flags1:  LookUp_Done
  TTLSRule(7E828110):       prTTLS-DEFAULT-RULE-OFF (Stale) {
  TTLSGroupAction(7E715190):  paTTLS-GLOBAL-OFF (Stale) {

7D5ED910 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(7D74DB10):  Webs_group_action_80
  TTLSEnvironmentAction(7D750590):  Webs_Environment_80

...
```

```
135 TCBs Found
 22 TCBs Formatted


TTLS Group: TNs_group_action_923
  Address Group Id    Conns    Tasks Elements Created
 7E5CB7B0       21        0        4        0 2005/01/14 14:38:32

  -----TTLS Environments----------------


  0 TTLS Environments Formatted

  -----TTLS Worker Tasks----------------


   TTLS Worker Task: 7DB06510
        Ducb FuncCode    Rcode     Busy Idle Time
    3A138000        3        0        0 2005/01/14 14:38:31

   TTLS Worker Task: 7DB06890
        Ducb FuncCode    Rcode     Busy Idle Time
    3A13E000        3        0        0 2005/01/14 14:38:31

   TTLS Worker Task: 7DB06A50
        Ducb FuncCode    Rcode     Busy Idle Time
    3A141000        3        0        0 2005/01/14 14:38:31

   TTLS Worker Task: 7DB05A90
        Ducb FuncCode    Rcode     Busy Idle Time
    3A126000        3        0        0 2005/01/14 14:38:31

 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



```
                                          ┌─TITLE───┐
►─┬──────────────────────────────────┬─┼─────────┼──────────────────────────►◄
  └─TCP──(──┬─tcp_proc_name─┬──)─┘ └─NOTITLE─┘
            └─tcp_index─────┘
```

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:

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” on page 177 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  UMUCBEYE. MUCB      USTKLNKD. 01        UDRVSTAT. 00
 +0008  UMUCB6FLG..........           00010000
 +0009  U6STKLNKD..........           01
 +000B  U6DRVSTAT..........           00
 +000C  UOPENPRT. 00000000  UFREEPRT. 041C      MCBMUTEX. 00000000
        00000000  00000000  D7D60402
 +0028  UDPCFG... 00000001  0000FFFF  0000FFFF  00000001  80000000
        00000000
 +0040  UDPCFG2.. 00000001  0000FFFF  0000FFFF  00000001  80000000
        00000000
 +0058  UDPMIB... 00000008  0000004B  00000000  0000004D
        USBCAST.. 00000000  USLPBACK. 00000000  USDN
 +0074  USRCVBUF. 0000FFFF  USSNDBUF. 0000FFFF  UFGPRC... 00
        USERIALV. 00000003  USERIAL1. 00000000  ULAS
 +008C  ULASTPRT. 0000      ULASTUCB. 00000000  USERIAL2. 00000000
        UIPWRQ@.. 7F407968  UIPRDQ@.. 7F407928  UIP6
 +00A4  UIP6RDQ@. 7F207928
 +00BC  UDMULTI_NUM........           00000000
 +00C0  UDMUX_TOKEN........           7F407B88  00000003
 +00D0  UDMULTI@. 00000000
 +00D4  UD6MULTI_NUM.......           00000000
 +00D8  UD6MULTI@..........           00000000
 +00DC  UD6MUX_TOKEN.......           7F207B88  00000006
 +00E4  UD6MUX_MULTI_TOKEN.           00000000  00000000

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
```

TCPIPCS VMCF

Use this subcommand to display information about VMCF (Virtual Machine Communication Facility) and IUCV (Inter-User Communication Vehicle) users.

Syntax

```
►►──TCPIPCS──VMCF──┬────────────────────────────────────────────────────┬──────────►
                   │                                  ┌─SUMMARY─┐       │
                   └─(──┬─*───────────────────────┬──┼─────────┼──)─┘
                        ├─variable_item───────────┤  └─DETAIL──┘
                        │    ┌──────────────────┐ │
                        │    ▼                  │ │
                        └─(──────variable_list──┴─)─┘

                                              ┌─TITLE───┐
►──┬──────────────────────────────────────┬──┼─────────┼─────────────────────────►◄
   └─TCP──(──┬─tcp_proc_name──┬──)──┘  └─NOTITLE─┘
             └─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. Specified as 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 177 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

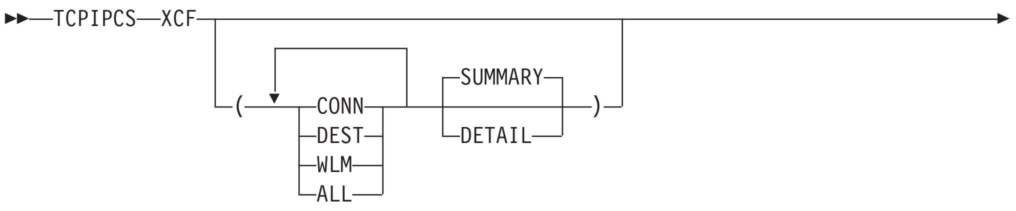
The following is sample output of the TCPIPCS VMCF subcommand:

```
TCPIPCS VMCF ((* )  SUMMARY)
Dataset: IPCS.JW11111.DUMPA
Title:   IPCS VMCF DUMP


The address of the TSAB is: 08EBC180

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

08EBC1C0  1 TCPCS     V2R10   089DC000 089DC0C8 01F7 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 TCPCS.  Index: 1

TCPIP VMCF Analysis

XINF at 09813000
  VMCF CVT            : 00A44078
  User Array          : 09813090
  Userid Count        : 1
  Userid Array        : 09817050
  Userid              : VMCF
  MSGBUILD            : 89802838
  MVPMSGS             : 8981A290
  Ecb                 : 00000000
  TNF CVT             : 00A63808
  VMCF QD             : 00000000
  VMCF QD Count       : 0
  TNF Manager Area    : 00008FE0
  SMSG Id             : 0

USER at 09813C50
  Userid              : USER18
  Asid                : 005D
  No UserData

Analysis of Tcp/Ip for TCPCS completed
```

TCPIPCS XCF

Use this subcommand to produce a cross-system coupling facility (XCF) analysis report.

Syntax



```
                                                    ┌─TITLE───┐
►──┬─────────────────────────────────────────┬──┼─────────┼──────────────────────────────────►◄
   └─TCP──(──┬─tcp_proc_name─┬──)─┘  └─NOTITLE─┘
              └─tcp_index─────┘
```

Parameters

If no parameters are specified, the dynamic VIPA hash table and partner tables are summarized

CONN
: Display only connection hash table optional information. CONN is the default.

DEST
: Display only destination hash table optional information.

WLM
: Display only workload manager optional information.

ALL
: Display all optional information.

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 177 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.
- If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one is used.

Sample output of the TCPIPCS XCF subcommand

The following is sample output of the TCPIPCS XCF subcommand:

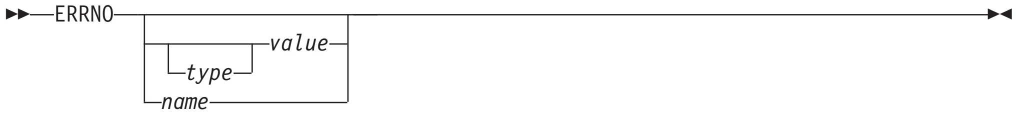
```
TCPIPCS XCF
Dataset: IPCS.MV20603.DUMPA
...
----XFCVT information----

XFCVT@        12CC7410       Member Name RUSSIATCPSVT
Local PTB     12CC752C       PTB Chain   1276A410
DVIPAHashT@   13239408       IPHashT@    12A9C010
ConnRteHashT@ 12A9B010       DPTHashT@   1277D010
WLMData@      00000000       PolicyPart@ 7F635108
================================================================================
----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
...
=========================================================
                         Next Partner Table
=========================================================
----Partner Table Control Block----
Partner Table at 1276A410
NextPtr:    12659410
MVSName:    SPAIN            CPName:        SPAIN
TCPname:    TCPSVT           IPTable:       13BA63A0
...
```

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



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.

hhhhhhhhx
: 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: EZBITSTO  ErrnoJr: 29417 JRCMNOCSM
      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.

```

    Command ===> errno 7093f98.

     ReasonCode: 74717273
       Module: EZBPFWRT  ErrnoJr: 29299 JRARPSVNOTDEFINED
       Description: The ATMARPSV name specified is not defined
```

For reason code by Errno in decimal, code the following:

```

    Command ===> errno e 129n

    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) : JRFILENOTTHERE
      Description: The requested file does not exist
```

For reason code by abend code in decimal, code the following:

```
Command ===>  errno a 9473n

Abend Reason Code: 00002501
  Module: Unknown  Reason: TcpitStorNoCSMstorage
  Description: No CSM storage available
```

For reason code by module ID in hexadecimal, code the following:

```
Command ===> errno m 74be

ModuleId: 74BE(29886) : EZBITSTO EZBTIINI
```

For reason code by module name, code the following:

```
Command ===> errno ezbifinb

ModuleId: 7418(29720) : EZBIFINB EZBTIINI
```

For reason code by ERRNO name, code the following:

```
Command ===> errno ebadf

Errno: 00000071(113) : EBADF
  Description: The file descriptor is incorrect
```

For reason code by ErrnoJr name, code the following:

```
Command ===> errno   jrmaxuids

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─┤  ├─size─┤  │             ┌─ALL──────┐     │
             ├─skmb_address─┤  └─*────┘  └─HELP──(─────┼──────────┼──)─┘
             └─*────────────┘                          ├─FUNCTION─┤
                                                       ├─OPERANDS─┤
                                                       └─SYNTAX───┘
```

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.

```
ICMPHDR 9D77428 256

   ICMPv6
    Type/Code        : ECHO Request     CheckSum: 4F51 0000
    Id               : 0028             Seq: 0
    Time             : 2002/05/23 18:43:00.332756
    Echo Data        : -8
  000000 3CED3834 000513D4 08090A0B 0C0D0E0F |<.84............|
  000010 10111213 14151617 18191A1B 1C1D1E1F |................|
  000020 20212223 24252627 28292A2B 2C2D2E2F | !"#$%&'()*+,-./|
  000030 30313233 34353637 38393A3B 3C3D3E3F |0123456789:;<=>?|
  000040 40414243 44454647 48494A4B 4C4D4E4F |@ABCDEFGHIJKLMNO|
  000050 50515253 54555657 58595A5B 5C5D5E5F |PQRSTUVWXYZ.\.^_|
  000060 60616263 64656667 68696A6B 6C6D6E6F |`abcdefghijklmno|
  000070 70717273 74757677 78797A7B 7C7D7E7F |pqrstuvwxyz{|}~.|
  000080 80818283 84858687 88898A8B 8C8D8E8F |................|
  000090 90919293 94959697 98999A9B 9C9D9E9F |................|
  0000A0 A0A1A2A3 A4A5A6A7 A8A9AAAB ACADAEAF |................|
  0000B0 B0B1B2B3 B4B5B6B7 B8B9BABB BCBDBEBF |................|
  0000C0 C0C1C2C3 C4C5C6C7 C8C9CACB CCCDCECF |................|
  0000D0 D0D1D2D3 D4D5D6D7 D8D9DADB DCDDDEDF |................|
  0000E0 E0E1E2E3 E4E5E6E7 E8E9EAEB ECEDEEEF |................|
  0000F0 F0F1F2F3 F4F5F6F7 F8F9FAFB FCFDFEFF |................|

  Protocol Header    : 8
  000000 80004F51 00280000

  Data               : 590    Data Length: 0
  000000 3CED3834 000513D4 08090A0B 0C0D0E0F |.......M........|
  000010 10111213 14151617 18191A1B 1C1D1E1F |................|
  000020 20212223 24252627 28292A2B 2C2D2E2F |................|
```

IPHDR

Use the IPHDR command to display the IP header fields.

Syntax

```
►►──IPHDR──┬─ip_header_address─┬──┬──────┬──┬──────────────────────────────┬──────►◄
           ├─rcb_address───────┤  ├─size─┤  │            ┌─ALL──────┐      │
           ├─tcb_address───────┤  └─*────┘  └─HELP──(──┼──────────┼──)─┘
           ├─ucb_address───────┤                        ├─FUNCTION─┤
           ├─skmb_address──────┤                        ├─OPERANDS─┤
           ├─skdb_address──────┤                        └─SYNTAX───┘
           └─*─────────────────┘
```

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

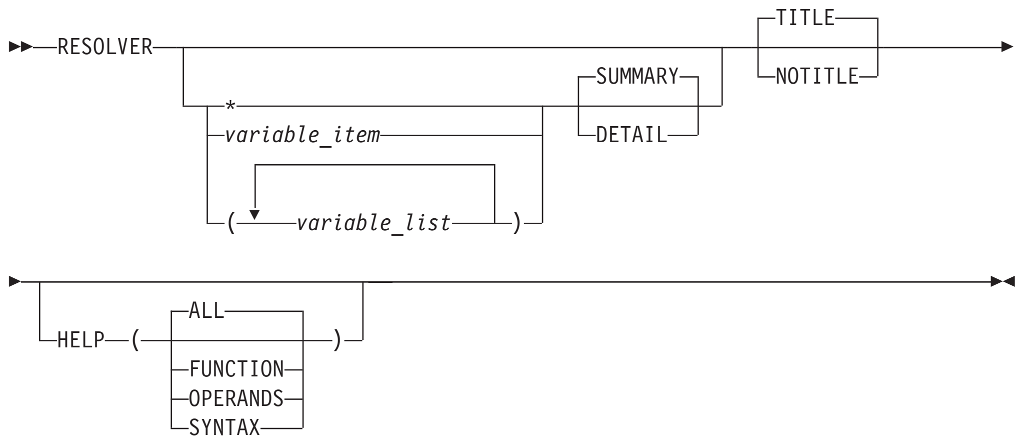
The following is a sample output of the IPHDR command:

```
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 3CED3834 000513D4 08090A0B 0C0D0E0F |<.84............|
000010 10111213 14151617 18191A1B 1C1D1E1F |................|
000020 20212223 24252627 28292A2B 2C2D2E2F | !"#$%&'()*+,-./|
000030 30313233 34353637 38393A3B 3C3D3E3F |0123456789:;<=>?|
000040 40414243 44454647 48494A4B 4C4D4E4F |@ABCDEFGHIJKLMNO|
000050 50515253 54555657 58595A5B 5C5D5E5F |PQRSTUVWXYZ.\.^_|
000060 60616263 64656667 68696A6B 6C6D6E6F |`abcdefghijklmno|
000070 70717273 74757677 78797A7B 7C7D7E7F |pqrstuvwxyz{|}~.|
000080 80818283 84858687 88898A8B 8C8D8E8F |................|
000090 90919293 94959697 98999A9B 9C9D9E9F |................|
0000A0 A0A1A2A3 A4A5A6A7 A8A9AAAB ACADAEAF |................|
0000B0 B0B1B2B3 B4B5B6B7 B8B9BABB BCBDBEBF |................|
0000C0 C0C1C2C3 C4C5C6C7 C8C9CACB CCCDCECF |................|
0000D0 D0D1D2D3 D4D5D6D7 D8D9DADB DCDDDEDF |................|
0000E0 E0E1E2E3 E4E5E6E7 E8E9EAEB ECEDEEEF |................|
0000F0 F0F1F2F3 F4F5F6F7 F8F9FAFB FCFDFEFF |................|

IP Header          : 40
000000 60000000 01083AFF 00000000 00000000  00000000 00000001 00000000 0000000
000020 00000000 00000001

Protocol Header    : 8
000000 80004F51 00280000

Data               : 720    Data Length: 256
000000 3CED3834 000513D4 08090A0B 0C0D0E0F |.......M........ <.84............|
000010 10111213 14151617 18191A1B 1C1D1E1F |................ ................|
000020 20212223 24252627 28292A2B 2C2D2E2F |................  !"#$%&'()*+,-./|
000030 30313233 34353637 38393A3B 3C3D3E3F |................ 0123456789:;<=>?|
000040 40414243 44454647 48494A4B 4C4D4E4F | .........¢.<(+| @ABCDEFGHIJKLMNO|
000050 50515253 54555657 58595A5B 5C5D5E5F |&.........!$*);^ PQRSTUVWXYZ.\.^_|
000060 60616263 64656667 68696A6B 6C6D6E6F |-/.........,%_>? `abcdefghijklmno|
000070 70717273 74757677 78797A7B 7C7D7E7F |.........`:#@'=" pqrstuvwxyz{|}~.|
000080 80818283 84858687 88898A8B 8C8D8E8F |.abcdefghi...... ................|
000090 90919293 94959697 98999A9B 9C9D9E9F |.jklmnopqr...... ................|
0000A0 A0A1A2A3 A4A5A6A7 A8A9AAAB ACADAEAF |..stuvwxyz...... ................|
0000B0 B0B1B2B3 B4B5B6B7 B8B9BABB BCBDBEBF |................ ................|
0000C0 C0C1C2C3 C4C5C6C7 C8C9CACB CCCDCECF |.ABCDEFGHI...... ................|
0000D0 D0D1D2D3 D4D5D6D7 D8D9DADB DCDDDEDF |.JKLMNOPQR...... ................|
0000E0 E0E1E2E3 E4E5E6E7 E8E9EAEB ECEDEEEF |..STUVWXYZ...... ................|
0000F0 F0F1F2F3 F4F5F6F7 F8F9FAFB FCFDFEFF |0123456789...... ................|
```

RESOLVER

Use the RESOLVER command to format and summarize resolver control blocks.

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.

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)+
         000000D6 S0C1/00000001 P=0027,S=0027,H=0027
===============================================================================

Resolver Analysis

 RCRT: 0178B058
 +0000  RCRTENT#. 0000001F  RCRTRCVT. 89E93000

        -- Array elements --
 +0008  RCRTENTS. 0178B128  RCRTENTS. 0178B1E0  RCRTENTS. 0178B298
 +0014  RCRTENTS. 0178B350  RCRTENTS. 0178B408  RCRTENTS. 0178B4C0
 +0020  RCRTENTS. 0178B578  RCRTENTS. 0178B630  RCRTENTS. 0178B6E8
 +002C  RCRTENTS. 0178B7A0  RCRTENTS. 0178B858  RCRTENTS. 0178B910
 +0038  RCRTENTS. 0178B9C8  RCRTENTS. 0178BA16  RCRTENTS. 0178BA64
 +0044  RCRTENTS. 0178BB1C  RCRTENTS. 0178BBD4  RCRTENTS. 0178BC8C
 +0050  RCRTENTS. 0178BD44  RCRTENTS. 0178BDFC  RCRTENTS. 0178BEB4
 +005C  RCRTENTS. 0178BF02  RCRTENTS. 0128FC00  RCRTENTS. 0128FC00
 +0068  RCRTENTS. 0128FC00  RCRTENTS. 0128FC00  RCRTENTS. 0128FC00
 +0074  RCRTENTS. 0128FC00  RCRTENTS. 0128FC00  RCRTENTS. 0178BF50
         -- End of array   --


RCVT: 89E93000
+0000  RCVTID... RCVT      RCVTASCB. 00F89D00  RCVTETKN. 7FFCA358
+000C  RCVTLX... 00002B00  RCVTREFR. 00000004  RCVTTCA.. 7F6C3C68
+0018  RCVTTOPT. FBFFFFFF  RCVTRSMT. 093F4000  RCVTDEFA. 00000001
+0024  RCVTDEFI. 00000002  RCVTGBLA. 00000003  RCVTGBLI. 00000004

        -- Array elements --
+0030  RCVTCFG.. ............................................................
+006C            ............................................................
+00A8            ............................................................
+00E4            ............................................................
+0120            ................
+0130  RCVTCFG.. ............................................................
+016C            ............................................................
+01A8            ............................................................
+01E4            ............................................................
+0220            ................
+0230  RCVTCFG.. SYS1.TCPPARMS(TCPDATA)......................................
+026C            ............................................................
+02A8            ............................................................
+02E4            ............................................................
+0320            ................
+0330  RCVTCFG.. ............................................................
+036C            ............................................................
+03A8            ............................................................
+03E4            ............................................................
+0420            ................
+0430  RCVTCFG.. ............................................................
+046C            ............................................................
+04A8            ............................................................
+04E4            ............................................................
+0520            ................
+0530  RCVTCFG.. ............................................................
+056C            ............................................................
+05A8            ............................................................
+05E4            ............................................................
+0620            ................
+0630  RCVTCFG.. ............................................................
+066C            ............................................................
+06A8            ............................................................
+06E4            ............................................................
+0720            ................
+0730  RCVTCFG.. ............................................................
+076C            ............................................................
+07A8            ............................................................
+07E4            ............................................................
+0820            ................
        -- End of array   --
+0830                      RCVTDEFD. 00000000  RCVTGBLD. 00000000
+0838  RCVTDEFLHOSTD.                00000000
+083C  RCVTGBLLHOSTD.                00000000
+0840  RCVTCART. 00000000  00000000            RCVTCNID. 00000001
+084C  RCVTINID. 00000000  RCVTABND. 00000000  RCVTJOBN. RESOLVER
+085C  RCVTSTIM. B79291E8  7F3A95C8            RCVTPTIM. 00000000  00000000
+0870  RCVTDEFLHOSTA.                00000005
+0874  RCVTDEFLHOSTI.                00000006
+0878  RCVTGBLLHOSTA.                00000007
+087C  RCVTGBLLHOSTI.                00000008
+0880  RCVTFLAGSCS...                80000000


Resolver Address Space is RESOLVER (ASID 01F8)
```

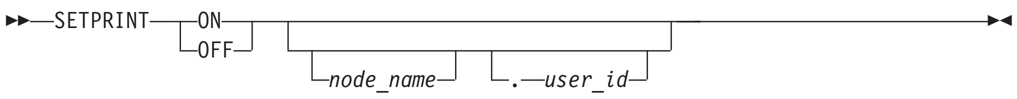
```
Global TCPIP.DATA file: SYS1.TCPPARMS(TCPDATA)

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_CTNAME..... C5E9C2C3  C3E3D9C3  00F89D00  00000000
+0018  CTCA_CTTOKEN.... B79291E8  7F6C3C68  0A448190  00F89D00
+0028  CTCA_CURCTBS.... 7E674000
+002C  CTCA_CURRCD..... 7E67CE0A
+0030  CTCA_TRACE...... 8A70CD98
+0034  CTCA_OPTWORD.... 09E93018
...
...
Resolver Task Data for USER557 Asid=0027 Tcb@=007F6AB0:

 RES_TASK: 7F6D6678
 +0000  RES_IDENTIFIER...... RTSK
 +0004  RES_LENGTH.......... 0BB8
 +0006  RES_SUBPOOL......... F9
 +0007  RES_USERKEY......... 06
 +0008  RES_SEQUENCE#....... 00000004
 +0010  RESMGR_TOKEN........ 00000000
 +0014  RESMGR_DATA......... 00000000
 +0018  RES_RTSK@........... 7F6D6678
 +001C  RES_RPID@........... 7F6D7230
        RES_STATE...........
 +0030  0000000A  00000001  000002C3  00000001  10020035  09438052  00000000
 +004C  00000000  00000000  00000000  00000000  00000000  00000000  00000000
 +0068  00000000  00000000  00000000  00000000  00000000  00000000  00000000
 +0084  00000000  00000000  00000000  00000000  00000000  00000000  00000000
 +00A0  00000000  00000000  00000000  00000000  00000000  00000000  00000000
...
...
+02D0  00000000  7F6D6950
       RES_STATE_EXT.......
+02D8  00000000  7F6D6BF8  00000000  7F6D6AF8  D4E5E2D9  00404040  00000000
+02F4  00000000  00000000  00000000  00000000  00000000  00000000  00000000
+0310  00000000  00000000  00000000  00000000  00000000  00000000  E3C3D7C3
+032C  E2000000  00E4E2C5  D9F5F500  00000000  00000000  00000000  00000000
+0348  00000000  00000000  00000000  00000000  00000000  00000000  00000000
+0364  00000000  00000000  00000000  00000000  00000000  00000000  00000000
...
...
+0680  RES_SEARCH_COUNT.... 00000003

        -- Array elements --
       RES_SEARCH_ENTRY....
+0684  RALEIGH.IBM.COM.......................................................
+06CA  ......................................................................
+0710  ......................................................................
+0756  ..............................................
       RES_SEARCH_ENTRY....
+0784  IBM.COM...............................................................
+07CA  ......................................................................
+0810  ......................................................................
+0856  ..............................................
       RES_SEARCH_ENTRY....
...
...
```

```
RRST: 7F6D66A8
+0000  RETRANS.. 0000000A  RETRY.... 00000001  OPTIONS.. 000002C3
+000C  NSCOUNT.. 00000001

        -- Array elements --
+0010  NSADDR_LIST...               10020035  09438052  00000000  00000000
+0020  NSADDR_LIST...               00000000  00000000  00000000  00000000
+0030  NSADDR_LIST...               00000000  00000000  00000000  00000000
+0040  NSADDR_LIST...               00000000  00000000  00000000  00000000
+0050  NSADDR_LIST...               00000000  00000000  00000000  00000000
+0060  NSADDR_LIST...               00000000  00000000  00000000  00000000
+0070  NSADDR_LIST...               00000000  00000000  00000000  00000000
+0080  NSADDR_LIST...               00000000  00000000  00000000  00000000
+0090  NSADDR_LIST...               00000000  00000000  00000000  00000000
+00A0  NSADDR_LIST...               00000000  00000000  00000000  00000000
...
...
+02A0  RES_VERSION...               00000000
+02A4  RES_EXTENSION.               7F6D6950

RRSX: 7F6D6950
+0004  STAT_EBCDICTOASCII..... 7F6D6BF8
+000C  STAT_ASCIITOEBCDIC..... 7F6D6AF8
       STAT_HOSTNAME..........
+0010  MVSR.   ..................................................
+0050  STAT_SERVICENAME....... TCPCS....
+0059  STAT_C_DSPRFX.......... USER55....................
+0078  $__IPDBCSNUM........... 00000000
...
```

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



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:

```
 setprint on ralvms.testid
  IKJ56861I  FILE IPCSPRNT NOT UNALLOCATED, DATA SET IS OPEN
```

Freeing IPCSPRNT when it is already freed:

```
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

```
►►──SKMSG──┬─skmb_address──────────────┬──┬─────────────────────────────┬──────────►◄
           ├─skdb_address──────────────┤  │              ┌─ALL──────┐   │
           ├─skbd_address──────────────┤  └─HELP──(──────┼──────────┼─)─┘
           ├─skqu_address──────────────┤                 ├─FUNCTION─┤
           ├─raw_control_block_address─┤                 ├─OPERANDS─┤
           ├─tcb_control_block_address─┤                 └─SYNTAX───┘
           ├─udp_control_block_address─┤
           └─*─────────────────────────┘
```

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


Message 1

 SKMB: 15D4D588
    +0000  id....... SKMB      next..... 00000000  prev..... 00000000
    +0008  tail..... 00000000  cont..... 15D55B08  flag..... 4000
    +0012  band..... 00        strx..... 16        hold..... 00000000
    +0018  datb..... 15D4D5B8  atch..... 00000000  rptr..... 15DE5A60
    +0024  wptr..... 15DE5AA8

 SKDB: 15D4D5B8
    +0000  id....... SKDB      msgb..... 15D4D588  cver..... 00
    +0009  csrc..... 80        ctyp..... 40
    +000C  tokn..... 15E3F040  15E3F3E0  00001358  alet..... 00000000
    +001C  base..... 15DE5000  size..... 00001000  flag..... 00000000
    +0028  ref...... 00000005

 Buffer: 15DE5000
    +0000  450005DC  24760000  4006DBF0  09433116  | ........ ..0.... |
    +0010  0943311A  0AB70866  481080F3  450C8271  | ...........3..b. |
    +0020  8010FFFE  DA3B0000  0101080A  3E456F23  | ..............?. |
    +0030  3E450C82  91A38897  D3C7E5D6  C297E8E3  | ...bjthpLGVOBpYT |
    +0040  D9F0E596  F2C989D9                      | R0Vo2IiR         |

 SKMB: 15D55B08
    +0000  id....... SKMB      next..... 00000000  prev..... 00000000
    +0008  tail..... 00000000  cont..... 00000000  flag..... 0000
    +0012  band..... 00        strx..... 00        hold..... 00000000
    +0018  datb..... 15D55B38  atch..... 00000000  rptr..... 13ECA758
    +0024  wptr..... 13ECACB8

 SKDB: 15D55B38
    +0000  id....... SKDB      msgb..... 15D55B08  cver..... 00
    +0009  csrc..... 40        ctyp..... 80
    +000C  tokn..... 15D41040  15D417F0  000003C7  alet..... 01FF0007
    +001C  base..... 13ECA000  size..... 00000CB8  flag..... 00800000
    +0028  ref...... 00000003
```

TCPHDR

Use the TCPHDR command to display the TCP header fields.

Syntax

```
►►──TCPHDR──┬─tcp_header_address─────────┬──┬──────┬───────────────────────────►
            ├─tcp_control_block_address──┤  ├─size─┤
            ├─skdb_address───────────────┤  └─*────┘
            ├─skmb_address───────────────┤
            └─*──────────────────────────┘

►──┬──────────────────────────────┬────────────────────────────────────────►◄
   │             ┌─ALL──────┐     │
   └─HELP──(─────┼──────────┼──)──┘
                 ├─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

```
►►──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: FFFFCA5B17000000
```

UDPHDR

Use the UDPHDR command to display the UDP header fields.

Syntax

```
►►──UDPHDR──┬─UDP_header_address─┬──┬──────┬──┬─────────────────────────────┬──────►◄
            ├─skdb_address───────┤  ├─size─┤  │         ┌─ALL──────┐        │
            ├─skmb_address───────┤  └─*────┘  └─HELP──(─┼──────────┼─)─────┘
            └─*──────────────────┘                      ├─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 08D0A0D8

UDP Header at 08D0A0EC

08D0A0EC  040700A1  0033CD23                          | ...~....          |

  Source port         : 1031
  Destination port    : 161
  Datagram Length     : 51
  Checksum            : CD23
```

Installing TCP/IP IPCS subcommands

Installation is made up of the following:
1. Install the members of the target data sets
2. Optionally, connect the TCP/IP panels to your existing ISPF panels

Installing the member of the target data sets

Table 15 shows the target data sets that contain the data necessary to set up the TCP/IP IPCS subcommands.

Copy the members to another data set that is currently in the concatenation DDNAME statements shown. Note that the target data sets contain other members, so you might not want to simply concatenate the target data set. A simple installation method is to change the TSO logon procedure to concatenate these data sets.

Table 15. Target data sets for TCP/IP IPCS subcommands

Target data set	Members	DD Name	Description
hlq.HELP	EZBIPCSH EZBIIHDH EZBIMHDH EZBIRESH EZBISKMH EZBITHDH EZBIUHDH SYSTCPDA SYSTCPIS	SYSHELP	TCPIPCS HELP text IPHDR Help Text ICMPHDR Help Text RESOLVER Help Text SKMSG Help Text TCPHDR Help Text UDPHDR Help Text SYSTCPDA Help Text SYSTCPIS Help Text
hlq.PARMLIB	EZAIPCSP	PARMLIB	IPCS verbexit mappings

Table 15. Target data sets for TCP/IP IPCS subcommands (continued)

Target data set	Members	DD Name	Description
hlq.SBLSCLI0	EZATFTHD (alias TCPHDR) EZATSPRI (alias SETPRINT) EZBTCPEX EZBTDENO (alias ERRNO) EZBTFICH (alias ICMPHDR) EZBTFIPH (alias IPHDR) EZBTFSKM (alias SKMSG) EZBTFTOD (alias TOD) EZBTFUPH (alias UDPHDR) EZBTIPCS (alias TCPIPCS) EZBTRESO (alias RESOLVER)	SYSEXEC	REXX execs
hlq.MIGLIB	EZBDGIPC	STEPLIB or IPCS TASKLIB	Load module
hlq.SBLSPNL0	EZBD* (approximately 170 members)	ISPPLIB	ISPF panels
hlq.SBLSTBL0	EZBDKEYS	ISPTLIB	ISPF key lists
hlq.SBLSMSG0	EZBF* (6 members)	ISPMLIB	ISPF messages

Connecting the TCP/IP panels

To use the panel interface to the TCP/IP IPCS subcommands, you can either invoke 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 invoke 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 invokes the following command:

```
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.

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 25 shows an example of an IPCS Primary Option Menu.

```
------------------------  IPCS PRIMARY OPTION MENU  --------
OPTION   ===>
   0   DEFAULTS    - Specify default dump and options
   1   BROWSE      - Browse dump data set
   2   ANALYSIS    - Analyze dump contents
   3   UTILITY     - Perform utility functions
   4   COMMAND     - Enter IPCS subcommand or CLIST
   5   TCP/IP      - TCP/IP analysis commands
   6   NCP         - NCP analysis commands
   7   NMP         - NMP analysis commands
   8   INVENTORY   - Inventory of problem data
   9   SUBMIT      - Submit problem analysis job to batch
   T   TUTORIAL    - Learn how to use the IPCS dialog
   X   EXIT        - Terminate using log and list defaults

Enter END command to terminate IPCS dialog
```

Figure 25. IPCS primary option menu

---

3. Select option 4, `COMMAND`.

---

4. Type the TCP/IP IPCS subcommand. Figure 26 shows the IPCS Subcommand Entry panel with a subcommand entered.

```
------------------------- IPCS Subcommand Entry -------------------------------
Enter a free-form IPCS subcommand or a CLIST or REXX exec invocation below:

===> tcpipcs help


----------------------- IPCS Subcommands and Abbreviations ---------------------
ADDDUMP          | DROPDUMP, DROP D  | LISTMAP, LMAP     | RUNCHAIN, RUN C
ANALYZE          | DROPMAP,  DROP M  | LISTSYM, LSYM     | SCAN
ARCHECK          | DROPSYM,  DROP S  | LISTUCB, LIST U   | SELECT
ASCBEXIT, ASCBX  | EQUATE,   EQU, EQ | LITERAL           | SETDEF,   SET D
ASMCHECK, ASMX   | FIND,     F       | LPAMAP            | STACK
CBFORMAT, CBF    | FINDMOD,  FMOD    | MERGE             | STATUS,   ST
CBSTAT           | FINDUCB,  FIND U  | NAME              | SUMMARY,  SUMM
CLOSE            | GTFTRACE, GTF     | NAMETOKN          | SYSTRACE
COPYDDIR         | INTEGER           | NOTE,    N        | TCBEXIT,  TCBX
COPYDUMP         | IPCS HELP, H      | OPEN              | WEBBEXIT, WEBBX
COPYTRC          | LIST,     L       | PROFILE, PROF     | WHERE,    W
CTRACE           | LISTDUMP, LDMP    | RENUM,   REN      |
```

Figure 26. IPCS subcommand entry panel with a TCP/IP IPCS subcommand entered

---

You can invoke the TCP/IP IPCS panels in one of the following ways:

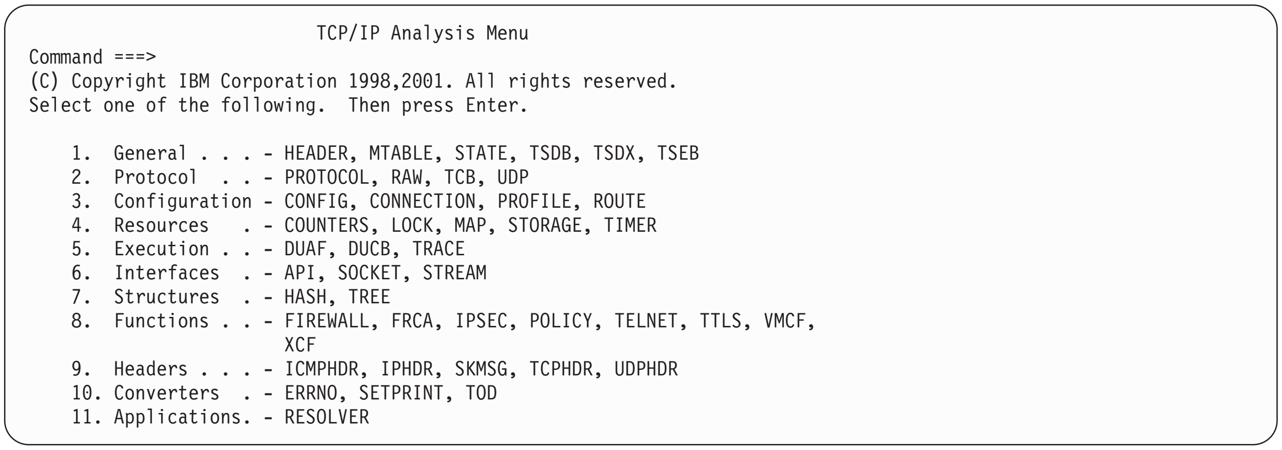
- Invoke the panel REXX exec as an IPCS Subcommand. Follow the steps above for entering a TCP/IP IPCS subcommand using the IPCS Subcommand Entry panel and enter the command:

  ```
  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 27.

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 running its own procedure, only the commands indicated "Available for Telnet" in the IPCS command list provide data. The commands not supported by Telnet return no data.



```
                           TCP/IP Analysis Menu
Command ===>
(C) Copyright IBM Corporation 1998,2001. All rights reserved.
Select one of the following.  Then press Enter.

    1.  General . . . - HEADER, MTABLE, STATE, TSDB, TSDX, TSEB
    2.  Protocol  . . - PROTOCOL, RAW, TCB, UDP
    3.  Configuration - CONFIG, CONNECTION, PROFILE, ROUTE
    4.  Resources   . - COUNTERS, LOCK, MAP, STORAGE, TIMER
    5.  Execution . . - DUAF, DUCB, TRACE
    6.  Interfaces  . - API, SOCKET, STREAM
    7.  Structures  . - HASH, TREE
    8.  Functions . . - FIREWALL, FRCA, IPSEC, POLICY, TELNET, TTLS, VMCF,
                        XCF
    9.  Headers . . . - ICMPHDR, IPHDR, SKMSG, TCPHDR, UDPHDR
    10. Converters  . - ERRNO, SETPRINT, TOD
    11. Applications. - RESOLVER
```

Figure 27. Main menu for TCP/IP IPCS subcommands.

Refer to *z/OS Communications Server: New Function Summary* for information regarding VTAM IPCS commands.

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 *z/OS MVS IPCS User's Guide* and *z/OS MVS IPCS Commands*.

The following is a sample command (single command):

```
%TCPIPCS 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 chapter discusses problem diagnosis of the Advisor and Agents and includes the following sections:

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 41 for details about how to use the IP packet trace facility.

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 41 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:

__ • 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.

__ • Issue display commands on the Advisor to determine whether any LBs have registered the application. Verify that the LB is connected to the Advisor.

__ • Verify that the Advisor’s *lb_id_list* statement includes the IP address of the LB in question.

__ • 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 it does not, 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. Correct the underlying network connectivity problem. For more information, see Chapter 4, "Diagnosing network connectivity problems," on page 27.

- __ 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 or the observed behavior. 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 it does not, correct the *agent_id_list* statement on the Advisor or the *advisor_id* statement on the Agent.
- __ 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 TCPSENDBFRSIZE 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 TCPSENDBFRSIZE 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.

Debug settings and corresponding syslogd priority levels

Table 16 summarizes the available debug levels and their associated syslogd priority levels.

Table 16. Available debug levels and associated syslogd priority levels

Logging category/Level	Description
None — 0	No messages of any kind are sent to the logging file after initialization is complete.
ERROR — 1	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.
WARNING — 2	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.
EVENT — 4	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.
INFO — 8	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.
MESSAGE — 16	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.

Table 16. Available debug levels and associated syslogd priority levels (continued)

Logging category/Level	**Description**
COLLECTION — 32	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. **Restriction:** COLLECTION is only used by the Agent. This level is intended for IBM service use only.
DEBUG — 64	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.
TRACE — 128	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.

Chapter 8. Diagnosing IKE daemon problems

This chapter describes how to diagnose IKE daemon problems, and contains the following sections:

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 325.

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 directory. The IKE daemon attempts to create the /var/ike directory at initialization. If IKE cannot create this directory, then initialization fails. The symptom for this problem is the following message: `EZD1045I IKE initialization error :  mkdir /var/ike failed`

  To correct this problem, ensure that /var is mounted read/write. If /var is mounted 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.

Common problems

Table 17 on page 293 lists common problems in establishing security associations.

Table 17. Establishing security associations problems

Problem	Symptom	Cause/response
Cannot send or receive packets on UDP ports 500 or 4500	EZD1065I When filter logging is active, message EZD0815I is issued, showing packets to UDP port 500 or UDP port 4500 that were denied.	The IKE daemon communicates using UDP ports 500 and 4500. See "Steps for verifying IP routing to a destination" on page 30 to verify that the IKE daemon is running and bound to ports 500 and 4500. Also, filter rules must be configured that permit inbound and outbound traffic on UDP ports 500 and 4500. Use the **ipsec -f display** command to confirm that filter rules that permit UDP ports 500 and 4500 traffic are installed in the stack. Activate filter logging for these rules so that you can observe packets sent and received on ports 500 and 4500 in the syslog. For information about the **ipsec** command, refer to *z/OS Communications Server: IP System Administrator's Commands*. Refer to *z/OS Communications Server: IP Configuration Guide* for general information about configuring IP filters.
Pre-shared key mismatch	Message EZD0965I was issued.	If IKE is using pre-shared key mode authentication and it cannot interpret a decrypted message that it has received, then message EZD0965I is issued, indicating a likely pre-shared key mismatch. In main mode, the responder gets the message upon receipt of message 5. In aggressive mode, the initiator gets the message upon receipt of message 2. EZD0965I can also be issued if IKE receives a corrupted message even though the pre-shared keys match. If the remote peer cannot decrypt the message that was sent by IKE because of a pre-shared key mismatch, the local symptom is that IKE retransmits the first encrypted message of the exchange. Review the pre-shared key configuration on the local and remote system and ensure that the keys match. **Tip:** The keys might be represented differently (for example, ASCII or EBCDIC) on the local and remote system.

Table 17. Establishing security associations problems (continued)

Problem	Symptom	Cause/response
RSA signature mode authentication failure	One of the following messages was issued: • EZD0990I • EZD1030I • EZD1037I • EZD0981I • EZD1075I	For the IKE daemon to support RSA signature mode authentication, it must be able to access certificates on the SAF key ring. IKE issues message EZD0990I to indicate that RSA signature mode is supported or EZD1030I if RSA signature mode is not supported. Refer to the messages to determine the appropriate response. The key ring is specified on the KeyRing parameter in the IkeConfig statement. When configuring with the z/OS Network Security Configuration Assistant GUI, the key ring is specified on the key ring database field on the IPSec: IKE Daemon Settings panel. 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. Display the certificates on the SAF key ring. Ensure that all the certificates on the key ring that are to be used by the IKE daemon include a digital signature. If you are using RACF, make sure that the trust status of the certificates is TRUST or HIGHTRUST. 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. 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 Remote Security End Point statement, message EZD0981I is issued. If the peer detects such a mismatch, it might send an "Invalid ID information" notification. If IKE receives such a notification, message EZD1075I is issued. Refer to the messages to determine the appropriate response.

Table 17. Establishing security associations problems (continued)

Problem	**Symptom**	**Cause/response**
RSA signature mode authentication failure (continued)	One of the following messages was issued: • EZD0990I • EZD1030I • EZD1037I • EZD0981I • EZD1075I	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 IKE SyslogLevel parameter in the IkeConfig statement. When configuring with the z/OS Network Security Configuration Assistant 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 *z/OS Communications Server: IP Configuration Reference* for information about setting the IkeSyslogLevel, or see the online help in the z/OS Network Security Configuration Assistant. **Tip:** The name of the key ring is case sensitive.

Table 17. Establishing security associations problems (continued)

Problem	Symptom	Cause/response
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 z/OS Network Security Configuration Assistant GUI, 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 17. Establishing security associations problems (continued)

Problem	Symptom	Cause/response
Phase 1 policy mismatch	Message EZD1093I or EZD1075I was issued.	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 z/OS Network Security Configuration Assistant GUI, 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.

Table 17. Establishing security associations problems (continued)

Problem	Symptom	Cause/response
Failure to locate phase 2 policy	Message EZD1024I was issued	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 z/OS Network Security Configuration Assistant GUI, 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 operation" on page 651, supplying the IP traffic characteristics identified on the EZD1024I message.

Table 17. Establishing security associations problems (continued)

Problem	Symptom	Cause/response
Phase 2 policy mismatch	Message EZD1022I, EZD1093I, or EZD1075I was issued.	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 z/OS Network Security Configuration Assistant GUI, 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.

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 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 z/OS Network

Security Configuration Assistant GUI, 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.
- Traversing a NAT that performs translation of the remote security endpoint's port is not supported.

  The z/OS IKE daemon does not support initiating or responding to a remote security endpoint when a NAT has translated the remote security endpoint's port (for example, when IKE detects the existence and of a NAPT in front of the remote security gateway). If this condition is detected during a phase 1 negotiation, message EZD1085I is issued and the negotiation fails.
- 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 can always initiate a phase 2 negotiation with a z/OS remote security endpoint that has NAT traversal support enabled. The z/OS IKE daemon provides limited support to initiate a phase 2 negotiation when a NAT is detected between IKE and a non-z/OS remote security endpoint.

  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 a 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.

  A new IPSec security association is the first IPSec security association negotiated for a particular traffic pattern. A remote security endpoint is acting as a security gateway when the remote endpoint of the IPSec security association is not the same as the IP address used as the remote IP address of the protecting ISAKMP security association.

- 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.

- z/OS might not be able to initiate to some remote security endpoints

  When z/OS initiates a phase 2 negotiation for a new host-to-host dynamic tunnel that protects all ports and all protocols, it allows the traffic pattern to default to the IP addresses of the local and remote security endpoints. This means that IKE and its peer views 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/OSs public address and the peer's private address.

  A host-to-host dynamic tunnel uses either transport or tunnel mode of encapsulation. Most non-z/OS IKE implementations operate with z/OS when the non-z/OS implementation initiates a phase 2 negotiation for a new transport mode host-to-host dynamic tunnel that protects all ports and all protocols. Interoperability with a non-z/OS implementation is less certain when z/OS initiates a phase 2 negotiation for a new tunnel mode host-to-host dynamic tunnel that protects all ports and all protocols. In this case, it is possible that the negotiation succeeds, but it produces an SA that cannot be used to send traffic. This is partially because the fact that 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.

The traffic endpoints cannot be defaulted 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 NATs are being traversed. When z/OS initiates a phase 2 negotiation for a new host-to-host dynamic tunnel that protects a specific ports or protocol, it defines the traffic pattern utilizing z/OS's private address as the local endpoint and the peer's public address as the remote endpoint.

Interoperability with a non-z/OS implementation is also uncertain when z/OS initiates a phase 2 negotiation for a host-to-host dynamic tunnel that protects a specific port or protocol. This is due partially to the fact that RFC 3947 does not define how the traffic pattern should be specified in this case.

In cases where interoperability is uncertain, the following might occur:

– The SA negotiation fails
– The SA negotiation succeeds in creating an SA that cannot be used to send and receive traffic
– The SA negotiation succeeds in creating an SA that can be used to send and receive traffic

The result depends on the NAT Traversal implementation being used by the remote peer. If the remote peer is z/OS IKE, the SA negotiation always succeeds, and the SA is always usable.

In order to help identify such configurations, three informational messages have been defined. When z/OS attempts to create a UdpEncapsulatedTunnel mode, SA message EZD1104I or EZD1105I is issued. When z/OS attempts to create an SA for a specific port protocol, message EZD1107I is issued. In all cases, the negotiation continues.

- SWSA implications

  During VIPA takeover/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 expects a non-IPv4 identity during a quick mode exchange; however, it can act as a responder in these cases. When a VIPA takeover/giveback occurs the IKE daemon does not attempt to re-establish such phase 2 security associations.

  There are also cases where the results might be unpredictable when the z/OS IKE daemon attempts to initiate the creation of new IPSec security associations. These cases include:

  – z/OS IKE attempts to create a new udp_encapsulated_tunnel mode SA
  – z/OS IKE attempts to create an SA for a specific port, protocol, or both

  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.

If the IKE daemon or stack detects such a change while there are one or more security associations with that endpoint, the IKE daemon attempts to verify the new IP address. It does this by initiating the creation of a new ISAKMP security association using the remote security endpoint's new IP address. 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. This mapping could expire as a result of inactivity. In order to prevent the expiration of this mapping, the stack occasionally sends messages known as NAT keep alive messages. If these messages are not sent frequently enough, the NAT box could expire the mapping of z/OS's 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* for more details. When configuring with the z/OS Network Security Configuration Assistant GUI, 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 28 shows a NAT keep alive message:

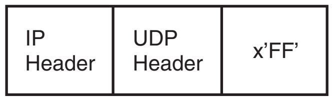


Figure 28. NAT keep alive message

- 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.

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. Refer to the *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(*)'
```

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 z/OS Network Security Configuration Assistant GUI, 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* or the z/OS Network Security Configuration Assistant'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 z/OS Network Security Configuration Assistant, 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 service as it impacts IKE performance. For more information about setting the Pagent Agent Syslog levels, refer to *z/OS Communications Server: IP Configuration Reference* or the z/OS Network Security Configuration Assistant's online helps.

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 41.

For detailed information, refer to the following books:

- *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 18 lists the IKE daemon trace options.

Table 18. IKE daemon trace options

Trace Event	Description
ALL	Trace all types of records. This option slows performance.
MINIMUM	Trace the IKE daemon's minimum level of tracing.
INIT	Trace IKE daemon initialization information.
TERM	Trace IKE daemon termination information.
EXCEPT	Trace IKE daemon exception information.
CONFIG	Trace IKE daemon configuration information.
WORKUNIT	Trace IKE workunit information.
SERIAL	Trace IKE serialization information.
IKE	Trace IKE protocol information.
CRYPTO	Trace IKE cryptographic information.
OPMSGS	Trace IKE operator messages.
LOGMSGS	Trace IKE syslog messages.
MSGQ	Trace IKE message queue information.
TIMER	Trace IKE timer information.
SOCKETS	Trace IKE socket information.
IOCTL	Trace IKE IOCTL call information.
REQUESTS	Trace IKE request information.
FLOW	Trace IKE code flow information.
STORAGE	Trace IKE storage information.
EVENT	Trace IKE event information.
POLICY	Trace IKE policy information.
CONTROL	Trace IKE daemon control information.
MISC	Trace IKE miscellaneous information.
DEBUG	Trace IKE debugging information.

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 18 on page 305.

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 IKE_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.

```
/*******************************************************************/
/*                                                                 */
/* z/OS Communications Server                                      */
/* SMP/E Distribution Name: CTIIKE00                               */
/*                                                                 */
/* PART Name: CTIIKE00                                             */
/*                                                                 */
/*                                                                 */
/* Copyright:                                                      */
/* Licensed Materials - Property of IBM                            */
/* 5694-A01                                                        */
/* (C) Copyright IBM Corp. 2005                                    */
/* Status:  CSV1R7                                                 */
/*                                                                 */
/*                                                                 */
/* 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
 /* ------------------------------------------------------------- */
 /* 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 256M                        */
 /* CTRACE buffers reside in IKE daemon Private storage           */
 /* which is in the regions address space.                        */
 /* ------------------------------------------------------------- */
 BUFSIZE(1M)
 /* WTR(wtr_procedure)                                            */
 /* ------------------------------------------------------------- */
 /* OPTIONS: NAMES OF FUNCTIONS TO BE TRACED, OR "ALL"            */
 /* ------------------------------------------------------------- */
 /* OPTIONS(          */
 /*   'ALL     '      */
 /* , 'MINIMUM '      */
 /* , 'INIT    '      */
 /* , 'TERM    '      */
 /* , 'EXCEPT  '      */
```

Figure 29. SYS1.PARMLIB member CTIIKE00 (Part 1 of 2)

```
/* , 'CONFIG  '      */
/* , 'WORKUNIT'      */
/* , 'SERIAL  '      */
/* , 'IKE     '      */
/* , 'CRYPTO  '      */
/* , 'OPMSGS  '      */
/* , 'LOGMSGS '      */
/* , 'MSGQ    '      */
/* , 'TIMER   '      */
/* , 'SOCKETS '      */
/* , 'IOCTL   '      */
/* , 'REQUESTS'      */
/* , 'FLOW    '      */
/* , 'STORAGE '      */
/* , 'EVENT   '      */
/* , 'POLICY  '      */
/* , 'CONTROL '      */
/* , 'MISC    '      */
/* , 'DEBUG   '      */
/* )                 */
```

Figure 29. SYS1.PARMLIB member CTIIKE00 (Part 2 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:

```
TRACE CT,WTRSTART=writer_proc
TRACE CT,ON,COMP=SYSTCPIK,SUB=(iked_jobname)
R xx,OPTIONS=(option[,option2...]),WTR=writer_proc,END
```

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

- Issue the following console commands to disable a CTRACE to an external writer:

```
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 41.

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 41 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 18 on page 305, except ALL and MINIMUM.

Chapter 9. Diagnosing dynamic VIPA and sysplex problems

This chapter presents diagnostic information for dynamic VIPA and sysplex problems, and contains the following sections:

- “Overview of diagnosing Sysplex Distributor problems”
- “Steps for diagnosing sysplex problems” on page 312
- “Steps for diagnosing sysplex-wide dynamic source VIPAs for TCP connections problems” on page 322
- “Steps for diagnosing SYSPLEXPORTS problems” on page 323
- “Diagnosing Sysplex-wide Security Association (SWSA) problems” on page 324
- “Steps for diagnosing sysplex routing problems” on page 328

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 **netstat/onetstat** from commands 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 312:

- For problems where the actual DVIPAs defined on a stack are not what you expected, confirm the current definitions on a stack. See step 1.
- For Sysplex Distributor workload monitoring, use steps 6 and 9. If the output from these commands is not what you expected, see step 5 for an overall picture of all DVIPA activity in your sysplex.
- If the output from step 5 reveals an expected target stack not listed for a distributed DVIPA, perform step 2 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 10 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. 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 30 on page 313 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 TIMEDAFFinity 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: 45    FLG: BASEWLM
    DESTINTF:    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
           SysPt:   No   TimAff: No    Flg: BaseWLM
```

Figure 30. Netstat VIPADCFG/-F example

---

2. 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 31 on page 314 shows the output of this command for the distributing TCP/IP:

```
D TCPIP,TCPCS,N,CONFIG
EZD0101I NETSTAT CS V1R7 TCPCS 928
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
UDP CONFIGURATION TABLE:
DEFAULTRCVBUFSIZE: 00065535  DEFAULTSNDBUFSIZE: 00065535
CHECKSUM:          YES
RESTRICTLOWPORT:   YES       UDPQUEUELIMIT:     YES
IP CONFIGURATION TABLE:
FORWARDING: YES    TIMETOLIVE: 00064  RSMTIMEOUT:  00060
FIREWALL:   NO
ARPTIMEOUT: 01200  MAXRSMSIZE: 65535  FORMAT:      LONG
IGREDIRECT: YES    SYSPLXROUT: YES    DOUBLENOP:   NO
STOPCLAWER: NO     SOURCEVIPA: YES
MULTIPATH:  NO     PATHMTUDSC: NO     DEVRTRYDUR:  0000000090
DYNAMICXCF: YES
  IPADDR/PREFIXLEN: 193.15.1.1/24       METRIC: 02
IQDIOROUTE: NO
TCPSTACKSRCVIPA: 203.15.1.1
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
SMF PARAMETERS:
TYPE 118:
  TCPINIT:      00   TCPTERM:    00   FTPCLIENT:    00
  TN3270CLIENT: 00   TCPIPSTATS: 00
TYPE 119:
  TCPINIT:      NO   TCPTERM:    NO   FTPCLIENT:    NO
  TCPIPSTATS:   NO   IFSTATS:    NO   PORTSTATS:    NO
  STACK:        NO   UDPTERM:    NO   TN3270CLIENT: NO
GLOBAL CONFIGURATION INFORMATION:
TCPIPSTATS: NO   ECSALIMIT: 0000000K  POOLLIMIT: 0000000K
MLSCHKTERM: NO
SYSPLEX MONITOR: TIMERSECS: 0060 RECOVERY: NO DELAYJOIN: NO AUTOREJOIN: NO
```

Figure 31. Netstat CONFIG/-f example

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*. Figure 32 on page 315 shows an example:

```
D WLM,SYSTEMS
IWM025I  16.38.58  WLM DISPLAY 963
  ACTIVE WORKLOAD MANAGEMENT SERVICE POLICY NAME: DEFAULT
  ACTIVATED: 2003/10/29  AT: 14:51:50  BY: N/A       FROM: VIC015
  DESCRIPTION: IBM'S WLM DEFAULT POLICY
  RELATED SERVICE DEFINITION NAME: N/A
  INSTALLED: 2003/10/29  AT: 14:51:50  BY: N/A       FROM: N/A
  WLM VERSION LEVEL:       LEVEL013
  WLM FUNCTIONALITY LEVEL: LEVEL001
  STRUCTURE SYSZWLM_WORKUNIT STATUS: DISCONNECTED
  *SYSNAME*  *MODE*  *POLICY*  *WORKLOAD MANAGEMENT STATUS*
  VIC015     GOAL    DEFAULT   ACTIVE, NOT RUNNING WITH ACTIVE POLICY
```

Figure 32. D WLM,SYSTEMS example

3. 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 33 shows an example:

```
D TCPIP,TCPCS,NET,VIPADYN
EZD0101I NETSTAT CS V1R7 TCPCS 827
IPADDR/PREFIXLEN: 201.2.10.11/28
  STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
IPADDR/PREFIXLEN: 201.2.10.12/28
  STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST
IPADDR/PREFIXLEN: 201.2.10.13/28
  STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
IPADDR/PREFIXLEN: 201.2.10.21
  STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT:
IPADDR/PREFIXLEN: 201.2.10.22
  STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT:
INTFNAME: DVIPA1
  IPADDR: 2001:0DB8:1::1
    STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
```

Figure 33. Netstat VIPADYN/-v example

4. 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 34 on page 316 shows an example:

```
D TCPIP,TCPCS2,NET,VIPADYN
EZD0101I NETSTAT CS V1R7 TCPCS2 905
IPADDR/PREFIXLEN: 201.2.10.11/28
  STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
IPADDR/PREFIXLEN: 201.2.10.12/28
  STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT: DEST
IPADDR/PREFIXLEN: 201.2.10.13/28
  STATUS: ACTIVE     ORIGIN:                  DISTSTAT: DEST
IPADDR/PREFIXLEN: 201.2.10.21
  STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT:
IPADDR/PREFIXLEN: 201.2.10.22
  STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT:
INTFNAME: DVIPA1
  IPADDR: 2001:0DB8:1::1
    STATUS: ACTIVE     ORIGIN:                  DISTSTAT: DEST
```

Figure 34. Netstat VIPADyn/-v example

5. 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 35 on page 317 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

```
D TCPIP,TCPCS2,SYSPLEX,VIPAD
EZZ8260I SYSPLEX CS V1R7 948
VIPA DYNAMIC DISPLAY FROM TCPCS2   AT VIC015
IPADDR: 201.2.10.11
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPCS    VIC015   ACTIVE      BOTH
  TCPCS2   VIC015   BACKUP 100  DEST
  TCPCS3   VIC015   BACKUP 010  DEST
IPADDR: 201.2.10.12
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPCS    VIC015   ACTIVE      DIST
  TCPCS2   VIC015   BACKUP 075  DEST
  TCPCS3   VIC015   BACKUP 010
IPADDR: 201.2.10.13
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPCS    VIC015   ACTIVE      BOTH
  TCPCS3   VIC015   BACKUP 010  DEST
  TCPCS2   VIC015   ACTIVE      DEST
IPADDR: 201.2.10.21
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPCS3   VIC015   ACTIVE
  TCPCS2   VIC015   BACKUP 100
  TCPCS    VIC015   BACKUP 080
IPADDR: 201.2.10.22
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPCS3   VIC015   ACTIVE
  TCPCS    VIC015   BACKUP 080
  TCPCS2   VIC015   BACKUP 075
INTFNAME: DVIPA1
IPADDR: 2001:0DB8:1::1
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPCS    VIC015   ACTIVE      BOTH
  TCPCS2   VIC015   ACTIVE      DEST
```

Figure 35. Sysplex VIPADyn example

| 6. Run the 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 target server connection responsiveness factors that make up the TSR and the current WLM or QOS weight for each service level mapping to a 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,vipadcfg**, 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 36 on page 319 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:   75
FLG:SERVERWLM
    TCSR: 100 CER:  75 SEF: 79
     QosPlcAct:*DEFAULT*
       W/Q:01

DEST:201.2.10.11..20
DESTXCF:193.15.1.1
TOTALCONN:0000000330 RDY:001 WLM:9 TSR:   65
FLG:SERVERWLM
    TCSR:  87 CER:  75 SEF: 79
     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
     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
     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
     QosPlcAct:*DEFAULT*
       W/Q:01
DEST:201.2.10.11..21
DESTXCF:193.15.3.1
TOTALCONN:0000000007 RDY:001 WLM:14 TSR:  92
FLG:SERVERWLM
    TCSR:  97 CER:  98 SEF: 97
     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
     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
     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
     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
     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
     QosPlcAct:*DEFAULT*
       W/Q:01
DESTINTF:DVIPA1
DEST:1::1..6000
DESTXCF:FEC0::151:0
TOTALCONN:0000000511 RDY:001 WLM:11 TSR:   79
FLG:SERVERWLM
    TCSR:  99 CER:  98 SEF:  80
     QosPlcAct:*DEFAULT*
       W/Q:01
DESTINTF:DVIPA1
DEST:1::1..6000
DESTXCF:FEC0::152:0
TOTALCONN:0000001410 RDY:001 WLM:10 TSR:    93
FLG:SERVERWLM
    TCSR: 100 CER:   95 SEF: 100
     QosPlcAct:*DEFAULT*
       W/Q:01
```

Figure 36. Netstat VDPT/-O example

7. 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 10 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.

---

8. 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 2 for instructions for verifying this. Also, verify that WLM is configured and active on all participating stacks. See step 2.

- 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 CER value is low, then this indicates that the target stack is having problems establishing connections with the client stack.
  - 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 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 TCPIP,TCPCS,NET,VCRT,DETAIL (see step 9) 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 605 for more information.

---

9. 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 37 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 10. Figure 37 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 37. d tcpip,tcpcs,net,vcrt,detail example

---

10. 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 38 shows an example:

```
D TCPIP,TCPCS2,NET,ALLCONN,IPA=201.2.10.12
EZD0101I NETSTAT CS V1R7 TCPCS2 846
USER ID  CONN     STATE
FTPD1    000000F3 ESTBLSH
  LOCAL SOCKET:   ::FFFF:201.2.10.12..21
  FOREIGN SOCKET: ::FFFF:203.110.1.1..1033
1 OF 1 RECORDS DISPLAYED
```

Figure 38. Netstat ALLConn/-a example with IPAddr/-I filter value of 201.2.1.12

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 sysplex-wide dynamic source VIPAs 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. 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. Figure 39 an example.

1. Run the Netstat CONFIG/-f command on the target stacks to verify that the sysplex-wide dynamic source VIPA was configured as expected (that is, verify that IPCONFIG/IPCONFIG6 SOURCEVIPA is set to YES and that IPCONFIG/IPCONFIG6 TCPSTACKSRCVIPA is specified with the correct address or interface). Figure 39 shows an example:

```
IP CONFIGURATION TABLE:
FORWARDING: YES    TIMETOLIVE: 00064  RSMTIMEOUT:  00060
FIREWALL:   NO
ARPTIMEOUT: 01200  MAXRSMSIZE: 65535  FORMAT:      LONG
IGREDIRECT: YES    SYSPLXROUT: YES    DOUBLENOP:   NO
STOPCLAWER: NO     SOURCEVIPA: YES
MULTIPATH:  NO     PATHMTUDSC: NO     DEVRTRYDUR:  0000000090
DYNAMICXCF: YES
  IPADDR/PREFIXLEN: 193.15.1.1/24       METRIC: 02
IQDIOROUTE: NO
TCPSTACKSRCVIPA: 203.15.1.1
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
TCPSTACKSRCVIPA: V6INTF1A
```

Figure 39. Netstat CONFIG/-f example

2. If you are using the job-specific source IP addressing (SRCIP) function, then run the Netstat SRCIP/-J command to display the SRCIP configuration. Verify that the jobname displayed matches the application performing the outbound CONNECT(), as shown in the following example.

```
D TCPIP,,N,SRCIP
EZD0101I NETSTAT CS V1R7 TCPCS 655
JOB NAME  TYPE  INTERFACE
--------  ----  ---------
USER*     IPV4  203.15.2.1
USER*     IPV6  2003::15:1:1
2 OF 2 RECORDS DISPLAYED
```

Figure 40. 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.

Steps for diagnosing SYSPLEXPORTS problems

Perform the following steps to diagnose SYSPLEXPORTS problems.

1. Run the Netstat VIPADCFG/-F command on the distributor stack to confirm that SYSPLEXPORTS was specified for all distributed Dynamic VIPAs as expected.

```
D TCPIP,TCPCS,NET,VIPADCFG
EZD0101I NETSTAT CS V1R7 TCPCS 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 41. Diagnosing SYSPLEXPORTS problems

In the preceding display, the distributed Dynamic VIPA 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 the system log that the following message was issued:

   IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBEPORT

   If this message was not issued and netstat vipadcfg shows that SYSPLEXPORTS was specified, refer to *z/OS Communications Server: SNA Network Implementation Guide* for more information about defining EZBEPORT 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 VIPA. 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 knows about the connection.
   c. Issue the VTAM display VTAM NET,STATS command to confirm that the coupling facility is managing ports for this Distributed VIPA. Note that for ephemeral ports the coupling facility assigns a block of 64 ports to the TCP/IP stack. For example:

```
D NET,STATS,TYPE=CFS,STRNAME=EZBEPORT,LIST=ALL,SCOPE=ALL
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = STATS,TYPE=CFS 180
IST1370I NETA.SSCP1A IS CONNECTED TO STRUCTURE EZBEPORT
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 -------------------------------------------------------------
IST1823I LIST DVIPA SYSNAME  TCPNAME                # ASSIGNED PORTS
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 42. VTAM NET,STATS example

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. 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 43. ipsec -f example

   - On FIREWALL stacks, use the **netstat,config** command to confirm that FIREWALL and DVIPSEC were specified on the IPCONFIG statement.

```
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 44. netstat,config example

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
   ```

   For SWSA functions to work correctly, the stacks involved must be connected to the EZBDVIPA coupling facility structure. If this message was not issued and the Netstat CONFIG/-f command or **ipsec** command show that DVIPSEC was specified, refer to *z/OS Communications Server: SNA Network Implementation Guide* for information about setting up the sysplex environment for VTAM function and defining EZBDVIPA 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:

```
# ipsec -y display
CS V1R7 ipsec  TCPIP Name: TCPCS1  Wed Jun 30 10:54:44 2004
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 2

TunnelID                      Y29
VpnActionName:                ESP10080m-tran
State:                        Active
LocalEndPoint:                197.11.235.107
RemoteEndPoint:               197.11.107.1
HowtoEncap:                   Transport
HowToAuth:                    ESP
 AuthAlgorithm:               Hmac_Md5
 AuthInboundSpi:              1508989086
 AuthOutboundSpi:             4121663008
HowToEncrypt:                 DES
 EncryptInboundSpi:           1508989086
 EncryptOutboundSpi:          4121663008
Lifesize:                     0K
LifesizeRefresh:              0K
CurrentByteCount:             0b
LifetimeRefresh:              2004/06/30 10:57:41
LifetimeExpires:              2004/06/30 11:10:53
CurrentTime:                  2004/06/30 10:11:46
VPNLifeExpires:               2004/07/07 10:10:29
ParentIKETunnelID:            K10
LocalDynVpnRule:              DT235.107.ftp1820
NAT Traversal Topology:
  UdpEncapMode: No
  LclNATDetected: No
  RmtNATDetected: No
  RmtIsGw: No
  RmtIsZOS: No
  zOSCanInitP2SA: Yes
  SrcNATOARcvd: n/a
  DstNATOARcvd: n/a
************************************************************************
```

Figure 45. ipsec -y example

b. Use the following command to verify that a shadow tunnel is active on IP security target stacks:

```
# ipsec -y display -s
CS V1R7 ipsec  TCPIP Name: TCPCS1  Wed Jun 30 10:54:48 2004
Primary:  Dynamic tunnel  Function: Display           Format:   Detail
Source:   Stack           Scope:    Current           TotAvail: 2

TunnelID                      Y29
VpnActionName:                ESP10080m-tran
State:                        Shadow
LocalEndPoint:                197.11.235.107
RemoteEndPoint:               197.11.107.1
HowtoEncap:                   Transport
HowToAuth:                    ESP
 AuthAlgorithm:               Hmac_Md5
 AuthInboundSpi:              1508989086
 AuthOutboundSpi:             4121663008
HowToEncrypt:                 DES
 EncryptInboundSpi:           1508989086
 EncryptOutboundSpi:          4121663008
Lifesize:                     0K
LifesizeRefresh:              0K
CurrentByteCount:             0b
LifetimeRefresh:              2004/06/30 10:57:41
LifetimeExpires:              2004/06/30 11:10:53
CurrentTime:                  2004/06/30 10:12:24
VPNLifeExpires:               2004/07/07 10:10:29
ParentIKETunnelID:            K10
LocalDynVpnRule:              DT235.107.ftp1820
NAT Traversal Topology:
  UdpEncapMode: No
  LclNATDetected: No
  RmtNATDetected: No
  RmtIsGw: No
  RmtIsZOS: No
  zOSCanInitP2SA: Yes
  SrcNATOARcvd: n/a
  DstNATOARcvd: n/a
**************************************************************************
```

Figure 46. ipsec -y display -s example

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:

```
d net,stats,type=cfs,strname=ezbdvipa,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 EZBDVIPA
IST1371I STRUCTURE TYPE = LIST
```

```
IST1517I LIST HEADERS = 1024 - LOCK HEADERS = 0
IST1373I STORAGE ELEMENT SIZE = 256
IST924I -------------------------------------------------------------
IST1374I                              CURRENT        MAXIMUM       PERCENT
IST1375I STRUCTURE SIZE                  17K          6896K          *NA*
IST1376I STORAGE ELEMENTS                 32          14839             0
IST1377I LIST ENTRIES                      4           1485             0
IST924I -------------------------------------------------------------
IST1834I LIST DVIPA SYSNAME   TCPNAME #ENTRIES TGCOUNT SEQNUMBER
IST1835I    1 203.15.1.1
IST1836I            VIC015   TCPCS           2       1
IST314I END
```

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 dynamic VIPA 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 dynamic VIPA, the tunnel has to be recovered by the same stack that took over the dynamic VIPA.

The list entry for the dynamic VIPA (list 1 above) shows the system and stack for which the coupling facility is maintaining information about the tunnel.

---

5. Use the following VTAM command to confirm that the coupling facility is managing the replay count:

```
d net,stats,type=cfs,strname=ezbdvipa,scope=all,list=all
```

For sysplex distribution of IPSec traffic, the dynamic tunnel's replay count (sequence number) is maintained in the EZBDVIPA coupling facility structure. The distributing stack's dynamic tunnel and all the target stack's shadow tunnels share the replay count.

The following output is displayed:

```
D NET,STATS,TYPE=CFS,STRNAME=EZBDVIPA,LIST=ALL
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = STATS,TYPE=CFS 854
IST1370I NETA.SSCP2A IS CONNECTED TO STRUCTURE EZBDVIPA
IST1797I STRUCTURE TYPE = LIST
IST1517I LIST HEADERS = 2048 - LOCK HEADERS = 0
IST1373I STORAGE ELEMENT SIZE = 256
IST924I -------------------------------------------------------------
IST1374I                          CURRENT      MAXIMUM    PERCENT
IST1375I STRUCTURE SIZE             6144K       10240K       *NA*
IST1376I STORAGE ELEMENTS              48         9576          0
IST1377I LIST ENTRIES                   5          958          0
IST924I -------------------------------------------------------------
IST1834I LIST DVIPA SYSNAME TCPNAME  #ENTRIES     TGCOUNT   SEQNUMBER
IST1835I    1 203.12.3.10
IST1836I         VIC012 TCPCS3 3 1
IST1839I 00000000000000000000000000000000
IST314I END
```

The list entry for the dynamic VIPA with a value in the SEQNUMBER column confirms that this tunnel's replay count is managed by the coupling facility.

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.

```
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:F003
    TARGETIP: 1EC0::5:1:1
    RTSTATUS: ACTIVE
  DESTXCF: 2EC0::943:F004
    TARGETIP: 1EC0::6:2:2
    RTSTATUS: INACTIVE
4 OF 4 RECORDS DISPLAYED
```

Figure 47. Netstat VIPADyn example

- 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 *z/OS 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,ROUTE
EZD0101I NETSTAT CS V1R7 TCPCS
IPV4 DESTINATIONS
DESTINATION        GATEWAY          FLAGS     REFCNT  INTERFACE
193.1.1.94/32      0.0.0.0          H         000000  EZASAMEMVS
193.1.1.94/32      0.0.0.0          UH        000000  EZAXCFC6
193.1.1.94/32      0.0.0.0          UH        000000  EZAXCFC7
193.1.3.94/32      0.0.0.0          UHS       000000  EZAXCFC6
193.1.4.94/32      0.0.0.0          UHS       000000  EZAXCFC7
203.1.1.94/32      0.0.0.0          UH        000000  VIPLCB01015E
213.4.1.1/32       0.0.0.0          UH        000000  LTRLE1A
213.4.2.2/32       0.0.0.0          H         000000  LTRLE2A
213.5.1.1/32       0.0.0.0          UHZ       000001  LTRLE1A
```

```
213.6.2.2/32        0.0.0.0          HZ        000001  LTRLE2A
IPV6 DESTINATIONS
DESTIP:   ::1/128
  GW:     ::
  INTF:   LOOPBACK6         REFCNT:  000000
  FLGS:   UH                MTU: 65535
DESTIP:   1EC0::4:1:1/128
  GW:     ::
  INTF:   V6TRLE1A          REFCNT:  000000
  FLGS:   UH                MTU: 14336
DESTIP:   1EC0::4:2:2/128
  GW:     ::
  INTF:   V6TRLE2A          REFCNT:  000000
  FLGS:   H                 MTU: 0
DESTIP:   1EC0::5:1:1/128
  GW:     ::
  INTF:   V6TRLE1A          REFCNT:  000001
  FLGS:   UHZ               MTU: 14336
DESTIP:   1EC0::6:2:2/128
  GW:     ::
  INTF:   V6TRLE2A          REFCNT:  000001
  FLGS:   HZ                MTU: 32000
.
.
32 OF 32 RECORDS DISPLAYED
END OF THE REPORT
```

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
```

Figure 48. Netstat VCRT/-V detail example

See "Routing failures" on page 668 for additional information about routing failures.

Chapter 10. Diagnosing access control problems

This chapter describes selected procedures for TCP/IP Services component trace, packet trace, and Socket API trace.

This chapter contains the following sections:

- "Overview of access control support"
- "Diagnosing multilevel security consistency check messages (EZD1215-EZD1234)" on page 335

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 chapter 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 chapter 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 RACLIST 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 STDATA 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

- 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 11. 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 MVS Diagnosis: Procedures*.

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:

```
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 49 on page 342 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"
1
EZB1057I Loaded translation table from "TCP31S.STANDARD.TC PXLBIN".
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
2
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
3
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
4
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
5
EZB1012I Receiving ACK  Port Number=721. Remote IP Addr=9.67.113.60
5
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
6
EZB1001I Send data starts at 96155 18:52:55  Port Number=721. Remote IP Addr=9.67.113.60
6
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
7
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
8
EZB1013I ReceiveACK: TRUE for byte value 00 Port Number=721. Remote IP Addr=9.67.113.60
9
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 49. Example of LPR trace output (Part 1 of 2)

```
10
EZB1011I Queuing control line "fdfA827MVSA"
EZB1011I Queuing control line "UdfA827MVSA"
EZB1011I Queuing control line "NTCPUSR4.SOTO.FILES/LPCONFIG"
11
EZB0916I Sending command 2 argument: 153 cfA827MVSA 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
12
EZB1013I ReceiveACK: TRUE for byte value 00 Port Number=721. Remote IP Addr=9.67.113.60
13
EZB1017I Control data sent 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
EZB1012I Receiving ACK Port Number=721. Remote IP Addr=9.6 7.113.60
14
EZB1013I ReceiveACK: TRUE for byte value 00  Port Number=721. Remote IP Addr=9.67.113.60
15
EZB1018I Control file sent  Port Number=721. Remote IP Addr=9.67.113.60
```

Figure 49. Example of LPR trace output (Part 2 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.

11 Specifies the byte size (153) and the name of the control file (cfA827MVSA) that was sent.

12 Indicates that the LPD server received the command and expected the control file to be sent.

13 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.

14 `TRUE (00)` indicates that the client received an acknowledgment from the LPD server that the control file was received.

15 Confirms that the control file was sent to the LPD server. The job was then terminated.

Figure 50 on page 345 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 55 on page 354.) 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
1
EZB0915I Begin "LPR" to printer "TIANNA" at host "9.67.113.6  0"
EZB1057I Loaded translation table from "TCP31S.STANDARD.TCPXLBIN".
EZB0920I Requesting TCP/IP service at 96155 19:22:15
EZB0921I Granted TCP/IP service at 96155 19:22:15
EZB0922I Resolving 9.67.113.60 at 96155 19:22:15
EZB0924I Host 9.67.113.60 name resolved to 9.67.113.60 at 96155 19:22:15
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
:
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"
EZB1011I Queuing control line "PTCPUSR4"
EZB1011I Queuing control line "JTCPUSR4.TEST"
EZB1011I Queuing control line "CMVSA.TCP.RALEIGH.IBM.COM"
EZB1011I Queuing control line "LTCPUSR4"
4
EZB1011I Queuing control line "xdfA947MVSA"
EZB1011I Queuing control line "UdfA947MVSA"
EZB1011I Queuing control line "NTCPUSR4.TEST"
EZB0916I Sending command 2 argument: 122 cfA947MVSA  Port Number = 721. Remote IP Addr = 9.67.113.60
EZB0917I Command successfully sent  Port Number = 721. Remote IP Addr = 9.67.113.60
:
5
EZB1018I Control file sent  Port Number = 721. Remote IP Ad dr = 9.67.113.60
```

Figure 50. Example of LPR trace with filter x option

Following are short descriptions of the numbered items in the trace:

1 Indicates that the print request was issued to a printer named TIANNA at IP address 9.67.113.60.

2 Indicates that the data file was sent. The error was not recognized until the LPD server tried to process the print job. (See Figure 55 on page 354.)

3 Indicates control commands sent to the LPD server. For details about these commands, refer to RFC1179.

4 Represents the name of the data file. The character string `xdf` indicates that the x filter was used.

5 Indicates that the control file was sent to the LPD server. The job was then terminated.

Figure 51 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 51. 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 52 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.

```
1
EZB0988I PostScript program is 635 bytes
EZB0915I Begin "LPR" to printer "TIANNA" at host "9.67.113.60"
EZB1057I Loaded translation table from "TCP31S.STANDARD.TCPXLBIN".
EZB0920I Requesting TCP/IP service at 96155 19:35:12
EZB0921I Granted TCP/IP service at 96155 19:35:12
EZB0922I Resolving 9.67.113.60 at 96155 19:35:12
EZB0924I Host 9.67.113.60 name resolved to 9.67.113.60 at 96155 19:35:12
EZB0925I TCP/IP turned on.
EZB0926I Host "MVSA" Domain "TCP.RALEIGH.IBM.COM" TCPIP Service Machine TCP31S
2
EZB0927I Trying to open with local port 721 to foreign host a ddress 9.67.113.60
EZB0927I Trying to open with local port 722 to foreign host address 9.67.113.60
EZB0927I Trying to open with local port 723 to foreign host address 9.67.113.60
EZB0927I Trying to open with local port 724 to foreign host address 9.67.113.60
3
EZB0928I Connection open from local port 724 to foreign host  address 9.67.113.60
4
EZB0961I Control file name is cfA111MVSA
EZB0962I Data file name is dfA111MVSA  Port Number = 724.   Remote I P Addr = 9.67.113.60
EZB0916I Sending command 2 argument: TIANNA  Port Number = 724.   Remote IP Addr = 9.67.113.60
:
EZB1009I Data file sent.  Port Number = 724.   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>TEST"
EZB1011I Queuing control line "CMVSA.TCP.RALEIGH.IBM.COM"
EZB1011I Queuing control line "LTCPUSR4"
5
EZB1011I Queuing control line "fdfA111MVSA"
EZB1011I Queuing control line "UdfA111MVSA"
EZB1011I Queuing control line "NTCPUSR4.TEST"
EZB0916I Sending command 2 argument: 122 cfA111MVSA  Port Number = 7 24.
```

Figure 52. Example of LPR trace with JNUM, LANDSCAPE, and TRACE options

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*.

```
%!PS-Adobe-2.0
    614 25 translate 90 rotate .88 .76 scale /n 1 def /fs 10 def /ls 11. 2 def /ld l
```

Figure 53 is a sample of LPR trace output for the following command with the XLATE option:

```
LPR test (p TIANNA h MVSA trace xlate GXS
```

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
EZB1057I Loaded translation table from "TCPUSR4.GXS.TCPXLBIN" .
EZB0920I Requesting TCP/IP service at 96155 20:04:14
EZB0921I Granted TCP/IP service at 96155 20:04:15
2
EZB0922I Resolving MVSA at 96155 20:04:15
3
EZB0924I Host MVSA name resolved to 9.67.113.60 at 96155 20:0 4:17
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
4
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 53. 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 54 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 00005B70
EZB0679I Allocated ObeyBlock at 00005B60
EZB0679I Allocated ObeyBlock at 00005B50
EZB0628I Allocated PrinterBlock at 000058C0
EZB0629I    prt1 added.
EZB0641I Service prt1 defined with address
EZB0628I Allocated PrinterBlock at 00005630
EZB0629I    PRT1 added.
EZB0641I Service PRT1 defined with address
1
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 000B1D40
EZB0629I    njesoto added.
EZB0641I Service njesoto defined with address
EZB0628I Allocated PrinterBlock at 000B1AB0
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 000B1820
EZB0629I    POST added.
```

Figure 54. Example of LPD trace specified with the DEBUG option (Part 1 of 5)

```
2
EZB0686I Host "9.67.105.55" resolved to 9.67.105. 55.  Printer name is "LPT2".
2
EZB0641I Service POST defined with address
EZB0697I    ...End of Printer chain...
EZB0626I Allocated ConnectionBlock at 00147E08
3
EZB0627I Passive open on port 515
EZB0705I 06/03/96 18:49:15
EZB0834I Ready
4
EZB0789I GetNextNote with ShouldWait of TRUE
⋮
5
EZB0790I GetNextNote returns.  Connection 1 NotificationConnection state changed (8681)
5
EZB0779I New connection state Open (8673) on connection 1 with reason OK.
5
EZB0782I Connection open.  Reading command.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0711I New command 2 data "2".
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification FSend response (8692)
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
6
EZB0754I New subcommand 3 operands "7434 dfA827MV SA".
EZB0723I Allocated StepBlock at 000B1320
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns. Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Trying to open (8676) on connection 2 with reason OK.
EZB0626I Allocated ConnectionBlock at 0015BE08
7
EZB0627I Passive open on port 515
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Open (8673) on connection 2 with reason OK.
EZB0782I Connection open.  Reading command.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification FSend response (8692)
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 2 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 2
EZB0711I New command 4 data "4".
EZB0708I FSend of response sent
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 2 Notification FSend response (8692)
EZB0763I Closing connection 2
EZB0789I GetNextNote with ShouldWait of TRUE
```

Figure 54. Example of LPD trace specified with the DEBUG option (Part 2 of 5)

```
EZB0790I GetNextNote returns.  Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Receiving only (8674) on connection 2 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification FSend response (8692)
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
 8
EZB0754I New subcommand 2 operands "153 cfA827MVS A".
EZB0723I Allocated StepBlock at 000B1168
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification FSend response (8692)
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Connection closing (8670) on connection 2 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification FSend response (8692)
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Connection state changed (8681)
EZB0779I New connection state Sending only (8675) on connection 1 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification FSend response (8692)
EZB0763I Closing connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Connection state changed (8681)
EZB0779I New connection state Connection closing (8670) on connection 1 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Connection state changed (8681)
EZB0779I New connection state Nonexistent (8672) on connection 1 with reason OK.
EZB0772I End Connection 1 for OK.
```

Figure 54. Example of LPD trace specified with the DEBUG option (Part 3 of 5)

```
 9
EZB0776I Released StepBlock at 000B1320
 9
EZB0719I Allocated JobBlock at 00147798
 9
EZB0723I Allocated StepBlock at 000B1320
 10
EZB0716I Job 827 received prt1 MVSA
 11
EZB0734I Job 827 added to work queue
 12
EZB0716I Job 827 scheduled prt1 MVSA
EZB0776I Released StepBlock at 000B1168
EZB0777I Released ConnectionBlock at 0014AE08
EZB0824I ProcessWork starting on job queue
 13
EZB0731I Work Queue start
 13
EZB0732I $          827 JOBstartPRINTING
EZB0733I    Work Queue end
EZB0825I    Job 827 for prt1 dispatched in state JOBstartPRINTING
EZB0716I Job 827 printing prt1 MVSA
EZB0827I ProcessWork end with queue
EZB0731I     Work Queue start
 14
EZB0732I $          827 JOBcontinuePRINTING
EZB0733I     Work Queue end
EZB0789I GetNextNote with ShouldWait of FALSE
EZB0824I ProcessWork starting on job queue
EZB0731I     Work Queue start
EZB0732I $        827 JOBcontinuePRINTING
EZB0733I     Work Queue end
EZB0825I     Job 827 for prt1 dispatched in state JOBcontinuePRINTING
       flpNewBlock: State first call IsAtEof FALSE
 15
       flpNewBlock: State build      IsAtEof FALSE
       flpNewBlock: State check last IsAtEof FALSE
 :
       flpNewBlock: State check last IsAtEof FALSE
       flpNewBlock: State build      IsAtEof FALSE
 :
EZB0825I    Job 827 for prt1 dispatched in state JOBcontinuePRINTING
 :
       flpNewBlock: State build      IsAtEof  TRUE
       flpNewBlock: State check last IsAtEof  TRUE
EZB0827I ProcessWork end with queue
EZB0731I       Work Queue start
EZB0732I $        827 JOBcontinuePRINTING
EZB0733I       Work Queue end
EZB0789I GetNextNote with ShouldWait of FALSE
EZB0824I ProcessWork starting on job queue
```

Figure 54. Example of LPD trace specified with the DEBUG option (Part 4 of 5)

```
EZB0731I       Work Queue start
EZB0732I $        827 JOBcontinuePRINTING
EZB0733I       Work Queue end
EZB0825I    Job 827 for prt1 dispatched in state JOBcontinuePRINTING
EZB0827I ProcessWork end with queue
EZB0731I       Work Queue start
EZB0732I $        827 JOBfinishPRINTING
EZB0733I       Work Queue end
EZB0789I GetNextNote with ShouldWait of FALSE
EZB0824I ProcessWork starting on job queue
EZB0731I       Work Queue start
 16
EZB0732I $        827 JOBfinishPRINTING
EZB0733I       Work Queue end
EZB0825I    Job 827 for prt1 dispatched in state JOBfinishPRINTING
 17
EZB0716I Job 827 sent prt1 MVSA
 17
EZB0769I Job 827 removed from work queue
EZB0751I Released StepBlock at 000B1320
 17
EZB0716I Job 827 purged prt1 MVSA
EZB0771I Released JobBlock at 00147798
 18
EZB0827I ProcessWork end with queue
EZB0731I       Work Queue start
EZB0733I       Work Queue end
 19
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 0 Notification Connection state changed (8681)
 20
EZB0779I New connection state Nonexistent (8672)  on connection 0 with reason OK.
 20
EZB0772I End Connection 0 for OK.
EZB0777I Released ConnectionBlock at 00147E08
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Nonexistent (8672) on connection 2 with reason OK.
EZB0772I End Connection 2 for OK.
EZB0777I Released ConnectionBlock at 0014DE08
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 3 Notification Connection state changed (8681)
EZB0779I New connection state Trying to open (8676) on connection 3 with reason OK.
EZB0626I Allocated ConnectionBlock at 00147E08
EZB0627I Passive open on port 515
EZB0789I GetNextNote with ShouldWait of TRUE
:
 21
EZB0790I GetNextNote returns.  Connection -48 Notification Other external interrupt received (8688)
 21
EZB0622I Terminated by external interrupt
```

Figure 54. Example of LPD trace specified with the DEBUG option (Part 5 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.

5 Indicates that a connection was opened for an incoming LPR client and that the LPD server was receiving a command from that client.

6 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.

7 Indicates that the LPD server had a passive open connection on the restricted LPD port, 515.

8 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 df*x*. Control files use the naming convention cf*x*.

9 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.

10 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.

11 Indicates that job 827 was added to this print queue. The LPD server maintains a work queue of jobs.

12 Indicates that job 827 was scheduled to be spooled to the output queue.

13 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.

14 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.

15 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.

16 Indicates that all data was processed and placed in the output queue.

17 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.

18 Indicates that the LPD server completed the jobs in that queue and scans the work queue again.

19 Indicates that the LPD server was waiting for more work to do.

20 Indicates that the LPR-to-LPD connection was closed normally.

21 Indicates that someone stopped the LPD server normally.

Figure 55 on page 354 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 50 on page 345 for the corresponding LPR trace output.)

```
EZB0831I IBM MVS LPD Version V2R10 on 05/05/98 at 19:21:46
EZB0832I
EZB0621I LPD starting with port 515
⋮
EZB0628I Allocated PrinterBlock at 000053A0
EZB0629I    TIANNA added.
EZB0641I Service TIANNA defined with address
⋮
EZB0627I Passive open on port 515
EZB0705I 06/03/96 19:21:47
EZB0834I Ready
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.Connection 0 Notification Connection state changed(8681)
EZB0779I New connection state Trying to open (8676) on connection 0 with reason OK.
EZB0626I Allocated ConnectionBlock at 0014AE08
EZB0627I Passive open on port 515
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.Connection 0 Notification Connection state changed(8681)
EZB0779I New connection state Open (8673) on connection 0 with reason OK.
EZB0782I Connection open.  Reading command.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0627I Passive open on port 515
⋮
 1
EZB0754I New subcommand 3 operands "333819 dfA947MV SA".
⋮
 2
EZB0754I New subcommand 2 operands "122 cfA947MVSA" .
⋮
EZB0776I Released StepBlock at 000B1438
EZB0719I Allocated JobBlock at 00147798
EZB0723I Allocated StepBlock at 000B1438
 3
EZB0716I Job 947 received TIANNA MVSA
 3
EZB0734I Job 947 added to work queue
 3
EZB0716I Job 947 scheduled TIANNA MVSA
EZB0776I Released StepBlock at 000B1280
EZB0777I Released ConnectionBlock at 0014AE08
EZB0824I ProcessWork starting on job queue
EZB0731I       Work Queue start
EZB0732I $        947 JOBstartPRINTING
EZB0733I       Work Queue end
EZB0825I    Job 947 for TIANNA dispatched in state JOBstartPRINTING
EZB0716I Job 947 printing TIANNA MVSA
```

Figure 55. Example of an LPD server trace of a failing job (Part 1 of 2)

```
4
EZB0801I Filter "x" not supported.  Job abandoned.
EZB0827I ProcessWork end with queue
EZB0731I       Work Queue start
EZB0732I $        947 JOBfinishPRINTING
EZB0733I       Work Queue end
EZB0789I GetNextNote with ShouldWait of FALSE
EZB0790I GetNextNote returns.Connection 0 Notification Connection state changed(8681)
EZB0779I New connection state Connection closing (8670) on connection 0 with reason OK.
EZB0824I ProcessWork starting on job queue
EZB0731I       Work Queue start
EZB0732I $        947 JOBfinishPRINTING
EZB0733I       Work Queue end
5
EZB0825I    Job 947 for TIANNA dispatched in state  JOBfinishPRINTING
EZB0716I Job 947 sent TIANNA MVSA
6
EZB0769I Job 947 removed from work queue
EZB0751I Released StepBlock at 000B1438
7
EZB0716I Job 947 purged TIANNA MVSA
EZB0771I Released JobBlock at 00147798
EZB0827I ProcessWork end with queue
EZB0731I       Work Queue start
EZB0733I       Work Queue end
:
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection -48 Notification Other external interrupt received (8688)
EZB0622I Terminated by external interrupt
```

Figure 55. 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 56 on page 357 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.

```
1
EZB0831I IBM MVS LPD Version V2R10
 on 05/05/98 at 19 :50:58
EZB0832I
EZB0621I LPD starting with port 515
EZB0679I Allocated ObeyBlock at 00005B70
EZB0679I Allocated ObeyBlock at 00005B60
EZB0679I Allocated ObeyBlock at 00005B50
EZB0628I Allocated PrinterBlock at 000058C0
EZB0629I    prt1 added.
EZB0641I Service prt1 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 000B1D40
EZB0629I    njesoto added.
EZB0641I Service njesoto defined with address
EZB0628I Allocated PrinterBlock at 000B1AB0
EZB0629I    SOTO added.
2
EZB0686I Host "9.37.34.39" resolved to 9.37.34.39.   Printer name is "lpt1".
3
EZB0641I Service SOTO defined with address
EZB0628I Allocated PrinterBlock at 000B1820
EZB0629I    POST added.
EZB0686I Host "9.67.105.55" resolved to 9.67.105.55.  Printer name is "LPT2".
EZB0641I Service POST defined with address
EZB0697I    ...End of Printer chain...
EZB0626I Allocated ConnectionBlock at 00147E08
EZB0627I Passive open on port 515
EZB0705I 06/05/96 19:50:00
EZB0834I Ready
EZB0789I GetNextNote with ShouldWait of TRUE
⋮
EZB0782I Connection open.  Reading command.
EZB0799I Reading additional data on 1
⋮
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
4
EZB0754I New subcommand 3 operands "14221 dfA502MVS 7".
EZB0723I Allocated StepBlock at 000B1438
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0789I GetNextNote with ShouldWait of TRUE
⋮
EZB0799I Reading additional data on 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 1
```

Figure 56. Example of an LPD server trace for a remote print request (Part 1 of 3)

```
5
  19:50:48 EZB0754I New subcommand 2 operands "134 cfA502MVS7" .
     19:50:48 EZB0723I Allocated StepBlock at 000B1280
     19:50:49 EZB0789I GetNextNote with ShouldWait of TRUE
:
6
EZB0763I Closing connection 1
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Connection state changed (8681)
EZB0779I New connection state Connection closing (8670) on connection 1 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 1 Notification Connection state changed (8681)
EZB0779I New connection state Nonexistent (8672) on connection 1 with reason OK.
EZB0772I End Connection 1 for OK.
EZB0719I Allocated JobBlock at 00147798
7
EZB0716I Job 502 received SOTO MVS7
7
EZB0734I Job 502 added to work queue
7
EZB0716I Job 502 scheduled SOTO MVS7
EZB0777I Released ConnectionBlock at 0014AE08
EZB0824I ProcessWork starting on job queue
EZB0731I Work Queue start
8
EZB0732I $502 JOBstartSENDING
EZB0733I Work Queue end
EZB0825I Job 502 for SOTO dispatched in state JOBstartSENDING
EZB0626I Allocated ConnectionBlock at 0014AE08
9
EZB0820I Trying to open with local port 721
9
EZB0716I Job 502 opening SOTO MVS7
10
EZB0769I Job 502 removed from work queue
EZB0827I ProcessWork end with queue
EZB0731I Work Queue start
EZB0733I Work Queue end
EZB0789I GetNextNote with ShouldWait of TRUE

:
EZB0790I GetNextNote returns.Connection 1 Notification Connection state changed(8681)
11
EZB0779I New connection state Nonexistent(8672) on connection 1 with reason
      Foreign host did not respond within OPEN
11
EZB0772I End Connection 1 for Foreign host
did not respond within OPEN timeout (8560).
EZB0705I 06/05/96 19:52:22
12
EZB0773I Connection 1 terminated because "Foreign host did not respond within OPEN timeout (8560)"
```

Figure 56. Example of an LPD server trace for a remote print request (Part 2 of 3)

```
13
EZB0744I 748656 HELO MVS7.tcp.raleigh.ibm.com
13
EZB0744I 748656 MAIL FROM:<LPDSRV3@MVSA>
13
EZB0744I 748656 RCPT TO:<TCPUSR4@MVS7.tcp.raleigh.  ibm.com>
13
EZB0744I 748656 DATA
13
EZB0744I 748656 To:<TCPUSR4@MVS7.tcp.raleigh.ibm.com>
13
EZB0744I 748656
13
EZB0744I 748656 Your job to print the files "TCPUSR 4.LPD.CONFIG" on SOTO at MVSA has failed for
13
EZB0744I 748656 this reason:  Remote connection
terminated (Foreign host did not respond within
13
EZB0744I 748656 OPEN timeout (8560)).
13
EZB0744I 748656 .
EZB0751I Released StepBlock at 000B1438
EZB0751I Released StepBlock at 000B1280
14
EZB0716I Job 502 purged SOTO MVS7
EZB0771I Released JobBlock at 00147798
EZB0777I Released ConnectionBlock at 0014AE08
EZB0789I GetNextNote with ShouldWait of TRUE
15
EZB0790I GetNextNote returns.  Connection 2 Notification Connection state changed (8681)
EZB0779I New connection state Nonexistent (8672) on connection2 with reason OK.
EZB0772I End Connection 2 for OK.
EZB0777I Released ConnectionBlock at 0014DE08
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 0 Notification Connection
state changed (8681)
EZB0779I New connection state Trying to open (8676) on connection0 with reason OK.
EZB0626I Allocated ConnectionBlock at 00147E08
EZB0627I Passive open on port 515
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 0 Notification Connection state changed (8681)
EZB0779I New connection state Open (8673) on connection 0 with reason OK.
EZB0782I Connection open.  Reading command.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 0 Notification Data delivered (8682)
EZB0767I Timer cleared for connection 0
EZB0711I New command 4 data "4".
EZB0708I FSend of response sent
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 0 Notification FSend response (8692)
EZB0763I Closing connection 0
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection 0 Notification Connection state changed (8681)
EZB0779I New connection state Receiving only (8674) on connection0 with reason OK.
EZB0789I GetNextNote with ShouldWait of TRUE
EZB0790I GetNextNote returns.  Connection -48 Notification Other external interrupt received (8688)
EZB0622I Terminated by external interrupt
```

Figure 56. Example of an LPD server trace for a remote print request (Part 3 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.

2 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.

3 Indicates that the name SOTO was defined on the SERVICE statement for the remote printer, lpt1, which had the address 9.37.34.39.

4 Indicates the byte size and the name of the data file sent from the LPR client on host MVS7.

5 Indicates the byte size and name of control file sent from the LPR client on host MVS7.

6 Indicates that the connection between the LPR client and the LPD server was closing, after the server received the data and control files.

7 Indicates that the print job was received, placed in the LPD print queue, represented by SOTO, and scheduled to be sent to its destination.

8 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`.

9 Indicates that the LPD server, acting as a client, was opening a connection to the remote printer using local port 721.

10 Indicates that the LPD server removed the job from its work queue.

11 Indicates that the connection to the remote server timed out.

12 Indicates that the remote server did not respond to the request to open.

13 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.

14 Indicates that the print job was completely purged.

15 Describes additional activity between the LPD server and other clients.

Chapter 12. Diagnosing File Transfer Protocol (FTP) problems

This chapter describes how to diagnose problems with the z/OS Communications Server FTP server and FTP client. If, after reading this chapter, 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 388 and "Documenting FTP client problems" on page 410.

| This chapter 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 362
- "Error exit codes" on page 362
- "Name considerations for z/OS UNIX FTP" on page 362
- "Translation and data conversion support" on page 363
- "DB2 query support" on page 365
- "JES support" on page 367
- "Common z/OS UNIX FTP problems" on page 369
- "Diagnosing FTP server problems with traces" on page 380
- "Documenting server problems" on page 388

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 IFAPRD*xx* 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 HFS 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 8characters.
  - 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.

HFS naming conventions

Guidelines: The following list describes some naming conventions you should know about when using HFS files with the z/OS UNIX FTP server:

- The HFS 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 19 on page 366 and Table 20 on page 367 show some symptoms and possible causes of SQL problems. Table 19 on page 366 shows problems that generate a reply beginning with 55x.

Table 19. SQL problems generating 55x replies (FTP Server)

Reply	Output file	Possible causes
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 <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 FTP.DATA does not contain a DB2PLAN statement to specify this planname. • 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 20 shows other SQL problems.

Table 20. Other SQL problems (FTP Server)

Problem	Possible causes
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 380 for information about activating FTP syslog tracing.

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 HFS files
- Deallocation of MVS data sets and HFS 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:

- 0012

  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`.

- 0028

  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 HFS 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:

  ```
  RA0nnn 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 ACCESSERRRORMSGS 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 HFS, 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 an HFS directory, ensure that the STARTDIRECTORY HFS 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 380 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 Service System Secure Sockets Layer Programming*, SC24-5901. This document describes how system SSL works and also contains a chapter 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 FTP.DATA 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 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 FTP.DATA 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()
tpAuth: environment initialization complete
uthClient: secure_socket_open()
uthClient: cipherspecs = 0102
uthClient: secure_socket_init()
uthClient: 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 381 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 380 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 on page 378. 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 HFS 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, (for the 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.

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” 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 383 and the *z/OS Communications Server: IP System Administrator’s Commands* for details about the MODIFY operator command.
- See “Stop tracing” on page 382, “Tracing activity for one user” on page 383, and the *z/OS Communications Server: IP User’s Guide and Commands* for details about the SITE 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, rexecd, rshd
 # 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 HFS 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 HFS 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 383.

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 383. 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:

```
EZB.FTP.systemname.ftpdaemonname.SITE.DEBUG
```

For example, if the procedure FTPD is used to start the server on system MVS164, the profile name is:

```
EZB.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 FTP.DATA, 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 FTP.DATA.

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:BA98:7654:3210:FEDC:BA98:7654:3210))`. This activates the JES trace for a client whose IP address is FEDC:BA98:7654:3210:FEDC:BA98:7654:3210.

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 `192.48.32.00` 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:FEDC:BA98:7654:3210))`. This activates all file services extended traces for a client whose IP address is FEDC:BA98:7654:3210:FEDC:BA98:7654:3210.

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 `192.48.32.00` 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,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
  - 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 380 for information about collecting FTP server traces)
- Helpful, but not essential
  - FTP client output
  - Server FTP.DATA data set
  - TCPIP.DATA data set
  - PROFILE.TCPIP data set
  - ETC.SERVICES data set
  - The output from the STAT command issued to the server
  - If applicable, sample data to re-create the problem

FTP client

This section describes the following topics:

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 Communications Server: IP User's Guide and Commands*, *z/OS Communications Server: IP Configuration Guide*, and *z/OS Communications Server: IP Configuration Reference*.

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 HFS files. For more information, see “Name considerations for z/OS UNIX FTP” on page 362.

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 LOADDBCS 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 MBDATACONN with a statement in the FTP.DATA file or with the LOCSITE MBDATACONN subcommand to name the code pages for the multibyte data transfer. If you attempt an ASCII data transfer with ENCODING=MBCS and no MBDATACONN 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 an HFS 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 HFS 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 an HFS 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
(02) >>> TYPE A
(03) 200 Representation type is Ascii NonPrint
2 (04) Command:  site sbd=(ISO8859-1,ISO8859-1)
(05) >>> SITE sbd=(ISO8859-1,ISO8859-1)
(06) 200 Site command was accepted
3 (07) Command:  put a afile
(08) >>> PORT 9,67,113,57,4,121
(09) 200 Port request OK.
(10) >>> STOR afile
(11) 125 Storing data set /u/user33/tagging2/afile
(12) 250 Transfer completed successfully.
(13) 200 bytes transferred in 0.070 seconds.  Transfer rate 2.86 Kbytes/sec.
4 (14) Command:  site sbd=(IBM-1047,ISO8859-1)
(15) >>> SITE sbd=(IBM-1047,ISO8859-1)
(16) 200 Site command was accepted
5 (17) Command:  put a efile
(18) >>> PORT 9,67,113,57,4,122
(19) 200 Port request OK.
(20) >>> STOR efile
(21) 125 Storing data set /u/user33/tagging2/efile
(22) 250 Transfer completed successfully.
(23) 200 bytes transferred in 0.005 seconds.  Transfer rate 40.00 Kbytes/sec.
6 (24) Command:  site sbd=FTP_STANDARD_TABLES
(25) >>> SITE sbd=FTP_STANDARD_TABLES
(26) 200 Site command was accepted
7 (27) Command:  put a ufile
(28) >>> PORT 9,67,113,57,4,123
(29) 200 Port request OK.
(30) >>> STOR ufile
(31) 125 Storing data set /u/user33/tagging2/ufile
(32) 250 Transfer completed successfully.
(33) 200 bytes transferred in 0.005 seconds.  Transfer rate 40.00 Kbytes/sec.
8 (34) Command:  ls -T
(35) >>> PORT 9,67,113,57,4,124
(36) 200 Port request OK.
(37) >>> NLST -T
(38) 125 List started OK
9 (39) t ISO8859-1   T=on  afile
10 (40) t IBM-1047    T=on  efile
11 (41) - untagged      T=off ufile
```

```
   (42) 250 List completed successfully.
12 (43) Command:  get afile
   (44) >>> PORT 9,67,113,57,4,125
   (45) 200 Port request OK.
   (46) >>> RETR afile
13 (47) 125-Tagged ASCII file translated with current data connection translation table
   (48) 125 Sending data set /u/user33/tagging2/afile
   (49) 250 Transfer completed successfully.
   (50) 190 bytes transferred in 0.005 seconds.  Transfer rate 38.00 Kbytes/sec.
   (51) Command:  get efile
   (52) >>> PORT 9,67,113,57,4,126
   (53) 200 Port request OK.
   (54) >>> RETR efile
13 (55) 125-Tagged EBCDIC file translated with current data connection translation table
   (56) 125 Sending data set /u/user33/tagging2/efile
   (57) 250 Transfer completed successfully.
   (58) 200 bytes transferred in 0.005 seconds.  Transfer rate 40.00 Kbytes/sec.
   (59) Command:  get ufile
   (60) >>> PORT 9,67,113,57,4,127
   (61) 200 Port request OK.
   (62) >>> RETR ufile
14 (63) 125 Sending data set /u/user33/tagging2/ufile
   (64) 250 Transfer completed successfully.
   (65) 200 bytes transferred in 0.005 seconds.  Transfer rate 40.00 Kbytes/sec.
15 (66) Command:  site sbd=(IBM-1047,ISO8859-1)
   (67) >>> SITE sbd=(IBM-1047,ISO8859-1)
   (68) 200 Site command was accepted
   (69) Command:  get afile
   (70) >>> PORT 9,67,113,57,4,128
   (71) 200 Port request OK.
   (72) >>> RETR afile
16 (73) 125-Tagged ASCII file translated with table built using file system cp=ISO8859-1,
              network transfer cp=ISO8859-1
   (74) 125 Sending data set /u/user33/tagging2/afile
   (75) 250 Transfer completed successfully.
   (76) 190 bytes transferred in 0.005 seconds.  Transfer rate 38.00 Kbytes/sec.
   (77) Command:  get efile
   (78) >>> PORT 9,67,113,57,4,129
   (79) 200 Port request OK.
   (80) >>> RETR efile
17 (81) 125-Tagged EBCDIC file translated with table built using file system cp=IBM-1047,
              network transfer cp=ISO8859-1
   (82) 125 Sending data set /u/user33/tagging2/efile
   (83) 250 Transfer completed successfully.
   (84) 200 bytes transferred in 0.005 seconds.  Transfer rate 40.00 Kbytes/sec.
18 (85) Command:  ebcdic
   (86) >>> TYPE E
   (87) 200 Representation type is Ebcdic NonPrint
   (88) Command:  get afile
   (89) >>> PORT 9,67,113,57,4,142
   (90) 200 Port request OK.
   (91) >>> RETR afile
19 (92) 557 File contains ASCII data - enter TYPE A command before entering RETR command
   (93) Command:  get efile
   (94) >>> PORT 9,67,113,57,4,143
   (95) 200 Port request OK.
   (96) >>> RETR efile
20 (97) 125 Sending data set /u/user33/tagging2/efile
   (98) 250 Transfer completed successfully.
   (99) 190 bytes transferred in 0.005 seconds.  Transfer rate 38.00 Kbytes/sec.
```

Notes:

1 Change the data type to ASCII.

2 Site command requests a file system code page ISO8859-1, an ASCII code page.

3 Put a file and name it afile.

4 Site command requests a file system code page IBM-1047, an EBCDIC code page.

5 Put a file and name it efile.

6 Site command requests standard FTP translation tables.

7 Put a file and name it ufile.

8 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.

9 afile is a tagged file. Its file system code page is IS08859-1. It is a Text file.

10 efile is a tagged file. Its file system code page is IBM-1047. It is a Text file.

11 ufile is an untagged file. It is not a Text file.

12 Retrieve afile, which is a tagged file.

13 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**).

14 Since this is an untagged file, no indication is needed about the tables used.

15 Specify translation tables with a file system code page and a network transfer code page.

16 The code page of the tagged ASCII file is used with the network transfer code page to translate the data in the file.

17 The code page of the tagged EBCDIC file is used with the network transfer code page to translate the data in the file.

18 Change the data type to EBCDIC.

19 The 557 reply informs the client that the data type must be ASCII when the file that is tagged as ASCII is retrieved.

20 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 an HFS 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:

```
. . .
1 (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
2 (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
3 (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 **ls** 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 **ls** 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 21 and Table 22 on page 397 show some symptoms and possible causes of SQL problems

Table 21 shows problems that generate a reply beginning with 55x.

Table 21. SQL problems generating 55x replies (FTP Client)

Reply	**Output file**	**Possible causes**
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 FTP.DATA 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.

Table 21. SQL problems generating 55x replies (FTP Client) (continued)

Reply	Output file	Possible causes
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*.		

Table 22 shows other SQL problems.

Table 22. Other SQL problems (FTP Client)

Problem	Possible causes
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 check pointing, 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 l 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 41 or INFOAPAR II12014 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 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 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
: INACTIME 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 fwfriendly** 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 **locstat** 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 **locstat** 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 SUPPRESIGNOREWARNINGS 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
 CEE5101C During initialization, the OpenEdition callable service
 BPX1MSS failed. The system return code was 0000000156
     , the reason code was 0B0C00FB . The application will be
     terminated
 IKJ56641I 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 398.

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 37, “Diagnosing resolver problems,” on page 761 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. For more information about the EPSV command, see RFC 2428. For more information about the **locsite** subcommand, 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 the EPSV4 statement in FTP.DATA, refer to *z/OS Communications Server: IP Configuration Reference*.

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* 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 398.

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 ISPFSstats setting is TRUE. Use the LOCSITE ISPFStats subcommand change the ISPFSstats 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_fle: 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
  - Precise problem description, including client console, expected results, and actual results
  - Client trace output. You can use DEBUG ALL to capture all details possible.
- Helpful, but not essential
  - Output from the client **locstat** 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 13. Diagnosing z/OS UNIX Telnet daemon (otelnetd) problems

This chapter provides diagnostic information for z/OS UNIX Telnet daemon (otelnetd) and contains the following section:

- "Common problems"
- "Debug traces" on page 414

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 525). 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 Telnet server. (Check EZYTE67I messages for more information; see Figure 58 on page 415.)
- 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 23 describes options that relate to user-controlled trace information.

Table 23. Debug trace options

Option	Sub-Option	Description
-t		Internal tracing, intended to replace the DIAGNOSTICS compile option currently in place within the BSD code.
-D	authentication	Turns on authentication debugging code.
-D	encryption	Turns on encryption debugging code.
-D	options	Prints information about the negotiation of TELNET options.
-D	report	Prints the options information, plus some additional information about what processing is going on.
-D	netdata	Displays the data stream received by telnetd.
-D	ptydata	Displays the data stream written to the pty.
-D	all	Supports all options/report/ptydata/netdata/authentication/encryption options.

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 57 is a pictorial representation of this flow.

- Int—client to parent
- Ont—parent to client
- Ipt—child to parent
- Opt—parent to child

```
---------------              ---------------           ----------------
|             |     Int      |             |    Opt    |              |
|             |------------> |   Telnet    |---------->|   Telnet     |
|   Client    |   socketfd   |   parent    | masterfd  |    child     |
|             |<------------ |             |<----------|              |
|             |     Ont      |             |    Ipt    |              |
---------------              ---------------           ----------------
```

Figure 57. Trace between the Telnet client, parent, and child

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 58 on page 415 gives an example of the trace generated from **-t -D all**, generated from an AIX Telnet client. A trace explanation follows the figure.

```
 1  EZYTE29I Starting new telnet session.  catfd = 168443936
EZYTO05I Initial EBCDIC codepage = IBM-1047, ascii codepage = ISO8859-1
 2  EZYTE05I Trace 1 Debug 3d keepalive 1 kludgelinemode 0
  hostinfo 1 Registered host 0 linemode 0 multi_proc 0
telnetd: doit(Second_pass=0)
 3  EZYTE11I doit: host_name  laph.raleigh.ibm.com
 4  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)
 5  EZYTS04I STATE:send_do: send DO   AUTHENTICATION
 6  EZYTU14I UTILITY: netwrite 3  chars.
 7  EZYTU21I Ont: fffd25 ...
 8  EZYTU03I UTILITY:ttloop read 33 chars.
 9  EZYTU20I Int: fffb25fffd26fffb26fffd03fffb18fffb1ffffb ...................
EZYTU20I Int: 20fffb21fffb22fffb27fffd05 ............
telrcv() encrypt_output=0
telrcv() decrypt_input =0
 10  EZYTS05I STATE:willoption: receive WILL   AUTHENTICATION
>>>TELNETD: Sending type 2 6
>>>TELNETD: Sending type 2 2
>>>TELNETD: Sending type 2 0
utility: printsub(length=10)
 11  EZYTU17I UTILITY: send suboption
 AUTHENTICATION
 SEND
KERBEROS_V5
CLIENT|MUTUAL|ENCRYPT
KERBEROS_V5
CLIENT|MUTUAL
KERBEROS_V5
CLIENT|ONE-WAY
 12  EZYTS10I STATE:dooption: receive DO   ENCRYPT
 13  EZYTS09I STATE:send_will: send WILL   ENCRYPT
 14  EZYTS05I STATE:willoption: receive WILL   ENCRYPT
state: send_do(option=38, init=0)
 15  EZYTS04I STATE:send_do: send DO   ENCRYPT
utility: printsub(length=6)
 16  EZYTU17I UTILITY: send suboption
 ENCRYPT
 SUPPORT
DES_CFB64
DES_OFB64
 17  EZYTS10I STATE:dooption: receive DO   SUPPRESS GO AHEAD
EZYTS09I STATE:send_will: send WILL   SUPPRESS GO AHEAD
 18  EZYTS05I STATE:willoption: receive WILL   TERMINAL TYPE
state: send_do(option=24, init=0)
 19  EZYTS04I STATE:send_do: send DO   TERMINAL TYPE
EZYTS05I STATE:willoption: receive WILL   NAWS
state: send_do(option=31, init=0)
EZYTS04I STATE:send_do: send DO   NAWS
EZYTS05I STATE:willoption: receive WILL   TSPEED
state: send_do(option=32, init=0)
EZYTS04I STATE:send_do: send DO   TSPEED
EZYTS05I STATE:willoption: receive WILL   LFLOW
state: send_do(option=33, init=0)
```

Figure 58. z/OS UNIX Telnet trace using -t -D all (Part 1 of 11)

```
EZYTS04I STATE:send_do: send DO   LFLOW
EZYTS05I STATE:willoption: receive WILL   LINEMODE
state: send_do(option=34, init=0)
EZYTS04I STATE:send_do: send DO   LINEMODE
EZYTS05I STATE:willoption: receive WILL   NEW-ENVIRON
state: send_do(option=39, init=0)
EZYTS04I STATE:send_do: send DO   NEW-ENVIRON
EZYTS10I STATE:dooption: receive DO   STATUS
EZYTS09I STATE:send_will: send WILL   STATUS
>>>TELNETD: in auth_wait.
EZYTU14I UTILITY: netwrite 50  chars.
EZYTU21I Ont: fffa2501020602020200fff0fffb26fffd26fffa ...........0........
EZYTU21I Ont: 26010102fff0fffb03fffd18fffd1ffffd20fffd .....0..............
EZYTU21I Ont: 21fffd22fffd27fffb05 ..........
EZYTU03I UTILITY:ttloop read 512 chars.
EZYTU20I Int: fffa2503757365723532fff0fffa25000206006e ...........0.......>
EZYTU20I Int: 8201c6308201c2a003020105a10302010ea20703 b.F.b.B.....~....s..
EZYTU20I Int: 050020000000a38201126182010e3082010aa003 ......tb../b...b....
EZYTU20I Int: 020105a1101b0e4b52423339302e49424d2e434f ...~............(..|
EZYTU20I Int: 4da22b3029a003020103a12230201b04686f7374 (s........~......?..
EZYTU20I Int: 1b186d76736a2e7463702e72616c656967682e69 .._........./%......
EZYTU20I Int: 626d2e636f6da381c33081c0a003020101a10302 ._..?_taC.a{.....~..
EZYTU20I Int: 0101a281b30481b01cbcb5a95fd2aa72297fae13 ..sa..a....z^K..."..
EZYTU20I Int: d12bd57b08d13133a485c8a4473c585733ded76e J.N#.J..ueHu......P>
EZYTU20I Int: 711511dfae0a732e0f62329f2c1ec3bd19b35b53 ..............C]..$.
EZYTU20I Int: 58e6ede82efb0c80d525a79d26708f5f78109a85 .W.Y....N.x....^...e
EZYTU20I Int: 54e17ca09ca7a245549229aeecd1a01125338a28 ..@..xs..k...J......
EZYTU20I Int: bb58c3dd526136471c4c0f0688317ac3fefc7f83 ..C../...<..h.:C.."c
EZYTU20I Int: b808ea2bfb64f6ebb0b041cf5edd2f5e43a17f52 ......6.....;..;.~".
EZYTU20I Int: 13a299b7925d3e923df5ef18d690c523e1c35834 .sr.k).k.5..O.E..C..
EZYTU20I Int: 62213fdb0d206b894adec1e1437d9e696d6de8b3 ......,i..A..'..__Y.
EZYTU20I Int: 724c0ed1a48196308193a003020101a2818b0481 .<.Juao.al.....sa..a
EZYTU20I Int: 88f7048e7c7e2b092c0c5301f15d8ed82b92a60d h7..@=......1).Q.kw.
EZYTU20I Int: 9c0524bb740e761ad609ffff09c2a13cbcd952ef ........O....B~..R..
EZYTU20I Int: 704a5a9426a6e2607cfe0d1a3fa9969ba8d20836 ..!m.wS-@....zo.yK..
EZYTU20I Int: b8fdf73528d73abebdb7bbd7135d08e815896c62 ..7..P..]..P.).Y.i%.
EZYTU20I Int: c06e44ce1df73816969e95b77ab5b8d95d2618b9 {>...7..o.n.:..R)...
EZYTU20I Int: 7b2abbe6c6d9adab0320a73aebe5e14f9373503d #..WFR[...x..V.|l.&.
EZYTU20I Int: 18a97c34a5698c5dc56364d871939fee0193fff0 .z@.v..)E..Q.l...l.0
EZYTU20I Int: fffa2605fff0fffa260102fff0fffa1f00780032 .....0......0.......
EZYTU20I Int: fff0fffa2203010300036203 .0.........
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=10)
 20  EZYTU17I UTILITY: receive suboption
 AUTHENTICATION
 NAME
u
s
e
r
5
2
>>>TELNETD: Got NAME [user52]
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=465)
```

Figure 58. z/OS UNIX Telnet trace using -t -D all (Part 2 of 11)

```
21  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
116 27 24 109 118 115 106 46 116 99 112 46 114 97 108 101 105 103 104 46 105 98
 109 46 99 111 109 163 129 195 48 129 192 160 3 2 1 1 161 3 2 1 1 162 129 179 4
 129 176 28 188 181 169 95 210 170 114 41 127 174 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
22  telnetd: Kerberos5 identifies him as ``user52@KRB390.IBM.COM''
EZYTU14I UTILITY: netwrite 130  chars.
EZYTU21I Ont: fffa25020206036f593057a003020105a1030201 .......?........~...
EZYTU21I Ont: 0fa24b3049a003020101a2420440117ac36284cd .s........s.. .:C.d.
EZYTU21I Ont: 65384025a9ee70511777fd91aa4c367edd20162f .. .z......j.<.=....
EZYTU21I Ont: 736d40b6e61fae60d2c74c25aa610dcd10526eea ._ .W..-KG<../....>.
EZYTU21I Ont: b096d7e7ed90a06f7a595cddbe92369be741fff0 .oPX...?:.*..k..X..0
EZYTU21I Ont: fffa250202060275736572353240 4b5242333930 ............. ......
EZYTU21I Ont: 2e49424d2e434f4dfff0 ...(..|(.0
utility: printsub(length=4)
23  EZYTU17I UTILITY: receive suboption
 ENCRYPT
 REQUEST-START
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=5)
24  EZYTU17I UTILITY: receive suboption
 ENCRYPT
 SUPPORT
DES_OFB64
>>>TELNETD: He is supporting DES_OFB64 (2)
Creating new feed
utility: printsub(length=14)
25  EZYTU17I UTILITY: send suboption
 ENCRYPT
 IS
DES_OFB64
OFB64_IV 123 117 204 223 5 21 98 2
>>>TELNETD: (*ep->start)() returned 6
EZYTS17I Defer suboption negotiation
EZYTU14I UTILITY: netwrite 16  chars.
EZYTU21I Ont: fffa260002017b75ccdf05156202fff0 ......#........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)
EZYTS04I 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)
EZYTS04I STATE:send_do: send DO   XDISPLOC
state: send_do(option=39, init=1)
state: send_do(option=36, init=1)
```

Figure 58. z/OS UNIX Telnet trace using -t -D all (Part 3 of 11)

```
EZYTS04I STATE:send_do: send DO   OLD-ENVIRON
EZYTS09I STATE:send_will: send WILL   ECHO
EZYTU14I UTILITY: netwrite 12  chars.
EZYTU21I Ont: fffd26fffd23fffd24fffb01 ............
EZYTU03I UTILITY:ttloop read 47 chars.
EZYTU20I Int: 04020f05030007621c08020409421a0a02080b02 ....................
EZYTU20I Int: 150c02170d02120e02160f021110021311020012 ....................
EZYTU20I Int: 0200fff0fffd03 ...0...
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 58. z/OS UNIX Telnet trace using -t -D all (Part 4 of 11)

```
 VARIABLE
 17;
 XOFF
 VARIABLE
 19;
 FORW1
 VARIABLE
 0;
 FORW2
 VARIABLE
 0;
EZYTS10I STATE:dooption: receive DO   SUPPRESS GO AHEAD
EZYTU03I UTILITY:ttloop read 16 chars.
EZYTU20I Int: fffa26000201094321d752693162fff0 .........P.....0
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=14)
 26  EZYTU17I UTILITY: receive suboption
 ENCRYPT
 IS
DES_OFB64
OFB64_IV 9 67 33 215 82 105 49 98
CFB64: initial vector received
Initializing Decrypt stream
utility: printsub(length=6)
 27  EZYTU17I UTILITY: send suboption
 ENCRYPT
 REPLY
DES_OFB64
OFB64_IV_OK
(*ep->is)(a09abc3, 9) returned MORE_TO_DO (7)
EZYTU14I UTILITY: netwrite 8  chars.
EZYTU21I Ont: fffa26020202fff0 .......0
EZYTU03I UTILITY:ttloop read 17 chars.
EZYTU20I Int: fffa26020202fff0fffc23fffc24fffd01 .......0.........
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=6)
 28  EZYTU17I UTILITY: receive suboption
 ENCRYPT
 REPLY
DES_OFB64
OFB64_IV_OK
utility: printsub(length=5)
 29  EZYTU17I UTILITY: send suboption
 ENCRYPT
 ENC_KEYID
 0
(*ep->reply)(a09abc3, 1) returned MORE_TO_DO (4)
>>>TELNETD: encrypt_reply returned 4
 30  EZYTS08I STATE:wontoption: receive WON'T   XDISPLOC
EZYTS08I STATE:wontoption: receive WON'T   OLD-ENVIRON
EZYTS10I STATE:dooption: receive DO   ECHO
>>>TELNETD: in encrypt_wait
EZYTU14I UTILITY: netwrite 7  chars.
EZYTU21I Ont: fffa260700fff0 ......0
EZYTU03I UTILITY:ttloop read 7 chars.
EZYTU20I Int: fffa260700fff0 ......0
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=5)
```

Figure 58. z/OS UNIX Telnet trace using -t -D all (Part 5 of 11)

```
 31  EZYTU17I UTILITY: receive suboption
 ENCRYPT
 ENC_KEYID
 0
utility: printsub(length=5)
 29  EZYTU17I UTILITY: send suboption
 ENCRYPT
 DEC_KEYID
 0
EZYTU14I UTILITY: netwrite 7  chars.
EZYTU21I Ont: fffa260800fff0 ......0
EZYTU03I UTILITY:ttloop read 14 chars.
EZYTU20I Int: fffa260800fff0fffa260300fff0 ......0......0
telrcv() encrypt_output=0
telrcv() decrypt_input =0
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=5)
 31  EZYTU17I UTILITY: receive suboption
 ENCRYPT
 DEC_KEYID
 0
>>>TELNETD: Encrypt start: initial negotiation in progress (0) DES_OFB64
utility: printsub(length=5)
 32  EZYTU17I UTILITY: send suboption
 ENCRYPT
 START
>>>TELNETD: Started to encrypt output with type DES_OFB64
EZYTU14I UTILITY: netwrite 7  chars.
EZYTU21I Ont: fffa260300fff0 ......0
utility: printsub(length=5)
 33  EZYTU17I UTILITY: receive suboption
 ENCRYPT
 START
>>>TELNETD: Start to decrypt input with type DES_OFB64
utility: printsub(length=5)
EZYTU17I UTILITY: send suboption
 ENCRYPT
 REQUEST-START
>>>TELNETD: Request input to be encrypted
>>>TELNETD: Encrypt start: initial negotiation in progress (0) DES_OFB64
utility: printsub(length=5)
EZYTU17I UTILITY: send suboption
 ENCRYPT
 START
>>>TELNETD: Started to encrypt output with type DES_OFB64
telnetd: getterminaltype() auth_negotiated=1
utility: printsub(length=4)
 34  EZYTU17I UTILITY: send suboption
 TERMINAL-TYPE
  SEND
EZYTU14I UTILITY: netwrite 32  chars.
EZYTU21I Ont: b306079a3675d15d45511c7172c0579b93f4b0ac ......J).....{..l4..
EZYTU21I Ont: afdc1e09ae1fe760ed4d59b4 ......X-.(..
EZYTU03I UTILITY:ttloop read 51 chars.
EZYTU20I Int: b306079c3675d15d45571f48bb00984d8ac37f6c ......J)......q(.C"%
EZYTU20I Int: afd6c6f276ef18cfa609f445b8b52a92fb4c2a25 .OF2....w.4....k.<..
EZYTU20I Int: ca539aa8f2401960713bc5 ...y2 .-..E
telrcv() encrypt_output=0xA00C848
telrcv() decrypt_input =0xA00C6C0
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=5)
```

Figure 58. z/OS UNIX Telnet trace using -t -D all (Part 6 of 11)

```
EZYTU17I UTILITY: receive suboption
 ENCRYPT
 START
>>>TELNETD: Start to decrypt input with type DES_OFB64
EZYTS17I Defer suboption negotiation
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=13)
EZYTU17I UTILITY: receive suboption
 TERMINAL-SPEED
  IS 9600,9600
EZYTS17I 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
e
r
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
 35  EZYTE10I terminaltypeok: call tgetent (buf, xterm)
telnetd: getterminaltype() return 3
EZYTO01I Int: 75 .
EZYTO02I Ont: 49 .
EZYTO01I Int: 73 .
EZYTO02I Ont: b7 .
EZYTO01I Int: 65 .
EZYTO02I Ont: 50 &
EZYTO01I Int: 72 .
EZYTO02I Ont: 77 .
EZYTO01I Int: 35 .
EZYTO02I Ont: 91 j
EZYTO01I Int: 32 .
EZYTO02I Ont: f6 6
EZYTO01I Int: 0d .
EZYTO02I Ont: e1 .
EZYTE59I read_pw: Character ignored 0
 36  EZYTO04I lusername  = user52
telnetd: krb name: user52, user: user52
EZYTE22I herald()
 37  EZYTE26E herald: stat error  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(tnmsgs.cat,0) code 81 reason 053B006C
```

Figure 58. z/OS UNIX Telnet trace using -t -D all (Part 7 of 11)

```
h_errno N/A
telnetd: main() -y getsubopt(tSave=2137884232)
telnsave: freeTelnetSave() rc=0
EZYTO11I DInt: fffa1f00780032fff0fffa220301030003620304 ........0...........
EZYTO11I DInt: 020f05030007621c08020409421a0a02080b0215 ....................
EZYTO11I DInt: 0c02170d02120e02160f02111002131102001202 ....................
EZYTO11I DInt: 00fff0fffa2000393630302c39363030fff0fffa ..0..............0..
EZYTO11I DInt: 27000005553455201757365723532fff0 ...............0
telnetd: doit(Second_pass=1)
EZYTY02I GETPTY: open of /dev/ptyp   EDC5114I Resource busy. rsn = 020A0155
EZYTY02I GETPTY: open of /dev/ptyp   EDC5114I Resource busy. rsn = 020A0155
EZYTY05I GETPTY: slave fd = 9 , masterfd = 8
telnetd: doit() deferred_processing=1
 38  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 .
 38  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   LFLOW
EZYTO09I options(33) = 12 .
EZYTS16I STATE:willoption:deferred receive WILL   LINEMODE
EZYTU14I UTILITY: netwrite 13  chars.
EZYTU21I Ont: b7c2d637e5d127b4aeced4cbad .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
EZYTS18I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=7)
EZYTU17I UTILITY: receive suboption
 NAWS
  0 120 (120)
  0 50 (50)
EZYTS18I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=52)
```

Figure 58. 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;
EZYTS18I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=13)
EZYTU17I UTILITY: receive suboption
 TERMINAL-SPEED
  IS 9600,9600
EZYTS18I Process deferred suboption negotiation
EZYTU14I UTILITY: netwrite 0  chars.
utility: printsub(length=16)
```

Figure 58. z/OS UNIX Telnet trace using -t -D all (Part 9 of 11)

```
EZYTU17I UTILITY: receive suboption
 NEW-ENVIRON
 IS
 VAR
U
S
E
R
 VALUE
u
s
e
r
5
2
telnetd: doit() deferred_processing=0
state: send_do(option=31, init=1)
state: send_do(option=33, init=1)
telrcv() encrypt_output=0xA00C848
telrcv() decrypt_input =0xA00C6C0
state: send_do(option=0, init=1)
EZYTS04I STATE:send_do: send DO   BINARY
EZYTS07I STATE:send_dont: send DON'T   LINEMODE
EZYTU14I UTILITY: netwrite 66  chars.
EZYTU21I Ont: 2e925cb2dbf210c689e053fbad994f522421dbb7 .k*..2.Fi\..[r|.....
EZYTU21I Ont: 062ef6d290bf59f7e40600bc4c43f4eef139b405 ..6K...7U...<.4.1...
EZYTU21I Ont: a59b84b3fc185a609644499e56c69fe7790b6e7e v.d...!-o....F.X`.>=
EZYTU21I Ont: 8cdd2ede8d42 ......
utility: printsub(length=52)
EZYTU17I UTILITY: send suboption
 LINEMODE
 SLC
 SYNCH
 NOSUPPORT
 0;
 IP
 VARIABLE
|ACK|FLUSHIN|FLUSHOUT
 3;
 AO
 VARIABLE
|ACK
 15;
 AYT
 NOSUPPORT
 0;
 ABORT
 VARIABLE
|ACK|FLUSHIN|FLUSHOUT
 28;
 EOF
 VARIABLE
|ACK
 4;
 SUSP
 VARIABLE
```

Figure 58. z/OS UNIX Telnet trace using -t -D all (Part 10 of 11)

```
|ACK|FLUSHIN
 26;
 EC
 VARIABLE
|ACK
 8;
 EL
 VARIABLE
|ACK
 21;
 EW
 VARIABLE
|ACK
 23;
 RP
 VARIABLE
|ACK
 18;
 LNEXT
 VARIABLE
|ACK
 22;
 XON
 VARIABLE
|ACK
 17;
 XOFF
 VARIABLE
|ACK
 19;
 FORW1
 VARIABLE
|ACK
 0;
 FORW2
 NOSUPPORT
 0;
EZYTU14I UTILITY: netwrite 0  chars.
EZYTE66I PROTOCOL:lmodetype=4, linemode=0, uselinemode=0
 39  EZYTY08I argv_fsum(0) =  fomtlinp
EZYTY08I argv_fsum(1) =  *4OurhrEa)R0,H/h
EZYTY08I argv_fsum(2) =
EZYTY08I argv_fsum(3) =  0
EZYTY08I argv_fsum(4) =  8
EZYTY08I argv_fsum(5) =  9
EZYTY08I argv_fsum(6) =  0
EZYTY08I argv_fsum(7) =  0
EZYTY08I argv_fsum(8) =  6
EZYTY08I argv_fsum(9) =  80
EZYTY08I argv_fsum(10) = laph.raleigh.ibm.com
EZYTY08I argv_fsum(11) = xterm
EZYTY08I argv_fsum(12) =
EZYTY08I argv_fsum(13) =
EZYTY08I argv_fsum(14) =
EZYTY08I argv_fsum(15) =
EZYTY08I argv_fsum(16) = 1
EZYTY08I inherit flag =  40000000
EZYTY09I login_tty: spawnp fsumoclp 33
 40  EZYTE67I S(nfd):socketfd..ibits=00000001 obits=00000000 ebits=00000000
         S(nfd) pty..ibits=00000000 obits=00000000 ebits=00000100
 41  EZYTE68I Ept: #bytes = 4 pkcontrol(cntl) 1003
EZYTE69I PROTOCOL: cntl = 1003
EZYTE65I PROTOCOL: send IAC Data Mark.  DMARK
```

Figure 58. z/OS UNIX Telnet trace using -t -D all (Part 11 of 11)

Following are short descriptions of the numbered items in the trace:

1 EZYTE29I indicates the start of a new z/OS UNIX Telnet client session.

2 EZYTE05I indicates what options were specified in /etc/inetd.conf for z/OS UNIX Telnet.

3 EZYTE11I indicates the resolved host name (from the client).

4 EZYTE11I shows the IP address of the z/OS UNIX Telnet client.

5 EZYTS04I indicates otelnetd agrees to send and receive authentication information.

6 EZYTU14I traces netwrites (writes to the client terminal).

7 EZYTU21I traces data from parent to client; that is, z/OS UNIX Telnet to the client terminal.

8 EZYTU03I indicates the number of bytes read from the client by z/OS UNIX Telnet.

9 EZYTU20I traces data from the client to the parent (z/OS UNIX Telnet server).

10 EZYTS05I indicates the client agrees to send and receive authentication information.

11 EZYTU17I shows otelnetd requesting that the client send authentication information for Kerberos Version 5.

12 EZYTS10I indicates the client agrees to receive encrypted data.

13 EZYTS09I indicates otelnetd agrees to send encrypted data.

14 EZYTS05I indicates the client agrees to send encrypted data.

15 EZYTS04I indicates otelnetd agrees to receive encrypted data.

16 EZYTU17I shows which types of encryption otelnetd supports when receiving data.

17 EZYTS10I shows the terminal option negotiation the client has sent/received.

18 EZYTS05I shows the terminal option negotiation the client has sent/received.

19 EZYTS04I indicates the terminal negotiation options sent to the client by the z/OS UNIX Telnet server.

20 EZYTU17I shows the account name on otelnetd that the client wishes to be authorized to use.

21 EZYTU17I shows the client authentication information for Kerberos Version 5.

22 Shows the Kerberos Version 5 principal of the user logging in.

23 EZYTU17I shows the client requesting that otelnetd enable encryption as soon as the initialization is completed.

24 EZYTU17I shows which types of encryption the client supports when receiving data.

25 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.

26 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.

27 EZYTU17I shows otelnetd acknowledging receipt of the initial encryption data from the client.

28 EZYTU17I shows the client acknowledging receipt of the initial encryption data from otelnetd.

29 EZYTU17I shows otelnetd verifying its keyids.

30 EZYTS08I shows the terminal option negotiation the client has sent/received.

31 EZYTU17I shows the client verifying its keyids.

32 EZYTU17I shows all data following this command in the data stream (otelnetd to client) are encrypted using the previously negotiated method of data encryption.

33 EZYTU17I shows all data following this command in the data stream (client to otelnetd) are encrypted via the previously negotiated method of data encryption.

34 EZYTU17I traces z/OS UNIX Telnet sending terminal negotiation suboptions to the client.

35 EZYTE10I traces the call to tgetent(), which determines client terminal type.

36 EZYTO04I shows the user name with which the telnet client logged in.

37 EZYTE26E indicates no /etc/banner file was found.

38 EZYTS15I and EZYTS16I show that a state change was processed due to options/responses received from the client.

39 EZYTY08I traces the parameters passed to the spawned/forked child address space where the OMVS shell runs.

40 EZYTE67I traces the socket sets to show whether input/ibits, output/obits, or exception/ebits data has been received.

41 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 14. Diagnosing Telnet problems

This chapter describes how to diagnosis Telnet problems, and contains the following chapters:

General 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 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 447.

Telnet can run as part of the TCP/IP stack or it can run as its own procedure in its own address space. Either way, 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 41 and Chapter 6, "IPCS subcommands for TCP/IP," on page 173 for details.

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 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 TCP/IP or Telnet to ensure that a useful dump is obtained in the event of an abend.

Analysis

Refer to *z/OS MVS Diagnosis: Procedures* 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 (within TCP/IP or as its own procedure) 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 24 on page 431 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 24. Telnet login problems

Login problem	Analysis steps
No LUs available	1, 2, 6, 10, 13
OPEN failure	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
x-clock (Telnet solicitor panel)	1, 2, 3, 4, 5, 6, 7, 10
x-clock (blank screen)	1, 2, 3, 6, 7, 8, 10, 12
x-clock (application panel)	7, 8, 10
Incorrect USSMSG or DEFAULTAPPL	3, 4, 5, 6, 1, 13, 14

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*.

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 TCP/IP address space and TCP/IP Dataspace, or a dump of the Telnet address space if running Telnet as its own procedure. 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)
  ```

For information about obtaining VTAM traces, refer to *z/OS Communications Server: SNA Operation* or to *z/OS Communications Server: SNA Diagnosis Vol 2, FFST Dumps and the VIT* for your release. Instructions on obtaining a dump can be found in *z/OS MVS Diagnosis: Tools and Service Aids* 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* 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 the Telnet server 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 the Telnet server 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 the TELNET server 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 server 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 the Telnet server. Otherwise, continue with Step 8 on page 434.

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.

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 server 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 server 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* 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 server, or Telnet client code.

Table 25 lists the types of incorrect output that might be seen and the steps needed to identify the code in error.

Table 25. Incorrect output types for Telnet

Incorrect Output	Analysis Steps
Blank screen	1, 6, 7

Table 25. Incorrect output types for Telnet (continued)

Incorrect Output	**Analysis Steps**
Garbled or unreadable characters on the screen	2, 3, 4, 5, 6, 7
Incorrectly formatted screen	6, 7

See Table 25 on page 434 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 434 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 the TELNET server altered the data stream.

   If not, the problem is with the TCP/IP platform code. Otherwise, continue with Step 4 on page 433.

---

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 server code. Otherwise, continue with Step 5 on page 433.

   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. The Telnet server 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. The Telnet server 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” on page 436 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://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS for more information. If this does not provide a solution, contact the IBM Software Support Center.

---

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* 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 41.

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    70010004  12:49:06.354966   Send Data to Client
HASID..002A       PASID...002A       SASID..002A       MODID..EZBTTSND
TCB....00000000    REG14...89D37F40    USER...TCPM1011   DUCB...0000000D
CID....092552C4    SEQ.....000024BE
  ...
  ...
  ADDR...00000000  08167AB0  LEN....00000004  Number of Bytes Sent
    +0000  0000002C                             | ....             |
  ADDR...00000000  7F687950  LEN....0000002C  Data to Client
    +0000  F5C1115D  7F1D4011  40401DC8  C9D2D1F5  | 5A.)". .  .HIKJ5 |
    +0010  F6F7F0F0  C140C5D5  E3C5D940  E4E2C5D9  | 6700A ENTER USER |
    +0020  C9C44060  1D4011C1  5013FFEF            | ID -. .A&...     |
```

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(SYSTCPIP) 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 23, for more information.

Analysis

Refer to *z/OS MVS Diagnosis: Procedures* or see Chapter 3, "Diagnosing abends, loops, and hangs," on page 23 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 59 on page 444. 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 Telnet server problems" on page 430. Otherwise, continue with Step 2.

2. Was the last activity at the terminal input or output? If input, go to step 5 on page 441. If output, continue with Step 3 on page 441.

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 the host application 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. If data is in the client trace, the error needs to be diagnosed from the server host. See "Session hangs (server)" on page 432 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 443.

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. Otherwise, continue with Step 2.

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 59 on page 444 is sample output from a Telnet client trace showing part of a Telnet login to a remote host.

```
1 EZA8310I DataDelivered; # bytes: 3
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
2 EZA8306I Option neg. stuff arrives
  EZA8310I DataDelivered; # bytes: 6
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8310I DataDelivered; # bytes: 12
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8338I ord: 255 asis:
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8310I DataDelivered; # bytes: 222
3 EZA8359I Data received from TCP:
4 EZA8361I FF FD 00 FF FB 00 05 C2 11 40 40 1D E4 C5 95 A3 85 99 40 E8
  EZA8361I 96 A4 99 40 E4 A2 85 99 89 84 7A 1D C4 00 00 00 00 00 00 00
  EZA8361I 00 1D E4 11 C1 50 1D E4 D7 81 A2 A2 A6 96 99 84 7A 1D CC 00
  EZA8361I 00 00 00 00 00 00 00 1D E4 11 C1 F7 1D E4 D5 85 A6 40 97 81
  EZA8361I A2 A2 A6 96 99 84 7A 1D CC 00 00 00 00 00 00 00 00 1D E4 11
  EZA8361I C2 60 1D E4 C1 97 97 93 89 83 81 A3 89 96 95 7A 1D C4 40 40
  EZA8361I 40 40 40 40 40 40 1D E4 11 C3 F0 1D E8 C1 97 97 93 89 83 81
  EZA8361I A3 89 96 95 40 99 85 98 A4 89 99 85 84 4B 40 D5 96 40 C9 95
  EZA8361I A2 A3 81 93 93 81 A3 89 96 95 40 C4 85 86 81 A4 93 A3 40 40
  EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
  EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 00 00 00 11 40 40 05 13
  EZA8361I FF EF
5 EZA8364I z" w" B"  "UEnter Your Userid:D " "" """U"A&"UPassword:">";
  EZA8364I "U"A7"UNew password:">"U"B-"UApplication:"D
  EZA8364I       "U"C0"YApplication required. No Installation Default
  EZA8364I                                 "   "┬Q
```

Figure 59. Telnet client trace (Part 1 of 4)

```
  EZA8339I In Transparent mode, found IAC at IacOffset 0, CurrentChar is 0
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8339I In Transparent mode, found IAC at IacOffset 0, CurrentChar is 3
  EZA8345I in TelnetRead
  EZA8305I in IacNoteArrives
  EZA8306I Option neg. stuff arrives
  EZA8339I In Transparent mode, found IAC at IacOffset 214, CurrentChar is 6
  EZA8345I in TelnetRead
6 EZA8313I got USERdeliversLINE
  EZA8371I in SendData
7 EZA8380I User data is...
  EZA8381I 7D '
  EZA8381I C2 B
  EZA8381I F1 1
  EZA8381I 11 "
  EZA8381I 40
  EZA8381I D4 M
  EZA8381I E4 U
  EZA8381I E2 S
  EZA8381I C5 E
  EZA8381I D9 R
  EZA8381I F2 2
  EZA8381I 11 "
  EZA8381I C2 B
  EZA8381I 6E >
  EZA8381I E3 T
  EZA8381I E2 S
  EZA8381I D6 O
  EZA8381I 40
  EZA8381I 40
  EZA8381I 40
  EZA8381I 40
  EZA8381I 40
8 EZA8382I ; Len is 22
  EZA8310I DataDelivered; # bytes: 48
  EZA8359I Data received from TCP:
  EZA8361I 05 C1 11 5D 7F 1D 40 11 40 40 1D C8 C9 D2 D1 F5 F6 F7 F0 F0
  EZA8361I C1 40 C5 D5 E3 C5 D9 40 E4 E2 C5 D9 C9 C4 40 60 1D 40 11 C1
  EZA8361I 50 13 FF EF 01 C2 FF EF
  EZA8364I  A")"" "  "HIKJ56700A ENTER USERID -" "A&"┬Q"B"Q;
  EZA8339I In Transparent mode, found IAC at IacOffset 42, CurrentChar is 0
  EZA8345I in TelnetRead
  EZA8339I In Transparent mode, found IAC at IacOffset 2, CurrentChar is 44
  EZA8345I in TelnetRead
  EZA8313I got USERdeliversLINE
9 EZA8371I in SendData
  EZA8380I User data is...
  EZA8381I 7D '
  EZA8381I C1 A
  EZA8381I D5 N
  EZA8381I 11 "
  EZA8381I 40
  EZA8381I 5A !
  EZA8381I A4 u
  EZA8381I A2 s
  EZA8381I 85 e
  EZA8381I 99 r
  EZA8381I F3 3
  EZA8382I ; Len is 11
```

Figure 59. Telnet client trace (Part 2 of 4)

```
EZA8310I DataDelivered; # bytes: 1106
EZA8359I Data received from TCP:
EZA8361I 05 C3 11 40 40 3C 40 40 40 11 40 40 1D E8 60 60 60 60 60 60
EZA8361I 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
EZA8361I 60 60 60 60 60 40 E3 E2 D6 61 C5 40 D3 D6 C7 D6 D5 40 60 60
EZA8361I 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
EZA8361I 60 60 60 60 60 60 60 60 60 60 60 60 60 60 11 C1 50 1D E8 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 11
EZA8361I C2 60 1D E8 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 11 5B 60 1D E8 D7 C6 F1 61 D7 C6 F1 F3 40 7E 7E
EZA8361I 6E 40 C8 85 93 97 40 40 40 40 D7 C6 F3 61 D7 C6 F1 F5 40 7E
EZA8361I 7E 6E 40 D3 96 87 96 86 86 40 40 40 40 D7 C1 F1 40 7E 7E 6E
EZA8361I 40 C1 A3 A3 85 95 A3 89 96 95 40 40 40 40 D7 C1 F2 40 7E 7E
EZA8361I 6E 40 D9 85 A2 88 96 A6 11 5C F0 1D E8 E8 96 A4 40 94 81 A8
EZA8361I 40 99 85 98 A4 85 A2 A3 40 A2 97 85 83 89 86 89 83 40 88 85
EZA8361I 93 97 40 89 95 86 96 99 94 81 A3 89 96 95 40 82 A8 40 85 95
EZA8361I A3 85 99 89 95 87 40 81 40 7D 6F 7D 40 89 95 40 81 95 A8 40
EZA8361I 85 95 A3 99 A8 40 86 89 85 93 84 11 C3 F3 1D E8 C5 95 A3 85
EZA8361I 99 40 D3 D6 C7 D6 D5 40 97 81 99 81 94 85 A3 85 99 A2 40 82
EZA8361I 85 93 96 A6 7A 11 C4 E3 1D E8 D9 C1 C3 C6 40 D3 D6 C7 D6 D5
EZA8361I 40 97 81 99 81 94 85 A3 85 99 A2 7A 11 C6 D2 1D 60 40 E4 A2
EZA8361I 85 99 89 84 40 40 40 40 7E 7E 7E 6E 11 C6 E2 1D E8 E4 E2 C5
EZA8361I D9 F3 40 40 1D F0 11 C8 F2 1D 60 40 D7 81 A2 A2 A6 96 99 84
EZA8361I 40 40 7E 7E 7E 6E 11 C9 C2 1D 4C 00 00 00 00 00 00 00 00 1D
EZA8361I F0 11 4D F2 1D 60 40 C1 83 83 A3 40 D5 94 82 99 40 7E 7E 7E
EZA8361I 6E 11 4E C2 1D C8 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EZA8361I 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EZA8361I 00 00 00 00 00 00 1D F0 11 4B D2 1D 60 40 D7 99 96 83 85 84
EZA8361I A4 99 85 40 7E 7E 7E 6E 11 4B E2 1D C8 D4 E5 E2 F4 F2 F2 40
EZA8361I 40 1D F0 11 50 D2 1D 60 40 E2 89 A9 85 40 40 40 40 40 40 7E
EZA8361I 7E 7E 6E 11 50 E2 1D C8 F4 F0 F9 F6 00 00 00 1D F0 11 D2 F2
EZA8361I 1D 60 40 D7 85 99 86 96 99 94 40 40 40 7E 7E 7E 6E 11 D3 C2
EZA8361I 1D C8 00 00 00 1D F0 11 4C C2 1D 60 40 C7 99 96 A4 97 40 C9
EZA8361I 84 85 95 A3 40 40 7E 7E 7E 6E 11 4C D5 1D C8 00 00 00 00 00
EZA8361I 00 00 00 1D F0 11 C9 E2 1D 60 40 D5 85 A6 40 D7 81 A2 A2 A6
EZA8361I 96 99 84 40 7E 7E 7E 6E 11 C9 F5 1D 4C 00 00 00 00 00 00 00
EZA8361I 00 1D F0 11 D7 F3 1D E8 C5 95 A3 85 99 40 81 95 40 7D E2 7D
EZA8361I 40 82 85 86 96 99 85 40 85 81 83 88 40 96 97 A3 89 96 95 40
EZA8361I 84 85 A2 89 99 85 84 40 82 85 93 96 A6 7A 1D 60 11 D9 C7 1D
EZA8361I E8 00 11 D9 C9 1D C8 40 1D F0 60 D5 96 94 81 89 93 1D 60 11
EZA8361I D9 D7 1D E8 00 11 D9 D9 1D C8 40 1D F0 60 D5 96 95 96 A3 89
EZA8361I 83 85 1D 60 11 D9 E8 1D E8 00 11 D9 6A 1D C8 00 1D F0 60 D9
EZA8361I 85 83 96 95 95 85 83 A3 1D 60 11 D9 7A 1D E8 00 11 D9 7C 1D
EZA8361I C8 40 1D F0 60 D6 C9 C4 83 81 99 84 40 1D 60 11 D5 D2 1D 60
EZA8361I 40 C3 96 94 94 81 95 84 40 40 40 7E 7E 7E 6E 11 D5 E2 1D C8
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40
EZA8361I 1D F0 11 C7 C2 1D 7C 40 E2 85 83 93 81 82 85 93 40 40 40 40
EZA8361I 40 7E 7E 7E 6E 11 C7 D5 1D 7C 40 40 40 40 40 40 40 40 1D F0
EZA8361I 11 C9 C3 13 FF EF
```

Figure 59. Telnet client trace (Part 3 of 4)

```
EZA8364I C -    Y -------------------------------- TSO/E LOGON --
EZA8364I ------------------ A&  Y
EZA8364I
EZA8364I B- Y
EZA8364I                       |$- YPF1/PF13==> Help PF3/PF15=
EZA8364I Logoff    PA1==> Attention PA2==> Reshow *0 YYou may
EZA8364I request specific help information by entering a '?' in any
EZA8364I entry field C3 YEnter LOGON parameters below: DT YRACF LOGON
EZA8364I parameters: FK - Userid     ===> FS YUSER3  0 H2 - Password
EZA8364I ===> IB <          0 (2 - Acct Nmbr ===> +B H
EZA8364I 0".K - Procedure===>  .S  HMVS422   EZA8364I 0
      &K - Size      ===>  &S  H4096   0 K2 - Perform   ===> LB
EZA8364I H    0 <B - Group Ident  ===>  <N H 0 IS- New Passw
EZA8364I ord ===> I5 < 0 P3 YEnter an 'S' before each option
EZA8364I desired below: - RG Y  RI H 0-Nomail RP Y  RR H  0-Nonoti
EZA8364I ce - RY Y R: H  0-Reconnect - R: Y  R@ H  0-OIDcard - NK -
EZA8364I  Command   ===> NS H
EZA8364I                                       0 GB @ Seclabel
EZA8364I  ===>GN @       0 IC 'Q
EZA8339I In Transparent mode, found IAC at IacOffset 1104, CurrentChar is 0
EZA8345I in TelnetRead
EZA8313I got USERdeliversLINE
EZA8371I in SendData
```

Figure 59. Telnet client trace (Part 4 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 26 on page 448 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 26. Telnet commands from RFC 854

Command	Code	Description
SE	X'F0'	End of subnegotiation parameters.
NOP	X'F1'	No operation.
Data Mark	X'F2'	The data stream portion of a Synch. This should always be accompanied by a TCP Urgent notification.
Break	X'F3'	NVT character BRK.
Interrupt Process	X'F4'	The function IP.
Abort output	X'F5'	The function AO.
Are You There	X'F6'	The function AYT.
Erase character	X'F7'	The function EC.
Erase Line	X'F8'	The function EL.
Go ahead	X'F9'	The GA signal.
SB	X'FA'	Indicates that what follows is subnegotiation of the indicated option.
WILL (option code)	X'FB'	Indicates the desire to begin performing, or confirmation that you are now performing, the indicated option.
WON'T (option code)	X'FC'	Indicates the refusal to perform, or continue performing, the indicated option.
DO (option code)	X'FD'	Indicates the request that the other party perform, or confirmation that you are expecting the other party to perform, the indicated option.
DON'T (option code)	X'FE'	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.
IAC	X'FF'	Data byte 255.

Table 27 lists the options available for Telnet commands from RFC 1060. For more information about Telnet protocols, refer to RFC 1060 and RFC 1011.

Table 27. Telnet command options from RFC 1060

Option	Option (Hex)	Name
0	0	Binary Transmission
1	1	Echo
2	2	Reconnection
3	3	Suppress Go Ahead
4	4	Approx Message Size Negotiation
5	5	Status
6	6	Timing Mark
7	7	Remote Controlled Trans and Echo
8	8	Output Line Width
9	9	Output Page Size
10	A	Output Carriage-Return Disposition

Table 27. Telnet command options from RFC 1060 (continued)

Option	Option (Hex)	Name
11	B	Output Horizontal Tab Stops
12	C	Output Horizontal Tab Disposition
13	D	Output Formfeed Disposition
14	E	Output Vertical Tabstops
15	F	Output Vertical Tab Disposition
16	10	Output Linefeed Disposition
17	11	Extended ASCII
18	12	Logout
19	13	Byte Macro
20	14	Data Entry Terminal
21	15	SUPDUP
22	16	SUPDUP Output
23	17	Send Location
24	18	Terminal Type
25	19	End of Record
26	1A	TACACS User Identification
27	1B	Output Marking
28	1C	Terminal Location Number
29	1D	Telnet 3270 Regime
30	1E	X.3 PAD
31	1F	Negotiate About Window Size
32	20	Terminal Speed
33	21	Remote Flow Control
34	22	Linemode
35	23	X Display Location
255	FF	Extended-Options-List

Chapter 15. 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 the send mail to a local or remote host.

This chapter describes how to diagnose problems with SMTP and contains the following chapters:

- "Sender SMTP"
- "Receiver SMTP"
- "SMTP environment"
- "SMTP definitions" on page 452
- "Diagnosing SMTP problems" on page 452
- "ADDRBLOK data set" on page 457
- "SMTP RESOLVER trace" on page 459

For information about diagnosing problems with the other z/OS Communications Server mail application, z/OS UNIX sendmail, see Chapter 16, "Diagnosing z/OS UNIX sendmail and popper problems," on page 463.

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 60 on page 452 shows the SMTP environment.

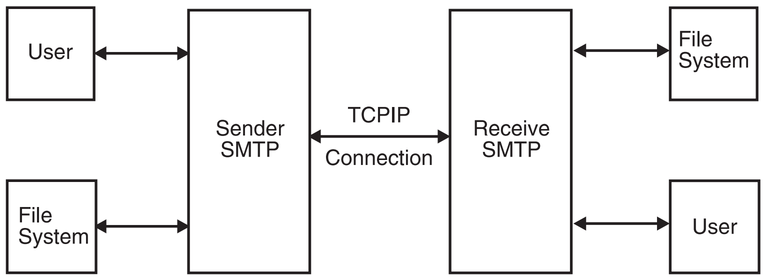


Figure 60. SMTP environment

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,MVPTSSI
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 MVS Diagnosis: Procedures* or see Chapter 3, "Diagnosing abends, loops, and hangs," on page 23, 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 457.

     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 459.

     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
- ADDRBLOK 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
  ```

See Chapter 3, “Diagnosing abends, loops, and hangs,” on page 23 for more diagnostic information about diagnosing loops.

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* 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 28 shows the format of Record 1 (the master control record) of an SMTP ADDRBLOK data set.

Table 28. Format of Record 1 of an SMTP ADDRBLOK data set

Characters	Description	Length (in characters)
1–7	Total number of recipients	7
8–14	Number of unresolved recipients	7
15–21	Number of recipients left to send this mail item to	7
22	Unused	1
23–30	File name of note file	8
31	Unused	1
32–39	Date	8
40	Unused	1
41–48	Time	8
49	Unused	1
50–53	Unused	4
54–55	Unused	2
56	Key **Value** **Meaning** **B** BSMTP RPLY file **S** Spool file **M** Spool file from Mailer **T** File from TCP **E** Error file	1
Note: Characters 57–80 are optional data used only when the key (Character 56) is "S" or "M."		

Table 28. Format of Record 1 of an SMTP ADDRBLOK data set (continued)

Characters	Description	Length (in characters)
57–64	Tag user ID	8
65–72	Tag node ID	8
73–80	Spool ID on the current system	8
77–80	Spool ID of the file source	4

Table 29 shows the format of Record 2 (for an unresolved From record) of an SMTP ADDRBLOK data set.

Table 29. Format of Record 2 (for an unresolved from record) of an SMTP ADDRBLOK data set

Characters	Description	Length (in characters)
1	Key **Value** **Meaning** **U** Unresolved	1
2	Sender path length (user host.domain)	1
3–4	Length of sender ID	2
5–(L1+4)	Sender ID (who sent the mail)	L1
(L1+5) –(L1+6)	Length of sender host.domain	2
(L1+7) –(L1+L2+6)	Sending host.domain	L2
(L1+L2+7)	Length of sender ID	1
(L1+L2+8) –(L1+L2+L3+7)	Sender ID (who sent the mail)	L3

Table 30 shows the format of Record 2 (for a resolved From record) of an SMTP ADDRBLOK data set.

Table 30. Format of Record 2 (for a resolved from record) of an SMTP ADDRBLOK data set

Characters	Description	Length (in characters)
1	Key **Value** **Meaning** **M** Resolved	1
2	Sender path length (user host.domain)	1
3–4	Length of sender ID	2
5–(L1+4)	Sender ID (who sent the mail)	L1
(L1+5) –(L1+6)	Length of sender host.domain	2
(L1+7) –(L1+L2+6)	Sending host.domain	(L1+L2+7)
(L1+L2+8)	Length of sender ID	1
(L1+L2+9) –(L1+L2+L3+8)	Sender ID (who sent the mail)	L3
(L1+L2+L3+9)	Length of encoded return path	1
(L1+L2+L3+10) –(L1+L2+L3+L4+9)	Encoded return path	L4

Table 31 shows the format of Records 3–*n* of an SMTP ADDRBLOK data set.

Table 31. Format of Record 3 (for an unresolved from record) of an SMTP ADDRBLOK data set

Characters	**Description**	**Length (in characters)**
1	Key **Value** **Meaning** **U** Unresolved **M** Resolved	1
2–5	Time-to-Live (TTL)	4
6	Length of return path	1
7–8	Length of recipient user ID	2
9–(L1+8)	Recipient user ID	L1
(L1+9) –(L1+11)	Length of recipient host.domain	2
(L1+12) –(L1+L2+11)	Recipient’s host.domain	L2
(L1+L2+12)	Length of recipient path	1
(L1+L2+13) –(L1+L2+L3+12)	Recipient path	L3
(L1+L2+L3+13)	Number of IP addresses	1
(L1+L2+L3+14) –(L1+L2+L3+17)	IP address 1	4
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 61 on page 460 shows an example of RESOLVER trace output. Short descriptions of the numbered items in the trace follow the figure.

```
  Userid of Caller:         ARMSTRNG
  TCP Host Name:            RALVMFE1
  Domain Origin:            RALEIGH.IBM.COM
  Jobname of TCP/IP:        TCPCS
  Communicate Via:          UDP
  OpenTimeOut:              30
  MaxRetrys:                1
  NSPort:                   53
  NameServer Userid:        NAMESRV
1 NSInternetAddress(.1.) := 9.67.1.5
  NSInternetAddress(.2.) := 9.67.5.44
  Data set prefix used:     TCPCS.BTA1

  Resolving Name:           RICKA
  Result from InitResolver: OK
  Building Name Server Query:
  * * * * * Beginning of Message * * * * *
2 Query Id:                 1
3 Flags:                    0000 0001 0000 0000
  Number of Question RRs:   1
4 Question   1: RICKA.RALEIGH.IBM.COM A IN
  Number of Answer RRs:     0
  Number of Authority RRs:  0
  Number of Additional RRs: 0
  * * * * * End of Message * * * * *
5 Sending Query to Name Server at 9.67.1.5 Result: OK
6 Notification Arrived: UDP data delivered RC = OK
7 UDP Data Length: 55
  Return from WaitForAnswer: OK
  * * * * * Beginning of Message * * * * *
  Query Id:                 1
  Flags:                    1000 0101 1000 0000
  Number of Question RRs:   1
  Question   1: RICKA.RALEIGH.IBM.COM A IN
  Number of Answer RRs:     1
8 Answer     1: RICKA.RALEIGH.IBM.COM 86400 A IN 9.67.97.3
  Number of Authority RRs:  0
  Number of Additional RRs: 0
  * * * * * End of Message * * * * *
  HostNumber (1) is: 9.67.97.3
```

Figure 61. Example of RESOLVER trace output

Following are short descriptions of 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.

Bit	Meaning
0	Operation: 0=query, 1=response
1–4	Query type: 0=standard, 1=inverse
5	Set if the answer is authoritative
6	Set if the message is truncated
7	Set if recursion is desired
8	Set if recursion is available
9–11	Reserved
12–15	Response type

Value Meaning

0	No error
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 16. Diagnosing z/OS UNIX sendmail and popper problems

This chapter 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 chapter:

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 chapter consistently refers to the *bat book* for further information.

You can also find more information about sendmail at the *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 15, "Diagnosing Simple Mail Transfer Protocol (SMTP) problems," on page 451.

Debugging switches

Table 32 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 32. Debugging switches by category

Category	Bat book reference	Useful for mail problems	Description
-d0.1	16.6.1	X	Print version, compliation, and interface information
-d0.4	16.6.2	X	Our name and aliases
-d0.10	16.6.3		Operating System defines
-d0.12	16.6.4	X	Print library (libsm) defines
-d0.13	16.6.5	X	FFR Defines: _FFR_MILTER_PERDAEMON

Table 32. Debugging switches by category (continued)

Category	Bat book reference	Useful for mail problems	Description
-d0.22	16.6.6		Dump delivery agents
-d0.40	16.6.7		Print network address of each interface
-d0.44	16.6.8		End with finis()
-d2.9	16.6.9		Show file descriptors with *dumpfd()*
-d1.1	16.6.10		Trace enoughspace()
-d1.5	16.6.11		Show failed mail
-d2.1	16.6.12		DNS name resolution
-d2.9	16.6.13		Call to getcanonname(3)
-d3.1	16.6.14		Trace dropped local hostnames
-d3.5	16.6.15		Hostname being tried in getcanonname(3)
-d3.15	16.6.16		Yes/no response to -d8.5
-d3.20	16.6.17		Resolver debugging
-d3.30	16.6.18		Trace delivery
-d11.2	16.6.19	X	Show the user-id running as during delivery
-d12.1	16.6.20		Show mapping of relative host
-d13.1	16.6.21		Show delivery
-d20.1	16.6.22		Show resolving delivery agent:parseaddr()
-d21.1	16.6.23	X	Trace rewriting rules
-d21.2	16.6.24		Trace $&macros
-d22.1	16.6.25		Trace tokenizing an address : prescan()
-d22.11	16.6.26		Show address before prescan
-d22.12	16.6.27		Show address after prescan
-d25.1	16.6.28		Trace ″sendlist″
-d26.1	16.6.29		Trace recipient queueing
-d27.1	16.6.30	X	Trace aliasing
-d27.2	16.6.31	X	Include file, self-reference, error on home
-d27.3	16.6.32	X	Forwarding path and alias wait
-d27.4	16.6.33	X	Print not safe
-d27.5	16.6.34	X	Trace aliasing with printaddr[]
-d27.8	16.6.35	X	Show setting up an alias map
-d27.9	16.6.36		Show user-id/group-id changes with:Include:reads
-d28.1	16.6.37		Trace user database transactions
-d29.1	16.6.38		Special rewrite of local recipient
-d29.4	16.6.39		Trace fuzzy matching
-d31.2	16.6.40		Trace processing of headers
-d34.1	16.6.41		Watch header assembly for output
-d34.11	16.6.42		Trace header generation and skipping
-d35.9	16.6.43		Macro values defined
-d37.1	16.6.44	X	Trace settings of options

Table 32. Debugging switches by category (continued)

Category	Bat book reference	Useful for mail problems	Description
-d37.8	16.6.45	X	Trace adding of words to a class
-d38.2	16.6.46		Show database map opens and failures
-d38.3	16.6.47	X	Show passes
-d38.4	16.6.48	X	Show result of database map open
-d38.9	16.6.49		Trace database map closing and appends
-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
  ```

  For descriptions of sendmail messages, refer to *z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)*.
- 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*.
- 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*.
- 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 33.

Table 33. qf File code letters

Code	Reference	Meaning	How Many
A	Bat book 11.11.1	AUTH=parameter	At most, one
B	Bat book 11.11.2	Message body type	At most, one
C	Bat book 11.11.3	Set controlling user	At most, one per R line
d	Bat book 11.11.4	Data file directory	Exactly one
D	Bat book 11.11.5	Data file name	Exactly one
E	Bat book 11.11.6	Send errors to	Many
F	Bat book 11.11.7	Save flagged bits	Exactly one
H	Bat book 11.11.8	Header line	Many
I	Bat book 11.11.9	Mode and device information for the df file	Exactly one
K	Bat book 11.1.10	Time last processed	Exactly one
M	Bat book 11.11.11	Message (why Manyqueued)	At most one
N	Bat book 11.11.12	Number times tried	At most, one
P	Bat book 11.11.13	Priority (current)	At most, one
Q	Bat book 11.11.14	The DSN ORCPT address	At most, one per R ine
r	Bat book 11.11.15	Final recipient	At most, one
R	Bat book 11.11.16	Recipient address	Many
S	Bat book 11.11.17	Sender address	Exactly one
T	Bat book 11.11.18	Time created	Exactly one
V	Bat book 11.11.19	Version	Exactly one
Z	Bat book 11.11.20	DSN envelope ID	At most, one

Table 33. qf File code letters (continued)

Code	Reference	Meaning	How Many
!	Bat book 11.11.21	Delivery by specification	At most, one
$	Bat book 11.11.22	Restore macro value	At most, one
.	Bat book 11.11.23	End of qf file	Exactly one

Bat book refers to *sendmail* by O'Reilly & Associates, Inc. (1-56592-839-3). Op refers to *Sendmail Installation and Operation Guide* that is shipped in /usr/lpp/tcpip/samples/sendmail/sendmail.ps.

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 170 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 Service System Secure Sockets Layer 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 stringI: 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.

EZZ9961 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 sendmails'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 (SYSTCPDA) for TCP/IP stacks" on page 86 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 SMMSPGRP 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 SMMSP DFLTGRP(SMMSPGRP) 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 BPX.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 (smmsp 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:

```
> 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: 9BPXROOT@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 17. 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 chapter describes how to diagnose problems with the SNALINK LU0 function and contains the following sections:

- "Definitions"
- "Problem diagnosis"
- "Traces" on page 477

SNALINK LU6.2 diagnosis is discussed in Chapter 18, "Diagnosing SNALINK LU6.2 problems," on page 481.

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, abend X'0C2' or X'0F8' 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 23.

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

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*.

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 the *z/OS Communications Server: IP Configuration Reference* 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 477.

Session outages

A session outage is an unexpected abend or termination of the task. Session outages are usually seen only when an irrecoverable 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=080A0000`, 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 41 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 62 on page 478 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.

```
00012A40   00000000   00000000   E3D9C3E3   C2D34040   | ........TRCTBL   |
00012A50   00018640   00018300   00020000   A020006C   | ..f ..c........% |
00012A60   00000000   80C15008   94000001   00000000   | .....A&.m....... |
.
.
.
00018300   B56D355D   F3C92348   00000000   00000000   | ._.)3I.......... |
00018310   40000000   00000045   00000000   800091DA   |  .............j. |
00018320   B56D355D   F76940CA   00000000   00000000   | ._.)7. ......... |
00018330   10230000   00012B80   00000000   00000000   | ................ |
00018340   B56D355D   F769568A   00000000   00000000   | ._.)7........... |
00018350   40000000   00000006   00000002   8000E256   |  .............S. |
00018360   B56D355D   F77C748A   00000000   00000000   | ._.)7@.......... |
00018370   40000000   00000007   00000001   8000E2A2   |  .............Ss |
00018380   B56D355D   F789628A   00000000   00000000   | ._.)7i.......... |
00018390   40000000   0000005A   00000002   8000E2D4   |  ......!......SM |
000183A0   B56D356C   173D7DC8   E2D5C1D3   E4F0F2C1   | ._.%..'HSNALU02A |
000183B0   40000000   00000034   00000000   8000CEF8   |  ..............8 |
000183C0   B56D356C   1788D188   E2D5C1D3   E4F0F2C1   | ._.%.hJhSNALU02A |
000183D0   10170000   00024E30   00000000   80000000   | ......+......... |
000183E0   B56D356C   58C0DACB   00000000   00000000   | ._.%.{.......... |
000183F0   17101001   00024E30   00024FC0   080A0000   | ......+...|{.... |
00018400   B56D3573   3F9CF1CB   00000000   00000000   | ._....1......... |
00018410   31000800   091BF1F8   31010207   00000000   | ......18........ |
00018420   B56D3573   3F9E5C4B   E2D5C1D3   E4F0F2C1   | ._....*.SNALU02A |
00018430   40000000   00000019   00000000   80009E74   |  ............... |
00018440   B56D3573   3FDE2DCB   00000000   00000000   | ._.............. |
00018450   2A100000   00024DC0   00024F80   00000000   | ......({..|..... |
00018460   B56D3573   3FDEF7CB   E2D5C1D3   E4F0F2C1   | ._....7.SNALU02A |
00018470   102A0000   00024DC0   00024F80   80000000   | ......({..|..... |
00018480   B56D3573   3FDF0C8B   E2D5C1D3   E4F0F2C1   | ._......SNALU02A |
00018490   40000000   00000034   00000000   8000CEF8   |  ..............8 |
000184A0   B56D3573   401997CB   E2D5C1D3   E4F0F2C1   | ._.. .p.SNALU02A |
000184B0   10170000   00024E30   00024FC0   80000000   | ......+...|{.... |
000184C0   B56D3573   4019F1CB   E2D5C1D3   E4F0F2C1   | ._.. .1.SNALU02A |
000184D0   40000000   00000024   00000000   8000A59C   |  .............v. |
000184E0   B56D3573   8161A8CB   00000000   00000000   | ._..a/y......... |
000184F0   17100000   00024E30   00024FC0   00000000   | ......+...|{.... |
.
.
.
000185F0   23100000   00024F10   00037000   80000118   | ......|......... |
00018600   B56D3585   4906F5C0   E2D5C1D3   E4F0F2C1   | ._.e..5{SNALU02A |
00018610   02000000   03790011   00037000   84000C60   | .....`......d..- |
00018620   B56D3585   49081240   E2D5C1D3   E4F0F2C1   | ._.e... SNALU02A |
00018630   10230000   00024F10   00037000   80009000   | ......|......... |
00018640.:018FFF.--All bytes contain X'00'
```

Figure 62. Example of a SNALINK LU0 DEBUG trace

The layout of a SNALINK trace table entry is shown in Table 34.

Table 34. Format of a SNALINK trace table entry

Bytes	**Definition**
00–07	TOD time stamp
08–0F	LU name, if any

Table 34. Format of a SNALINK trace table entry (continued)

Bytes	Definition
10	Entry Type **Value** 01 DLC Accept 02 DLC Send 03 DLC Receive 04 DLC Sever 05 DLC Msg Pend Queue Request 06 DLC Msg Pend D-Queue Request 0E MVS DLC emulation 0F DLC Interrupt 10 VTAM Request 17 VTAM OPNDST Exit 1F VTAM CLSDST Exit 22 VTAM SEND Exit 23 VTAM Receive Exit 25 VTAM SESSIONC Exit 2A VTAM OPNSEC Exit 2C VTAM TERMSESS Exit 31 VTAM SCIP Exit 32 VTAM LOSTERM Exit 33 VTAM NSEXIT Exit 34 VTAM TPEND Exit 35 VTAM LOGON Exit 40 SNALINK Internal Message Routine Call
11	DLC Interrupt Code/VTAM RPL REQ Code/ VTAM Receive Exit Chain field
12	VTAM CMD: R15/VTAM Exit: RTNCD
13	VTAM CMD: R0 /VTAM Exit: FDB2/DLC IPRCODE
14-17	RPL Address/DLC MSG ID/TPEND reason code/Internal Message ID
18–1B	VTAM Send/Receive/DLC buffer address/Number of Arguments Passed to Internal Message routine
1C–1F	VTAM Send/Receive/DLC buffer length/Internal Message Routine caller's return address

Chapter 18. Diagnosing SNALINK LU6.2 problems

This chapter describes how to diagnose problems with the SNALINK LU6.2 function and contains the following sections:

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 63.

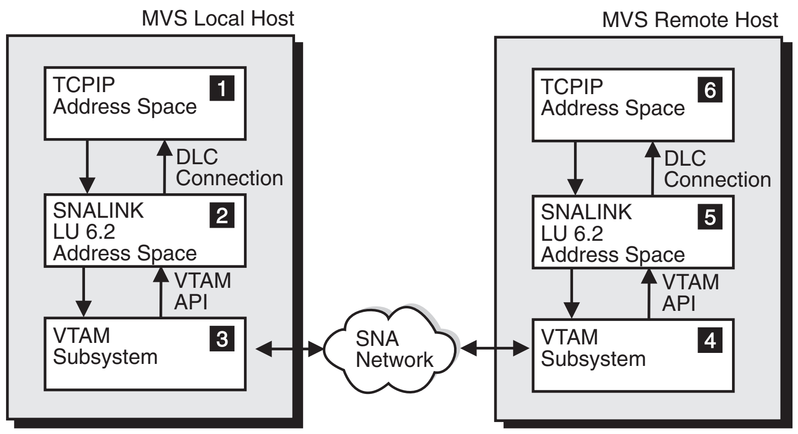


Figure 63. Components of a SNALINK LU6.2 connection on MVS

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.

4 VTAM routines on the destination system receive the data and pass it through to the SNALINK LU6.2 address space.

5 The SNALINK LU6.2 address space sends the data to the TCP/IP address space using a DLC connection.

6 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 63 on page 481.

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 495 for details about how to verify SNALINK LU6.2 connections.

The example in Figure 64 on page 483 shows the messages that are expected when the SNALINK LU6.2 address space is started and a network connection is established.

```
S SNAL621A
$HASP100 SNAL621A ON STCINRDR
$HASP373 SNAL621A STARTED
 1 IEF403I SNAL621A - STARTED - TIME=15.26.03
 2 EZA5927I LU62CFG : NO ERRORS DETECTED - INITIALIZATION WILL CONTINUE
 3 EZA5932I INITIALIZATION COMPLETE - APPLID: SNAL621A  TCP/IP: TCPCS
 4 EZA5935I SEND CONVERSATION ALLOCATED FOR 9.67.22.2
 5 EZA5933I LINK SNALU62L OPENED
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE SNALU621
 4 EZA5936I RECEIVE CONVERSATION ALLOCATED FOR 9.67.22.2
```

Figure 64. Sample MVS System Console Messages on SNALINK LU6.2 Address Space Startup

The following list explains the MVS system console messages on SNALINK LU6.2 address space startup as shown in Figure 64.

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.TCPIP 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 489 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 482 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 496 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 496 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 485 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 504 to 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 495 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 487 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 35 shows some of the common SNALINK LU6.2 address space startup problems.

Table 35. Common SNALINK LU6.2 address space startup problems

If this is displayed. . .	Then this might have occurred . . .	Resolution
The message `Errors Detected - Address Space will Terminate` has been displayed on the MVS system console with no other error messages.	This error message indicates that an error has occurred with the SNALINK LU6.2 configuration data set	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.
The message `Error in open of LU62CFG - no data will be read` has been displayed on the MVS system console.	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.	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.

Table 35. Common SNALINK LU6.2 address space startup problems (continued)

If this is displayed. . .	Then this might have occurred . . .	Resolution
The message `Address Space Already Active - this Address Space will Terminate` has been displayed on the MVS system console.	An address space with the same name as the SNALINK LU6.2 address space is already active.	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.
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 496 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 35. Common SNALINK LU6.2 address space startup problems (continued)

If this is displayed. . .	Then this might have occurred . . .	Resolution
The SNALINK LU6.2 address space abends with an abend code of S0F8 after the `Initialization Complete...` message.	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.	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.

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 504 and "Finding error message documentation" on page 504 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 495 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 36 on page 488 lists some of the common DLC connection problems between the SNALINK LU6.2 address space and the TCP/IP address space.

Table 36. Common DLC connection problems

If this is displayed. . .	Then this might have occurred . . .	Resolution
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`.	The TCP/IP address space is attempting to attach to the SNALINK LU6.2 address space, but the SNALINK LU6.2 address space is not responding.	Check whether the SNALINK LU6.2 address space is active and start it, if necessary.
The SNALINK LU6.2 address space has started, but the `Link open` message has not been displayed on the MVS system console, no other error messages have been displayed on the console, and the NETSTAT DEVLINKS output shows that the DLC connection status is either `Issued Connect` or `Will retry connect`.	This problem can be due to one of the following situations: 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. 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.	• 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 ddname 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* for an example of a SNALINK LU6.2 catalogued 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* 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* for details about the SNALINK LU6.2 LINK statement syntax and the TCPIP LINK statement syntax.

Table 36. Common DLC connection problems (continued)

If this is displayed. . .	Then this might have occurred . . .	Resolution
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`.	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.	Check the START statements in *hlq*.PROFILE.TCPIP. 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. 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 495 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 37 lists some of the common SNALINK LU6.2 address space network establishment problems.

Table 37. SNALINK LU6.2 address space network establishment problems

If this is displayed. . .	Then this might have occurred . . .	Resolution
The SNALINK LU6.2 address space issued error message: `Unable to allocate send conversation`.	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.	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 496 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.
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 495 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 504 for details on finding the documentation for these messages. Text in the message documentation specifies the action required to fix the problem.

Table 38 on page 492 lists an example of an outage problem.

Table 38. Outage problem

If this is displayed. . .	Then this might have occurred . . .	Resolution
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.	Large packet size sent in a PIU is rejected by the NCP with sense 800A0000 (PIU too large).	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 *z/OS Communications Server: SNA Network Implementation Guide* for more information on defining the MAXDATA, MAXBFRU, and UNITSZ operands.

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 495 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 502, 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 39 lists some of the common SNALINK LU6.2 data loss problems.

Table 39. Common SNALINK LU6.2 data loss problems

If this is displayed. . .	Then this might have occurred . . .	Resolution
Data packets are lost between the TCP/IP and the SNALINK LU6.2 address space (either end).	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.	1. See “DLC connection problems” on page 487 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 504 for details on finding the documentation for these messages. Text in the message documentation specifies the action required to fix the problem.
Data packets are actually not lost but the protocol (PING) times out.	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.	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.
Data packets are lost between the SNALINK LU6.2 devices.	The network is failing.	Check for VTAM error messages on the MVS system console. See “VTAM buffer traces” on page 504 for more details about using VTAM traces to diagnose the SNA network.

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 502.

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 65 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 65. 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 66 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     RECV:-  Status: Allocated      Packets In:  1
EZA5974I LIST COMPLETED
```

Figure 66. 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 504. 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” for an explanation of the active states for a minor node.

Refer to *z/OS Communications Server: SNA Operation* for a complete description of the VARY ACT subcommand.

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 496 for the major node activation subcommand.

Refer to *z/OS Communications Server: SNA Operation* for a complete description of the VARY INACT 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 67 on page 498 shows a sample of the output from an LU in connectable state.

```
D NET,ID=SNAL621A,E
IST097I DISPLAY ACCEPTED
IST075I NAME = NETA.SNAL621A, TYPE = APPL 573
IST486I STATUS= CONCT, DESIRED STATE= CONCT
IST1447I REGISTRATION TYPE = CDSERVR
IST977I MDLTAB=***NA*** ASLTAB=***NA***
IST861I MODETAB=***NA*** USSTAB=***NA*** LOGTAB=***NA***
IST934I DLOGMOD=LU62MODE USS LANGTAB=***NA***
IST1632I VPACING =  0
IST597I CAPABILITY-PLU INHIBITED,SLU INHIBITED,SESSION LIMIT NONE
IST231I APPL MAJOR NODE = SNALNK1A
IST654I I/O TRACE = OFF, BUFFER TRACE = OFF
IST1500I STATE TRACE = OFF
IST271I JOBNAME = ***NA***, STEPNAME = ***NA***, DSPNAME = ***NA***
IST228I ENCRYPTION = OPTIONAL , TYPE = DES
IST1563I CKEYNAME = SNAL621A CKEY = PRIMARY CERTIFY = NO
IST1552I MAC = NONE MACTYPE = NONE
IST1050I MAXIMUM COMPRESSION LEVEL - INPUT = 0, OUTPUT = 0
IST1633I ASRCVLM = 1000000
IST1634I DATA SPACE USAGE: CURRENT = ***NA*** MAXIMUM = ***NA***
IST171I ACTIVE SESSIONS = 0000000000, SESSION REQUESTS = 0000000000
IST172I NO SESSIONS EXIST
IST314I END
```

Figure 67. DISPLAY subcommand output example for connectable LU

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 68 shows a sample of the output from a DISPLAY command for an LU in active state.

```
D NET,ID=SNAL621A,E
IST097I DISPLAY ACCEPTED
IST075I NAME = NETA.SNAL621A, TYPE = APPL 545
IST486I STATUS= ACT/S, DESIRED STATE= ACTIV
IST1447I REGISTRATION TYPE = CDSERVR
IST977I MDLTAB=***NA*** ASLTAB=***NA***
IST861I MODETAB=***NA*** USSTAB=***NA*** LOGTAB=***NA***
IST934I DLOGMOD=LU62MODE USS LANGTAB=***NA***
IST1632I VPACING =  0
IST597I CAPABILITY-PLU ENABLED  ,SLU ENABLED  ,SESSION LIMIT NONE
IST231I APPL MAJOR NODE = SNALNK1A
IST654I I/O TRACE = OFF, BUFFER TRACE = OFF
IST1500I STATE TRACE = OFF
IST271I JOBNAME = SNAL621A, STEPNAME = SNAL621A, DSPNAME = IST84E76
IST228I ENCRYPTION = OPTIONAL , TYPE = DES
IST1563I CKEYNAME = SNAL621A CKEY = PRIMARY CERTIFY = NO
IST1552I MAC = NONE MACTYPE = NONE
IST1050I MAXIMUM COMPRESSION LEVEL - INPUT = 0, OUTPUT = 0
IST1633I ASRCVLM = 1000000
IST1634I DATA SPACE USAGE: CURRENT = 0 MAXIMUM = 136
IST171I ACTIVE SESSIONS = 0000000003, SESSION REQUESTS = 0000000000
IST206I SESSIONS:
IST634I NAME     STATUS         SID          SEND RECV VR TP NETID
IST635I SNAL622A ACTIV-S    EAABEEC3D9B90C37 0002 0000  0  0 NETA
IST635I SNAL622A ACTIV/SV-S EAABEEC3D9B90C35 0002 0002  0  0 NETA
IST635I SNAL622A ACTIV-P    F6ABEEC3DCB90B7D 0000 0002  0  0 NETA
IST314I END
```

Figure 68. DISPLAY subcommand output example for active LU

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
1 EZA5927I LU62CFG : NO ERRORS DETECTED - INITIALIZATION WILL CONTINUE
EZA5932I INITIALIZATION COMPLETE - APPLID: SNAL621A  TCP/IP: TCPCS
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
7 EZA5994I CNOS FOR INDEPENDENT PARTNER; SESSLIM=0002, WINNER=0001, LOSER=0001
EZA5984I   OLU= SNAL621A, DLU= SNAL622A, IP ADDRESS= 9.67.22.2
EZA6029E OPRCNOS  ERR. R15 00000000 R0 0000000B RTNCD 00000000 FDBK2 0000000B
EZA6030E OPRCNOS  ERR.  RCPRI= 0008,  RCSEC= 0001
EZA6031E OPRCNOS  SENSE CODE RECEIVED:  08570003
EZA6032E          SENSE CODE SPECIFIED: 00000000
EZA6023E UNABLE TO COMPLETE CNOS ON LU SNAL622A FOR 9.67.22.2
2 EZA6009W CONVERSATIONS FOR 9.67.22.2 TERMINATED
EZA6011E UNABLE TO ALLOCATE SEND CONVERSATION FOR 9.67.22.2
1 EZA5933I LINK SNALU62L OPENED
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
EZA5936I RECEIVE CONVERSATION ALLOCATED FOR 9.67.22.2
EZA5986I VTAM CONVERSATION ALLOCATED; CONVID= 01000011, SID= F6ABEEC3DCB90B89
EZA5984I   OLU= SNAL622A, DLU= SNAL621A, IP ADDRESS= 9.67.22.2
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
2 EZA5935I SEND CONVERSATION ALLOCATED FOR 9.67.22.2
EZA5986I VTAM CONVERSATION ALLOCATED; CONVID= 01000012, SID= EAABEEC3D9B90C3F
EZA5984I   OLU= SNAL621A, DLU= SNAL622A, IP ADDRESS= 9.67.22.2
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
4 EZA5992I IP DATAGRAM ADDED TO THE VTAM SEND QUEUE, LENGTH= 276,
QUEUE COUNT = 1
EZA5985I   LU= SNAL622A, LINKNAME= SNALU62L, IP ADDRESS = 9.67.22.2
3 EZA5988I VTAM SENT LOGICAL RECORD; CONVID= 01000012,
SID= EAABEEC3D9B90C3F, LENGTH= 280
EZA5984I   OLU= SNAL621A, DLU= SNAL622A, IP ADDRESS= 9.67.22.2
EZA5995I NUMBER OF IP PACKETS SENT ON 9.67.22.2 = 1
6 EZA5999I   450001140315000040014D4C0943160109430102080 0D58BEFBF74
0AB56D2E96EF8F85CA03EFDD80
EZA5999I   78B6FE8035FE858058EE968017DAAD8015B4AE80092C55803DA14680271B7D807DCC5
E8074502580
EZA5999I   39CDF68000F24D8023BE0E8012A9F5805434A6804C9F1D806249BE805779C5807B955
6800961ED80
EZA5999I   7C2F6E800DFF958006B006800E7ABD801C2F1E80597B65803444B6800B298D805508
CE80352D3580
EZA5999I   2B13668006AE5D80217C7E807455058079DC168060492D80644A2E804232D580175E
C6804F39FD80
EZA5999I   6831DE802206A580625B768062C0CD805FF38E806F1075802192268002 1D9D80664F3
E805C904580
EZA5999I   03C2D6806C906D807E04EE8075C615801FAD868039593D8011D49E801DF1E5807412
368000000000
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
EZA5997I CONNECTION 9.67.22.2 WILL TIMEOUT IN 600 SECONDS
3 EZA5989I VTAM RECEIVED LOGICAL RECORD; CONVID= 01000011,
SID= F6ABEEC3DCB90B89, LENGTH= 280
EZA5984I   OLU= SNAL622A, DLU= SNAL621A, IP ADDRESS= 9.67.22.2
EZA5996I NUMBER OF IP PACKETS RECEIVED ON 9.67.22.2 = 1
6 EZA5999I   4500011402E8000040014D7909430102094316010000DD8BEFBF74
0AB56D2E96EF8F85CA03EFDD80
EZA5999I   78B6FE8035FE858058EE968017DAAD8015B4AE80092C55803DA14680271B7D807DCC5
E8074502580
EZA5999I   39CDF68000F24D8023BE0E8012A9F5805434A6804C9F1D806249BE805779C5807B955
6800961ED80
EZA5999I   7C2F6E800DFF958006B006800E7ABD801C2F1E80597B65803444B6800B298D805508C
E80352D3580
EZA5999I   2B13668006AE5D80217C7E807455058079DC168060492D80644A2E804232D580175E
C6804F39FD80
EZA5999I   6831DE802206A580625B768062C0CD805FF38E806F10758021922680021D9D80664F3
E805C904580
```

Figure 69. SNALINK LU6.2 internal trace output (Part 1 of 2)

```
EZA5999I   03C2D6806C906D807E04EE8075C615801FAD868039593D8011D49E801DF1E5807412
368000000000
5 EZA5990I IP DATAGRAM PACKED INTO MESSAGE, LENGTH= 276
EZA5985I   LU= SNAL622A, LINKNAME= SNALU62L, IP ADDRESS = 9.67.22.2
EZA5938I RECEIVED OPERATOR SHUTDOWN REQUEST
2 EZA5937I LINK SNALU62L CLOSED
3 EZA5989I VTAM RECEIVED LOGICAL RECORD; CONVID= 01000011,
SID= F6ABEEC3DCB90B89, LENGTH= 280
EZA5984I   OLU= SNAL622A, DLU= SNAL621A, IP ADDRESS= 9.67.22.2
EZA5996I NUMBER OF IP PACKETS RECEIVED ON 9.67.22.2 = 1
6 EZA5999I   4500011402E8000040014D7909430102094316010000DD8BEFBF74
0AB56D2E96EF8F85CA03EFDD80
EZA5999I   78B6FE8035FE858058EE968017DAAD8015B4AE80092C55803DA14680271B7D807DCC5
E8074502580
EZA5999I   39CDF68000F24D8023BE0E8012A9F5805434A6804C9F1D806249BE805779C5807B955
6800961ED80
EZA5999I   7C2F6E800DFF958006B006800E7ABD801C2F1E80597B65803444B6800B298D805508C
E80352D3580
EZA5999I   2B13668006AE5D80217C7E807455058079DC168060492D80644A2E804232D580175E
C6804F39FD80
EZA5999I   6831DE802206A580625B768062C0CD805FF38E806F1075802192268002 1D9D80664F3
E805C904580
EZA5999I   03C2D6806C906D807E04EE8075C615801FAD868039593D8011D49E801DF1E5807412
368000000000
5 EZA5990I IP DATAGRAM PACKED INTO MESSAGE, LENGTH= 276
EZA5985I   LU= SNAL622A, LINKNAME= SNALU62L, IP ADDRESS = 9.67.22.2
EZA5938I RECEIVED OPERATOR SHUTDOWN REQUEST
2 EZA5937I LINK SNALU62L CLOSED
```

Figure 69. SNALINK LU6.2 internal trace output (Part 2 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.

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 41 for details about how to use the IP packet trace facility.

Figure 70 on page 503 shows an example of a CTRACE formatted packet trace record.

```
  19 VIC127   PACKET   00000001 22:52:18.648744 Packet Trace
 To Link          : SNALU62L        Device: SNA_LU6.2       Full=276
  Tod Clock       : 2001/02/20 22:52:18.648743
  Lost Records    : 0               Flags: Pkt Ver2 Out
  Source Port     : 0               Dest Port: 0     Asid: 01F6 TCB: 007AEE88
 IpHeader: Version : 4              Header Length: 20
  Tos             : 00 QOS: Routine  Normal Service
  Packet Length   : 276             ID Number: 1543
  Fragment        :                 Offset: 0
  TTL             : 64              Protocol: ICMP          CheckSum: 4A5A FFFF
  Source          : 9.67.22.1
  Destination     : 9.67.1.2

 ICMP
  Type/Code       : ECHO            CheckSum: 5B3F FFFF
  Id              : 4923            Seq: 11849
  Echo Data       : 248
000000 B56D52DB 47A07ACA 2D523F5F 72BE75F9 |.mR.G.z.-R?_r.u.
000010 36332E6F 5A2D7969 5F7DDC7F 401715D9 |63.oZ-yi_}..@...
000020 2B9B598F 6414BB49 0D13B59F 08F8D9B9 |+.Y.d..I........
000030 099E00AF 643EE129 5C304ABF 667B4199 |....d>.)\0J.f{A.
000040 260FA3CF 4CCB6B09 3EE01BDF 6B45CD79 |&...L.k.>...kE.y
000050 33B4C2EF 2069D8E9 051FA8FF 618FFD59 |3... i......a..Y
000060 3441DE0F 5059AAC9 2EDB721F 49215139 |4A..PY....r.I!Q9
000070 2A5B752F 5A6A60A9 7DEFF73F 15514919 |*.u/Zj`.}..?.QI.
000080 0B96084F 26FB7A89 4829B85F 2B0764F9 |...O&.z.H)._+.d.
000090 7276176F 26FC7869 0945357F 1EBB24D9 |rv.o&.xi.E5...$.
0000A0 1070228F 31ECDA49 34EEEE9F 327408B9 |.p".1..I4...2t..
0000B0 5FE8A9AF 23DC2029 48C363BF 13C99099 |_...#. )H.c.....
0000C0 16342CCF 3B69CA09 1E4F14DF 59E33C79 |.4,.;i...O..Y.<y
0000D0 55972BEF 37C557E9 7D0E81FF 43788C59 |U.+.7.W.}...Cx.Y
0000E0 1F46270F 36AE49C9 6C6E2B1F 34D10039 |.F'.6.I.ln+.4..9
0000F0 05659E2F 52741FA9                   |.e./Rt..

IP Header          : 20     IP: 9.67.22.1, 9.67.1.2
000000 45000114 06070000 40014A5A 09431601  09430102

Data               : 256    Data Length: 256
000000 08005B3F 49232E49 B56D52DB 47A07ACA |..$......_....:. ...?I#.I.mR.G.z.|
000010 2D523F5F 72BE75F9 36332E6F 5A2D7969 |...¼...9...?!... -R?_r.u.63.oZ-yi|
000020 5F7DDC7F 401715D9 2B9B598F 6414BB49 |¼'." ..R........ _}..@...+.Y.d..I|
000030 0D13B59F 08F8D9B9 099E00AF 643EE129 |.....8R......... ............d>.)|
000040 5C304ABF 667B4199 260FA3CF 4CCB6B09 |*....#.r..t.<.,. \0J.f{A.&...L.k.|
000050 3EE01BDF 6B45CD79 33B4C2EF 2069D8E9 |....,.....B...QZ >...kE.y3... i..|
000060 051FA8FF 618FFD59 3441DE0F 5059AAC9 |..y./..........I ....a..Y4A..PY..|
000070 2EDB721F 49215139 2A5B752F 5A6A60A9 |.........$..!.-z ..r.I!Q9*.u/Zj`.|
000080 7DEFF73F 15514919 0B96084F 26FB7A89 |'.7......o.|..:i }..?.QI....O&.z.|
000090 4829B85F 2B0764F9 7276176F 26FC7869 |...¼...9...?.... H)._+.d.rv.o&.xi|
0000A0 0945357F 1EBB24D9 1070228F 31ECDA49 |..."...R........ .E5...$..p".1..I|
0000B0 34EEEE9F 327408B9 5FE8A9AF 23DC2029 |........¼Yz..... 4...2t.._...#. )|
0000C0 48C363BF 13C99099 16342CCF 3B69CA09 |.C...I.r........ H.c......4,.;i..|
0000D0 1E4F14DF 59E33C79 55972BEF 37C557E9 |.|...T...p...E.Z .O..Y.<yU.+.7.W.|
0000E0 7D0E81FF 43788C59 1F46270F 36AE49C9 |'.a............I }...Cx.Y.F'.6.I.|
0000F0 6C6E2B1F 34D10039 05659E2F 52741FA9 |%>...J.........z ln+.4..9.e./Rt..|
```

Figure 70. 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 19. Diagnosing name server and dynamic domain name server (DDNS) problems

This chapter describes how to diagnose problems involving the BIND-based dynamic domain name server (DNS) and contains the following sections:

- "Diagnosing name server problems"
- "Diagnosing problems with connection optimization" on page 517

Problem diagnosis involving connection optimization is also described.

For additional information on diagnosing problems with a BIND-based name server, refer to *DNS and BIND, 4th Edition, Paul Albitz and Cricket Liu, 4th Edition Apr 2001 ISBN 0-596-00158-4*, available from the O'Reilly Online Catalog.

If, after reading this chapter and *DNS and BIND*, you are unable to solve a DNS-related problem and you require the services of the IBM Software Support Center, have the following available:

- For BIND 4.9.3 name server, the output from syslogd and documentation from debug level 11.
- For the BIND 9 name server, 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 41 on page 510. 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.

Diagnosing name server problems

The following methods are available for identifying name server problems:

- "Determining the name server version" on page 506
- "Checking messages sent to the operator's console" on page 507
- "Checking the log messages" on page 507
- "Tools for querying the name server" on page 509
- "Using the debug option with the name server" on page 509
- "Debugging with a resolver directive" on page 511
- "Using the remote name daemon control (rndc) program, BIND 9 name server only" on page 511
- "Using name server signals" on page 512
- "Interpreting BIND 4.9.3 name server statistics" on page 514
- "Statistics file for the Bind 9 name server" on page 515
- "Using the nsupdate command" on page 515
- "Using component trace" on page 516

These methods are discussed in the following sections.

Determining the name server version

BIND (Berkeley Internet Name Domain) is an implementation of the Domain Name System (DNS) protocols and provides an openly redistributable reference implementation of the major components of the Domain Name System, including:

- A Domain Name System server (NameD)
- A Domain Name System resolver library
- Tools for verifying the proper operation of the DNS server

The z/OS Communications Server supports the following versions of BIND:

- BIND 4.9.3
- BIND 9

The name server and several name server utilities operate with either BIND 4.9.3 (v4) or BIND 9 (v9). Generally, the v4 utilities should be used with the v4 name server, and the v9 utilities should be used with the v9 name server.

You might need to determine which name server version you are communicating with (on a z/OS platform) in order to solve some problems.

Queries can be sent using `dig` or `nslookup` (version 4 or version 9) to tell if you are communicating with a BIND 4.9.3 name server or a BIND 9 name server. The queries for the name *version.bind* must be of class CHAOS and the query type must be TXT or ANY.

- If the query response is `Server failed` or another type of negative response, the name server that sent the answer is a BIND 4.9.3 name server.
- If the query response does not indicate failure, the name server that responded is a BIND 9 name server.

  Tip: The answer might contain the actual version of BIND, or it might contain other text, depending on whether the name server administrator has overridden the default text.

The following examples show typical responses that might be received when determining name server version.

- Responding name server is a BIND 9 name server:

```
nslookup - 127.0.0.1
Defaulting to nslookup version 4
Starting nslookup version 4
Default Server:  localhost
Address:  127.0.0.1

> set class=chaos
> set type=any
> version.bind
Server:  localhost
Address:  127.0.0.1

version.bind    text = "9.2.0"
>
```

- Responding name server is a BIND 4.9.3 name server:

```
nslookup - 127.0.0.1
Defaulting to nslookup version 4
Starting nslookup version 4
Default Server:  localhost
Address:  127.0.0.1

> set class=chaos
> set type=any
> version.bind
Server:  localhost
Address:  127.0.0.1

localhost can't find version.bind: Server failed
>
```

Refer to *z/OS Communications Server: IP System Administrator's Commands* for additional information.

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 (EZZ, 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. An MVS console message is issued indicating that the name server ended with a fatal 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 (EZY)*.

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:

```
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 40:

Table 40. Logging categories

Category	Description
Default	Defines the logging options for those categories where no specific configuration has been defined.
general	Any items not otherwise categorized.
queries	Queries the server is receiving (not logged through default category).
database	Messages relating to the databases used internally by the name server to store zone and cache data.
security	Configuration file parsing and processing.
config	Configuration file parsing and processing.
resolver	DNS resolution, such as the recursive lookups performed on behalf of clients by a caching name server.
unmatched	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.
xfer-in	Zone transfers the server is receiving.
xfer-out	Zone transfers the server is sending.
notify	The NOTIFY protocol.
client	Processing of client requests.
network	Network operations.
update	Dynamic updates.

Table 40. Logging categories (continued)

Category	Description
dispatch	Dispatching of incoming packets to the server modules where they are to be processed.
dnssec	DNSSEC and TSIG protocol processing.
lame-servers	Lame servers. These are misconfigurations in remote servers, discovered by BIND 9 when trying to query those servers during resolution.

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 resolver tracing from nslookup v4, enter the following commands from the z/OS UNIX shell:

```
onslookup
set debug
```

To turn the resolver tracing off, enter the **set nodebug** command.

You can also turn on resolver tracing for v4 nslookup by entering the following command:

```
onslookup
set d2
```

To turn d2 off, enter `set nod2`. For v4 nslookup, turning off d2 does not turn off the resolver tracing if debug is also on. To turn off both d2 and debug, enter `set nodebug`. With BIND 9 nslookup, debug and d2 can be turned on and off independently. Using debug does not turn on resolver tracing as it does for v4 nslookup, 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 4.9.3 name server, the USR1 kill signal can be used to increment debugging to the desired level (see "Using name server signals" on page 512 for more information). For the BIND 9 name server, the **rndc** command can be used to dynamically turn on tracing. Valid levels for BIND 4.9.3 for the -d option are in the range of 1–11, where 11 supplies the most information. Valid levels for BIND 9 are in the range of 1-99. Debugging information is sent to the file /tmp/named.run, for BIND 4.9.3. 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, BIND 9 name server only" on page 511 for more information.

For BIND 9, Table 41 lists the types of debug information that are captured at the following debug levels.

Table 41. BIND 9 debug information

BIND 9 debug level	Debugging information
1	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.
2	Multicast requests.
3	Journal activity when dynamic update is enabled, DNSSEC and TSIG validation (if configured), and lower level task creation operations.
4	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.
5	Captures the view being used in order to answer a request.
6	Some outgoing zone transfer requests including the query that initiates the transfer.

Table 41. BIND 9 debug information (continued)

7	The additions and deletions to journal files. and the number of bytes returned on zone transfers.
8	Most dynamic update activity and more detailed information on zone transfers.
10	Timer activity for zones.
20	Zone refresh timer updates.
90	Detailed information about task dispatching and operations.

For BIND 4.9.3, if named is started from the z/OS UNIX shell with the -d option, use the ampersand (&) character as a shell operator at the end of the command line to run named in the background. If you do not use the ampersand, the named tracing process occupies the z/OS UNIX shell.

With BIND 4.9.3, debug information generated during zone transfers is written to /tmp/xfer.ddt.*xxxxxx*, where *xxxxxx* is a unique identifier. One of these files is generated for each zone for which the named daemon is a secondary server. 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 BIND 4.9.3, if the debug level is 6 or greater, the debug information exchanged during the last initiated zone transfer is written to /tmp/xfer.trace.

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 37, "Diagnosing resolver problems," on page 761.

Using the remote name daemon control (rndc) program, BIND 9 name server only

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 *z/OS Communications Server: IP System Administrator's Commands* and the *z/OS Communications Server: IP Configuration Reference* for additional information.

Table 42The following **rndc** commands can be used to provide diagnostic data for the BIND 9 name server.

Table 42. **rndc** *commands*

Command	Description
dumpdb	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).
trace	Increment the server's debugging level by one.
trace *level*	Increment the servers debugging level to an explicit value.
notrace	Sets the servers debugging level to 0.
flush	Flushes the servers cache.
status	Displays the status of the server.
stats	Writer server statistics to the statistics file.
querylog	Toggle query log

Refer to *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.

The name server signals that are used to collect diagnostic information for the BIND 4.9.3 name server do not perform the same function as the signals that can be sent to the BIND 9 name server.

Restriction: The sigINT signal terminates the BIND 9 name server. If you send the signal to a BIND 4.9.3 name server, it dumps the name server database.

Table 43 lists the signals for the BIND.4.9.3 name server:

Table 43. BIND.4.9.3 name server signals

Signal	Description
HUP	Reloads the boot file, named.boot, from the disk.
INT	Dumps the contents of the name server database and hints (root server) file into the /tmp/named_dump.db file.
ABRT	Dumps the current statistics of the name server in the /tmp/named.stats file.
USR1	Starts debug tracing for the name server and causes the named daemon to write debugging information to the file /tmp/named.run. USR1 can also be used to increase the debug level. Every time the USR1 signal is received, the debug level is increased until it reaches 11.
USR2	Stops debug tracing for the named daemon.

Table 43. BIND.4.9.3 name server signals (continued)

Signal	Description
SIGWINCH	Toggles query logging on and off. Use query logging to identify resolver configuration errors. When query logging is turned on, a running name server logs every query with the syslog daemon. The syslog messages that are displayed include the IP address of the host that made the query and the query itself.

Restriction: Signals do not affect zone transfers in progress. If debug is on, debugging for zone transfers occurs when the command **named_xfer** is invoked.

SIGHUP
Causes the server to read named.conf and reload the database.

SIGTERM
Causes the server to clean up and exit.

SIGINT
Causes the server to clean up and exit.

There are three signals that currently can be used with the BIND 9 name server. Some of the signals might have different consequences if sent to the BIND.4.9.3 name server. Table 44 lists the BIND 9 name server signals:

Table 44. BIND 9 name server signals

Signal	Description
SIGHUP	Causes the server to read named.conf and reload the database.
SIGTERM	Causes the server to clean up and exit.
SIGINT	Causes the server to clean up and exit.

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:

s nssig,sig=hup

Values for the sig parameter are the same as those for the -s parameter of the OMVS **kill** command. The following examples show how BIND 4.9.3 uses name server signals with the **kill** command. The process ID of the named daemon is stored in the /etc/named.pid file on startup.

- To dump the contents of the name server database, enter the **kill -INT $(cat /etc/named.pid)** command from the z/OS UNIX shell, and then check the file /tmp/named_dump.db.
- To get short status from the named daemon, enter the **kill -ABRT $(cat /etc/named.pid)** command from the z/OS UNIX shell, and then check the file /tmp/named.stats.
- To enable debug message logging for the named daemon, enter the **kill -USR1 $(cat /etc/named.pid)** command from the z/OS UNIX shell, and then check the file /tmp.named.run.

- To disable debugging, enter the **kill -USR2 $(cat /etc/named.pid)** command from the z/OS UNIX shell.
- To turn query logging on, enter the **kill -WINCH $(cat /etc/named.pid)** command from the z/OS UNIX shell. Before logging queries, make sure that the syslog daemon is logging LOG_INFO messages. To turn off query logging, send another **kill -WINCH $(cat /etc/named.pid)** signal to the name server.

  Note: You can also turn query logging on by inserting the directive `options query-log` in the name server boot file or by starting the name server with -q on the command line.

Interpreting BIND 4.9.3 name server statistics

Message EZZ6469I can display name server statistics. You can use this information to get an impression of your name server's health by comparing sets of statistics over time. For example, if your name server is sending SERVFAIL responses, you might have a name server configuration error. Another use of statistics is to find out what a normal query load is on your name server. For example subtract the "time now" field from the "time since last boot" field to find the total number of seconds the name server has been up. Divide the total number of queries by this number to receive your queries-per-second answer.

The following shows an example of how to interpret NSTATS in message EZZ6469I:

```
EZZ6469I NSTATS 992442646 992439045 A=1 SOA=26 X25=4 AXFR=23
                   ^         ^       ^     ^
                   |         |       |     |
                   |         |       |     Number of 'SOA' type queries received
                   |         |       Number of 'A' type queries received
                   |         Time since last boot
                   Time now
```

Message EZZ6469I also displays XSTATS information following the NSTATS information. The XSTATS information breaks down the number of queries and responses sent and received into more detail. The following describes that breakdown:

```
RR:          Received an answer
RNXD:        Received a negative answer ("don't know")
RFwdR:       Received an answer we had to forward
RDupR:       Received an extra answer
RFail:       Received a SERVFAIL
RFErr:       Received a FORMERR
RErr:        Received some other error
RAXFR:       Zone transfers initiated
RLame:       Received a lame delegation
ROpts:       Packets received with IP options set
SSysQ:       Sent a sysquery
SAns:        Sent an answer
SFwdQ:       Had to forward a query
SDupQ:       Sent a retry question (?)
SErr:        Sent failed (in sendto)
RQ:          Received a query
RIQ:         Received an inverse query
RFwdQ:       Received an query we had to forward
RDupQ:       Received a retry question
RTCP:        Received a query using TCP. EG. large query
SFwdr:       Forwarded  a response
SFail:       Sent a SERVFAIL
SFErr:       Sent a FORMERR
SNaAns:      Sent non-authoritative answer (from cache)
SNXD:        Sent a negative response ("I don't know")
```

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.

nxrrset
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. The -v option is used for nsupdate Bind 4 version debugging. It turns on verbose mode and displays all requests to and responses from the name server. To turn on debugging for Bind 4 version in interactive mode, enter the following commands from z/OS UNIX:

nsupdate

set v

For details on the **nsupdate** command, refer to the *z/OS Communications Server: IP System Administrator's Commands*.

Return codes

Following are the return codes, origination of the return codes, and explanations for the most common problems you might encounter with v4 nsupdate:

Return Code	Origin	Explanation
0	N/A	Successful.

Return Code	Origin	Explanation
-2	Local error	Input error.
-10	Local error	No key found in ETC\DDNS.DAT. A key is needed because either -f was specified or there is a KEY RR already in the name server data.
-11	Local error	Key in ETC\DDNS.DAT not valid. Does not authenticate the user.
-12	Local error	No response received from the name server.
-1	Local error	Represents any other (local) error not specified above.
1	Server error	Format error. The name server was unable to interpret the request.
2	Server error	Server failure. The name server was unable to process this request because of a problem with the name server.
3	Server error	Name error. The domain name specified does not exist.
4	Server error	Not implemented. The name server does not support the specified Operation code.
5	Server error	Refused. The name server refuses to perform the specified operation for security or policy reasons.
6	Server error	Alias error. A domain name specified in an update is an alias.
7	Server error	Name Exists error. A name already exists. This return code is only meaningful from a server in response to an ADDNAMENEW operation.
8	Server error	Record error. Indicates that a resource record (RR) does not exist. This return code is only meaningful from a server in response to a DELETE operation.
9	Server error	Zone error. Indicates that the update is to be performed on a zone for which the server is not authoritative, or that the records to be updated exist in more than one zone.
10	Server error	Ordering error. If an ordering mechanism is used (for example, a SIG RR or a SOA RR), this code indicates an ordering error. Time-signed problems are also indicated by this return code.

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 41.

For more information about Resolver CTRACE, see Chapter 37, “Diagnosing resolver problems,” on page 761.

Diagnosing problems with connection optimization

Connection optimization is a technique that uses the BIND.4.9.3 DNS for balancing IP connections and workload in a sysplex domain. You might encounter two types of problems involving connection optimization:

- Addresses not being returned
- Connection problems

Addresses not being returned

If the interface IP addresses defined for TCP/IP in the *hlq*.PROFILE.TCPIP data set and in your forward domain data file are not returned to your clients, one or more of the following situations is possible

- The adapters associated with those addresses have not been started. If this is the problem, start the adapters.
- The adapters are started, but the stack is not registered with Workload Manager (WLM). This situation affects clients using the sysplex domain name (for example, mvsplex.mycorp.com, where mvsplex is the name of the sysplex and mycorp.com is the domain name). For information on how to register stacks, refer to the *z/OS Communications Server: IP Configuration Reference*.
- WLM has not refreshed the name server since the adapters associated with those addresses started. By default, WLM updates the name server every minute. If the name server has not received the most recent information from WLM, waiting at least two minutes should remedy the situation. To set the refresh, use the -t option on the **named** command.
- The name server did not start and did not return any addresses. If this is the problem, start the name server. For directions on starting the name server, refer to the *z/OS Communications Server: IP Configuration Reference*.
- The CLUSTER keyword was not coded in the primary or secondary directive in the boot file. This causes the name server to only use the statically defined names in the forward domain data file; the name server does not add dynamically generated names and optimization does not occur. If this is the problem, code the CLUSTER keyword to identify the sysplex domain.
- The host that owns the addresses defined for TCP/IP in the *hlq*.PROFILE.TCPIP data set and in the forward domain data file is short on capacity. If a host system has little or no capacity for new connections, the name server receives weights from WLM that favor other hosts. Consequently, the overloaded host system might not receive any new connections.
- No server applications are registered with WLM or they are not currently available. This affects clients that attempt to use the server application group (for example, myserver.mvsplex.mycorp.com, where myserver is the name of the server group). For information on registering servers, refer to the *z/OS Communications Server: IP Configuration Reference*.
- A server application on a particular host is not registered with WLM or is not available. This affects clients that use the group name qualified with the server name (for example, myserver3.myserver.mysplex.mycorp.com).
- The sysplex connections between hosts in the sysplex are not functioning.

Connection problems

If clients attempting to reach servers in your sysplex occasionally get connection timeouts or are unable to access servers in your sysplex, one or more of the following situations are possible:

- The server running at the address given to the client application has been started, but is not totally active due to hardware problems or system definition problems. If this is the problem, refer to the *z/OS Communications Server: IP Configuration Guide*.
- The adapter associated with the address stopped recently, and that information has not yet reached the name server. Because the name server and WLM synchronize their data at one-minute intervals by default (they are not in constant communication), the name server does not learn immediately about stopped adapters. To change the length of the interval, use the named -t option on the **named** command.
- The host owning an unusable address is unreachable in your TCP/IP network. Because WLM and the name server communicate through the sysplex communication mechanisms (sysplex CTCs or XCF) and your TCP/IP network does not, it is possible that the adapter associated with the unusable address is active, but routers in the TCP/IP network cannot reach it. Avoid this type of problem by using VIPA addresses on your sysplex hosts.

Chapter 20. Diagnosing REXEC, REXECD, and RSH problems

This chapter 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 520 for information about REXECD.

The following sections are included:
- "General information about REXEC and RSH"
- "General information about REXECD" on page 520

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 71 shows the principle behind REXECD.

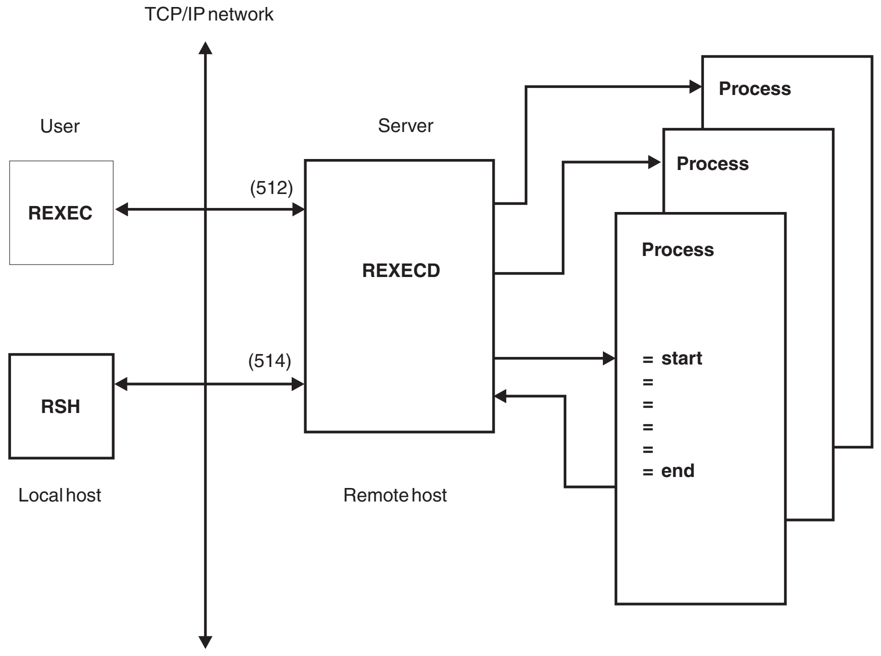


Figure 71. Remote execution protocol principle

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 72 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.

```
rexec -d -l debfox -p mypwd norway time
Established affinity with TCPCS
EZA4801I MVS TCP/IP REXEC CS V1R5
EZA4775I Calling function rexec_af with the following:
Host: norway   user: debfox   cmd: time   port: 512
EZA4774I  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
EZA4789I  rexec complete
```

Figure 72. Example of an REXEC trace

RSH trace example and explanation

Figure 73 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.

```
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 73. Example of an RSH trace

General information about REXECD

The remote execution server allows execution of a TSO batch command that has been received from a remote host. REXECD supports both the remote execution command (REXEC) and remote shell (RSH) client protocols.

Note: When the REXECD 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 74 on page 522 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.

```
EZA4801I MVS TCP/IP REXEC CS V1R5
1
EZA4383I SSCSARAY:0: JOB00043 40
   EZA4383I SSCSARAY:0: JOB00043 80 2
EZA4383I SSCSARAY:0: JOB00043 80
3
EZA4383I SSCSARAY:0: JOB00043 20
   EZA4385I SSSORT(CTRL): 00000000
4
EZA4392I S99ret: 00000000, A RSHD NEWALTON.RSHD5.JOB00043.D0000105.?
5
TIME-12:04:12 PM. CPU-00:00:00 SERVICE-1157 SESSION-00:00:01 MARCH 9,1998
   EZA4393I S99ret: 00000000
6
EZA4390I SSSORT(next): 00000004
```

Figure 74. Example of an REXECD trace of a client using a SEND command

Following are short descriptions of numbered items in the trace:

1 JOB00043 is the JES job number. The 40 indicates 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:

- 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.

2 The 80 indicates 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 executing.

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.

3 The 20 indicates 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. Refer to *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 is the return code expected when there is no more work to do.

Example trace of an RSH client using the SEND command

Figure 75 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
EZA4383I SSCSARAY:0: JOB00043 20
   EZA4385I SSSORT(CTRL): 00000000
   EZA4389I SSSORT(init): 00000004
2
EZA4383I SSCSARAY:1: JOB00044 40
   EZA4383I SSCSARAY:0: JOB00043 20
   EZA4385I SSSORT(CTRL): 00000000
   EZA4389I SSSORT(init): 00000004
3
EZA4383I SSCSARAY:1: JOB00044 80
   EZA4383I SSCSARAY:0: JOB00043 20
   EZA4385I SSSORT(CTRL): 00000000
   EZA4389I SSSORT(init): 00000004
4
EZA4383I SSCSARAY:1: JOB00044 20
   EZA4385I SSSORT(CTRL): 00000000
5
EZA4392I S99ret: 00000000, A RSHD NEWALTON.RSHD5.JOB00044.D0000105.?
6
TIME-12:07:02 PM. CPU-00:00:00 SERVICE-1134 SESSION-00:00:00 MARCH 9,1998
   EZA4393I S99ret: 00000000
7
EZA4390I SSSORT(next): 00000004
```

Figure 75. 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.

Chapter 21. Diagnosing z/OS UNIX REXEC, RSH, REXECD, and RSHD problems

This chapter 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 76:

```
#======================================================================
# service | socket | protocol | wait/ | user | server   | server program
# name    | type   |          | nowait|      | program |   arguments
#======================================================================
#
 shell    stream    tcp       nowait  OMVSKERN /usr/sbin/orshd rshd -l
 exec     stream    tcp       nowait  OMVSKERN /usr/sbin/orexecd rexecd -LV
 otelnet  stream    tcp       nowait  OMVSKERN /usr/sbin/otelnetd otelnetd  -LV
 login    stream    tcp       nowait  bpxroot  /bin/rlogind      rlogind   -d 1
 # Add the following line to enable Kerberos for orshd
 kshell   stream    tcp       nowait  OMVSKERN /usr/sbin/orshd orshd -l -k KRB5
```

Figure 76. Adding applications to /etc/inetd.conf

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 **1**) 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 77.

```
#======================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#======================================================================
#
shell      stream    tcp       nowait OMVSKERN /usr/sbin/orshd rshd -d   2
exec       stream    tcp       nowait OMVSKERN /usr/sbin/orexecd rexecd -d   3
```

Figure 77. Setting traces in /etc/inetd.conf

The traces are turned on for both servers by passing a -d argument to the server programs. **2** is the RSHD server and **3** 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/server.debug.a
```

In this example, the trace data is written to /tmp/syslogd/daemon.debug.a in your hierarchical file system. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about syslogd.

For more information about inetd, refer to *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 `rexec` or `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

```
EZYRC02I Host: 197.22.190.1, user debfox, cmd ls -al, port 1512
```

IPv6

```
EZYRC02I Host: fec0:0:0:12BE::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. EZYRC191 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 `rexecd` or `orexecd`. 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  rexecd.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  rexecd.851981.: EZYRD03I  Remote address = 9.67.113.61
Jun 12 13:31:49  rexecd.851981.: EZYRD05I  clisecport = 1029
Jun 12 13:31:49  rexecd.851981.: EZYRD08I  User is: user21
Jun 12 13:31:49  rexecd.851981.: EZYRD09I  Command is: ls -l
Jun 12 13:31:49  rexecd.851981.: EZYRD12I  Name is: USER21, user is user21
Jun 12 13:31:49  rexecd.851981.: EZYRD13I  dir is: /u/user21
Jun 12 13:31:49  rexecd.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  Clisecport = 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 22. Diagnosing network database system (NDB) problems

The network database system (NDB) allows workstation or mainframe users to issue SQL statements interactively, or to invoke NDB services from within a C application program. NDB services can then be used to pass SQL statements to the DB2 subsystem and handle replies from the DB2 subsystem. The NDB client uses the remote procedure call (RPC) to package the request and issue a remote procedure call that sends the request to the NDB server. The NDB server passes the SQL request to the DB2 subsystem for processing. When processing is complete, the DB2 subsystem passes data or a return code, or data and a return code, to the NDB server, which returns them to the NDB client.

The following sections are included:

- "Documentation for NDB problem diagnosis" on page 533
- "Definitions" on page 533
- "Steps for diagnosing NDB problems" on page 533
- "NDB trace examples and explanations" on page 534

Refer to *z/OS Communications Server: IP Messages Volume 1 (EZA)* for more information about NDB usage.

Tip: Unless specifically shown otherwise, all examples in this chapter are valid for V1R2.

The components of the Network Database System are shown in Figure 78.

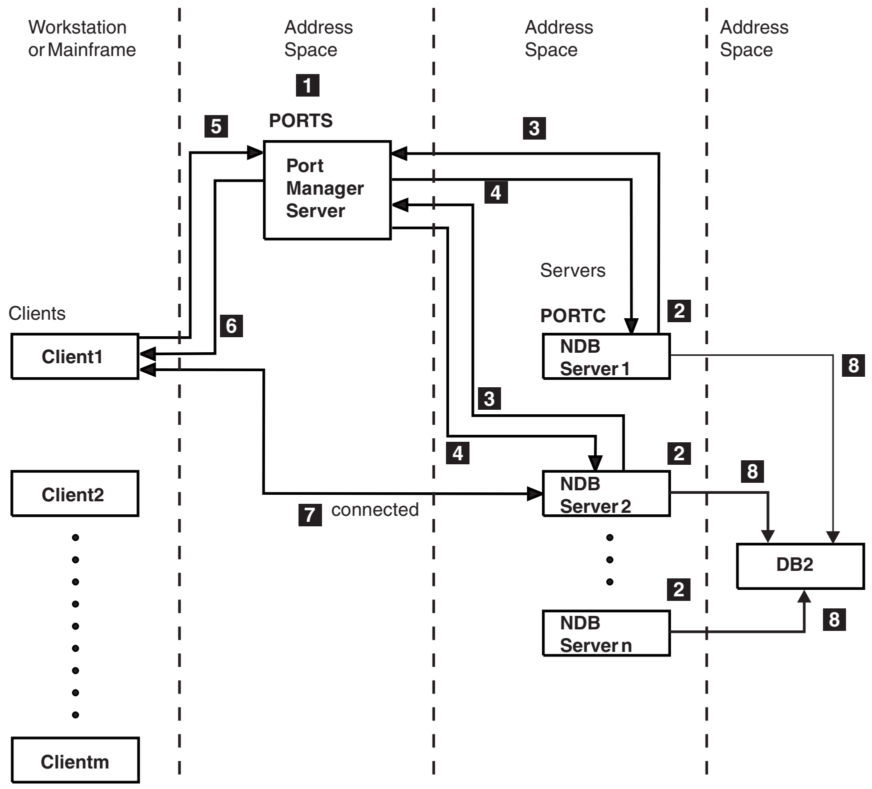


Figure 78. Components of the network database system

Following are a list of steps corresponding to the numbered items in the figure:

1 Bring up the NDB port manager (PORTS). When PORTS is started, it registers its program number with the portmapper, so that portmapper knows on which port PORTS is listening.

2 Bring up the NDB servers (to a maximum of 20). Note that the C multitasking facility is used by PORTC. The number of NDB servers brought up is specified as a startup parameter.

3 Each NDB server issues a request to PORTS for a program number, through the NDB port client (PORTC).

4 PORTS updates its port status and returns a program number.

5 When an NDB client wants NDB services, it calls PORTS at its program number and requests the program number of an available NDB server.

6 PORTS returns the program number of an available NDB server.

7 The NDB client then calls the NDB server with the program number and RPC looks up the port number that is used for the connection.

8 With the connection established, the client can use NDB services to issue SQL requests by using the NDBCLNT command.

Multiple PORTC PROCs can be started, each supporting 1–20 NDB servers. Each PORTC address space can access a different DB2 subsystem. A total of 100 NDB servers, across from 5–100 PORTC address spaces, is supported.

Refer to the *z/OS Communications Server: IP Configuration Reference* for more information about setting up and starting the NDB clients, the NDB port manager, the NDB servers, and the NDB port client.

Documentation for NDB problem diagnosis

The following kinds of information are always required to diagnose an NDB problem:

- Environment description
  - Client environment (for example, OS/2, AIX, or MVS), client level of TCP/IP, and current CSD level for workstation environments
  - Host level of TCP/IP and current maintenance level
- Console output
  - Console output from the NDB server (PORTC)
  - Keystrokes entered, in sequence, from the client side
  - All error messages

    The following trace is requested if needed:

    - DB2 trace

Definitions

Requirements: The following definitions are required for you to use NDB:

- The DB2 subsystem that you intend to use with NDB must be defined.
- Portmapper must be installed and functional.
- The NDB port manager address space must be started.

  The NDB port manager address space consists of one module, PORTMGR.
- The NDB port client and server address spaces must be started.
  - The NDB server address space consists of two modules, PORTCLNT (the NDB port client) and NDBSS (the NDB server). The NDB server code uses the C multitasking facility and can manage from 1–20 NDB servers within this address space. Refer to the *z/OS Communications Server: IP Configuration Reference* for information about configuring and starting the address spaces.
  - For all platforms, except MVS, the NDB client code must be moved to the platform from which the user plans to issue SQL statements, and an executable file must be built. Refer to the *z/OS Communications Server: IP Configuration Reference* for more information.

Steps for diagnosing NDB problems

Before you begin: Gather most of the needed information through return codes.

Perform the following steps to gather information you need:

1. Check the return code and error message. Refer to the *z/OS Communications Server: IP User's Guide and Commands* and *z/OS Communications Server: IP Messages Volume 1 (EZA)* for more information about the return codes.

2. If the return code is +1 or -20000, make sure that the Portmapper is up and running (use RPCINFO), that the path to the host running the DB2 subsystem is available (use PING), and that the DB2 subsystem is up and running (check the MVS system console). Any of these conditions could result in an RPC error or timeout.

3. If the return code is -20100, an incompatibility exists between the NDB client settings of, and the NDB server accepted values for, specific fields of the NDBC control block. Currently accepted settings are given in the return code explanations in the *z/OS Communications Server: IP User's Guide and Commands*. Another possible cause of the problem could be corruption of NDBC control block on either the client or server side. If you believe this is the problem, contact the IBM Software Support Center.

If there are problems obtaining DB2 data from the database, use SPUFI to check the system tables by performing the following analysis steps:

1. **Has a BIND has been issued for DBRM DBUTIL2?**

   To verify a BIND has been issued for DBRM DBUTIL2, issue the following SQL query:

   ```
   select * from sysibm.sysplan where name='EZAND320'
   ```

   If the plan is not found, refer to the *z/OS Communications Server: IP Configuration Reference* for information about binding the DBRM DBUTIL2 (NDBSETUP) to create the plan EZAND320.

2. **Is the TSO user ID that is trying to use the plan EZAND320 authorized?**

   To verify that the TSO user that is trying to use NDB (trying to execute the plan EZAND320) is authorized, issue the following SQL query:

   ```
   select * from sysibm.sysplanauth where name='EZAND320'
   ```

   If neither the user ID executing the plan nor PUBLIC is authorized (that is, listed in the table under the column `grantee`), execute one of the following commands:

   - Grant execute on plan EZAND320 to *user_id*
   - Grant execute on plan EZAND320 to public

   The *user_id* is the TSO user ID that executes EZAND320.

3. **Has the procedure PORTC been updated to point to the correct DB2 load library with the suffix DSNLOAD?**

   Verify that the level of DB2 being used is Communications Server for OS/390 V2R3, or higher. Check the PORTC PROC to ensure that it is pointing to the same subsystem that was specified in the PORTC start up parameter DB2SSID, and that the BIND for DBUTIL2 was completed.

NDB trace examples and explanations

Figure 79 on page 535 shows an example of a trace of the NDB port manager showing the console trace when two NDB servers are started and one NDB client is invoked. It corresponds to the NDB port client and server trace found in Figure 80 on page 537.

```
1
18:58:42 EZA3950I NDB PORT MANAGER FOR CS/390 V2R10 STARTED
2
NDBPS received request. Calling PORTMGR.
   Entering program PORTMGR. SSCB at entry is:
   who is 1. smid is MVSL. suid is SYSADM.
   prognum is 0. portnum is 0. status is 1.
   Entering case: who is NDBSRV(1)
   Entering case: status is NEW(1) or INIT(0)
   Available NDB Server found. Prognum is 536870944
   Exiting program PORTMGR. SSCB at exit is:
   who is 1. smid is MVSL. suid is SYSADM.
   prognum is 536870944. portnum is 0. status is 3.
2
NDBPS received request. Calling PORTMGR.
   Entering program PORTMGR. SSCB at entry is:
   who is 1. smid is MVSL. suid is SYSADM.
   prognum is 0. portnum is 0. status is 1.
   Entering case: who is NDBSRV(1)
   Entering case: status is NEW(1) or INIT(0)
   Available NDB Server found. Prognum is 536870945
   Exiting program PORTMGR. SSCB at exit is:
   who is 1. smid is MVSL. suid is SYSADM.
   prognum is 536870945. portnum is 0. status is 3.
3
NDBPS received request. Calling PORTMGR.
   Entering program PORTMGR. SSCB at entry is:
   who is 2. smid is     . suid is     .
   prognum is 0. portnum is 0. status is 1.
   Entering case: who is NDBCLNT(2)
   Entering case: status is NEW(1)
   Found PORTINFO entry with STATUS of NOT_BUSY. Updating WHO's SSCB fields from PORTINFO entry.
   Exiting program PORTMGR. SSCB at exit is:
   who is 2. smid is MVSL. suid is SYSADM.
   prognum is 536870944. portnum is 0. status is 2.
4
NDBPS received request. Calling PORTMGR.
   Entering program PORTMGR. SSCB at entry is:
   who is 2. smid is     . suid is     .
   prognum is 536870944. portnum is 0. status is 5.
   Entering case: who is NDBCLNT(2)
   Entering case: status is DONE(5)
   Found PORTINFO entry with PROGNUM same as WHO's SSCB PROGNUM.
   Setting STATUS in both to NOT_BUSY and reinitializing other fields of PORTINFO.
   Exiting program PORTMGR. SSCB at exit is:
   who is 2. smid is     . suid is     .
   prognum is 536870944. portnum is 0. status is 3.
```

Figure 79. NDB port manager trace with two NDB servers started and one client invoked

Guidelines:

- The NDB port manager tracing is off by default. To turn it on, the IBM Software Support Center must build a module using the DEF(DEBUG) option and send it to the customer.
- NDB trace output is included in the job log output from the started NDB procedure.

Following are short descriptions of the numbered items in the trace:

1 This message indicates that the NDB port manager procedure has successfully completed startup.

2 The following 10 messages indicate that one NDB server has been started.

These 10 messages are printed out once for each NDB server started, but with each NDB server being assigned a unique program number.

3 The following 10 messages are issued each time an NDB client contacts the NDB port manager for an available NDB server. This is done when an NDB client is first invoked.

4 The following 11 messages are issued each time an NDB client-user issues the NDB END command. The END command indicates to the NDB port manager that this NDB client session has finished and the NDB server associated with it is again available.

Figure 80 on page 537 shows a trace of the NDB port client and NDB servers when two NDB servers are started, and one NDB client is invoked. It corresponds to the NBD port manager trace shown in Figure 79 on page 535.

```
 1
18:59:07 EZA4000I PORTCLNT ENTRY FOR MVS VERSION 3
 2
Tracing is now active. Enjoy your output!
 3
Program PORTCLNT being executed.
    The input parms from startup are as follows:
    7 parms were supplied.
    argv(0), hopefully name of this module, is PORTCLNT
    argv(1), hopefully host name, is MVSL
    argv(2), hopefully userid to run under, is SYSADM
    argv(3), hopefully constant, is NDBSRV
    argv(4), hopefully DB2 subsystem name, is D23
    argv(5), hopefully number of servers to start, is 2
    argv(6), hopefully trace on indictor, is on
 4
About to call clnt_create
    Returned from clnt_create without error
 5
Timeout value is 300
 6
PORTCLNT invoked with Requester NDBSRV
    SSCB of PCb contents after setup:
    WHO is: 1
    SMID is: MVSL
    SUID is: SYSADM
    PROGNUM is: 0(Dec)
    PORTNUM is: 0(Dec)
    STATUS is: 1
 7
PORTCLNT: DB2 name is 3 chars long.
    Copied DB2 name into db2sys, D23
 8
18:59:08 EZA4007I NUMBER OF NDB SERVERS BEING STARTED IS 2
 9
tinit of MTF about to be called
    tinit of MTF successfully called
 10
Server number 1 is starting up
    MVS only code: about to call NDB Port Manager
    Successfully returned from call NDB Port Manager
    SSCB of Result contents after ports_msg_1:
    WHO is: 1
    SMID is: MVSL
    SUID is: SYSADM
    PROGNUM is: 536870944(Dec)
    PORTNUM is: 0(Dec)
    STATUS is: 3
 11
18:59:11 EZA4011I SERVER 1 STARTED. PROGNUM IS 20000020(HEX), 536870944(DEC).
 12
tsched of MTF about to be called
    Parms being passed are:
    result->prognum is 536870944
    db2sys is D23
    trace is 1
```

Figure 80. NDB port client trace with two NDB servers started and one client invoked (Part 1 of 7)

```
 13
Server number 2 is starting up
    MVS only code: about to call NDB Port Manager
    Successfully returned from call NDB Port Manager
    SSCB of Result contents after ports_msg_1:
    WHO is: 1
    SMID is: MVSL
    SUID is: SYSADM
    PROGNUM is: 536870945(Dec)
    PORTNUM is: 0(Dec)
    STATUS is: 3
 14 18:59:11 EZA4011I SERVER 2 STARTED. PROGNUM
    IS 20000021(HEX), 536870945(DEC).
 15 tsched of MTF about to be called     Parms being passed are:     result->prognum is 536870945
    db2sys is D23     trace is 1
 16 18:59:13 EZA4150I NDB SERVER STARTED WITH PROGNUM 20000020(HEX), 536870944(DEC)
 17 Got DB2 name into NDBSS. It is: D23     Now have copied it into db2ssid. It is: D23
     Value of Trace global variable is 1
 18 NDBSRV about to be called on behalf of NDB Client     18:59:36 EZA4151I MVS NDB SERVER
     RECEIVED A CALL FROM HOST USERID user1
 19 Entering program NDBSRV
     Static var NewUser is 1
NDBC contents at NDBSRV entry is:
---------------------------------
ndbrel   is 1
ndbver   is 2
ndbcb    is NDBC
ndbsrc   is 0
ndbappl  is 1
ndbstat  is 0
ndbsname is netdbsrv
ndbusrid is user1
ndbpswd  is not echoed in trace
ndbrqdln is 77
ndbrqd   is 3bb4668 (Hex)
ndbrpdln is 8192
ndbrpd   is 3bc9ff8 (Hex)
ndbrqd contents is:
create table empinfo (empno int, name char(15),salary dec(8,2),hiredate date)
 20 NDBC Reply buffer has been initialized
    NDBC Host userid and password verified
    NDBC Control Block header fields verified
 21 Entered NewUser conditional code
Calling DBOpen from NDBSRV. name is D23
    Entering DBOpen function
    DBUTIL2: ssid is D23  and plan is DBUTIL2
    DB OPEN: rtc is 0 and rsc is 0
    Exiting DBOpen function
    In NDBSRV:Open: rtc is 0.  rsc is 0(Hex).
    CAF OPEN DB was successful.
    End of NewUser conditional code. NewUser is 0.
 22 Processing SQL statement. Calling SQLOpen.
    Entering SQLOpen function
    Value of Init_Done is 0
    Value of LocalStat is 0
    Value of rowBuffer is 0
    rowBufferp is set at 0
    Value of colBytes is 0
    Value of numEntries is 0
    Value of numBytes is 0
```

Figure 80. NDB port client trace with two NDB servers started and one client invoked (Part 2 of 7)

```
23 In SQLOpen, in conditional code for not Init_Done
    End of not Init_Done conditional code
    Value of Init_Done is 1
    Value of numEntries is 60
    Value of numBytes is 2656
 24 SQL variables set up. SQLLEN is 77
     and SQLSTR is <create table empinfo (empno int, name char(15),salary dec(8,2),hiredate date)>
    token, representing type of SQL stmt, is 7
 25 Exiting SQLOpen function
    Value of Init_Done is 1
    Value of LocalStat is 0
    Value of rowBuffer is 0
    rowBufferp is set at 0
    Value of colBytes is 0
    Value of numEntries is 60
    Value of numBytes is 2656
    Back from SQLOpen. RC is  0. NDBSRC is 0
 26 Exiting program NDBSRV
    Static var NewUser is 0
    NDBC contents at NDBSRV exit is:
    --------------------------------
    ndbrel   is 1
    ndbver   is 2
    ndbcb    is NDBC
    ndbsrc   is 0
    ndbappl  is 1
    ndbstat  is 0
    ndbsname is netdbsrv
    ndbusrid is USER1
    ndbpswd  is not echoed in trace
    ndbrqdln is 77
    ndbrqd   is 3bb4668 (Hex)
    ndbrpdln is 8192
    ndbrpd   is 3bc9ff8 (Hex)
 27 NDBSRV about to be called on behalf of NDB Client
    18:59:43 EZA4151I MVS NDB SERVER RECEIVED A CALL FROM HOST USERID user1
19 Entering program NDBSRV
    Static var NewUser is 0
    NDBC contents at NDBSRV entry is:
    ---------------------------------
    ndbrel   is 1
    ndbver   is 2
    ndbcb    is NDBC
    ndbsrc   is 0
    ndbappl  is 1
    ndbstat  is 0
    ndbsname is netdbsrv
    ndbusrid is user1
    ndbpswd  is not echoed in trace
    ndbrqdln is 69
    ndbrqd   is 3bcbf60 (Hex)
    ndbrpdln is 8192
    ndbrpd   is 3bd7ff8 (Hex)
    ndbrqd contents is:
    insert into empinfo values (10001, 'Andersen', 23456.78,'01/02/1983')
```

Figure 80. NDB port client trace with two NDB servers started and one client invoked (Part 3 of 7)

```
20 NDBC Reply buffer has been initialized
    NDBC Host userid and password verified
    NDBC Control Block header fields verified
 22 Processing SQL statement. Calling SQLOpen.
    Entering SQLOpen function
    Value of Init_Done is 1
    Value of LocalStat is 0
    Value of rowBuffer is 0
    rowBufferp is set at 0
    Value of colBytes is 0
    Value of numEntries is 60
    Value of numBytes is 2656
 28 In SQLOpen, in else code, therefore Init_Done
    End of Init_Done code
    Value of Init_Done is 1
    Value of numEntries is 60
    Value of numBytes is 2656
 24 SQL variables set up. SQLLEN is 69
     and SQLSTR is <insert into empinfo values (10001, 'Andersen', 23456.78,'01/02/1983')>
    token, representing type of SQL stmt, is 6
25 Exiting SQLOpen function
    Value of Init_Done is 1
    Value of LocalStat is 0
    Value of rowBuffer is 0
    rowBufferp is set at 0
    Value of colBytes is 0
    Value of numEntries is 60
    Value of numBytes is 2656
    Back from SQLOpen. RC is  0. NDBSRC is 0
26 Exiting program NDBSRV
    Static var NewUser is 0
    NDBC contents at NDBSRV exit is:
    --------------------------------
    ndbrel   is 1
    ndbver   is 2
    ndbcb    is NDBC
    ndbsrc   is 0
    ndbappl  is 1
    ndbstat  is 0
    ndbsname is netdbsrv
    ndbusrid is USER1
    ndbpswd  is not echoed in trace
    ndbrqdln is 69
    ndbrqd   is 3bcbf60 (Hex)
    ndbrpdln is 1
    ndbrpd   is 3bd7ff8 (Hex)
```

Figure 80. NDB port client trace with two NDB servers started and one client invoked (Part 4 of 7)

```
27 NDBSRV about to be called on behalf of NDB Client
    18:59:44 EZA4151I MVS NDB SERVER RECEIVED A CALL FROM HOST USERID user1
 19 Entering program NDBSRV
    Static var NewUser is 0
    NDBC contents at NDBSRV entry is:
    ---------------------------------
    ndbrel   is 1
    ndbver   is 2
    ndbcb    is NDBC
    ndbsrc   is 0
    ndbappl  is 1
    ndbstat  is 0
    ndbsname is netdbsrv
    ndbusrid is user1
    ndbpswd  is not echoed in trace
    ndbrqdln is 21
    ndbrqd   is 3bcbf90 (Hex)
    ndbrpdln is 8192
    ndbrpd   is 3bd7ff8 (Hex)
    ndbrqd contents is:
    select * from empinfo
 20 NDBC Reply buffer has been initialized
    NDBC Host userid and password verified
    NDBC Control Block header fields verified
 22 Processing SQL statement. Calling SQLOpen.
    Entering SQLOpen function
    Value of Init_Done is 1
    Value of LocalStat is 0
    Value of rowBuffer is 0
    rowBufferp is set at 0
    Value of colBytes is 0
    Value of numEntries is 60
    Value of numBytes is 2656
 28 In SQLOpen, in else code, therefore Init_Done
    End of Init_Done code
    Value of Init_Done is 1
    Value of numEntries is 60
    Value of numBytes is 2656
 24 SQL variables set up. SQLLEN is 21
     and SQLSTR is <select * from empinfo>
    token, representing type of SQL stmt, is 5
```

Figure 80. NDB port client trace with two NDB servers started and one client invoked (Part 5 of 7)

```
29 token, representing an SQL SELECT, is 5
    SQL PREPARE using SQLDA was successful
    SQL DECLARE CURSOR was successful
    SQL OPEN CURSOR was successful
    Storage for one row plus indicator vars obtained
    colBytes is 50. rowBuffer is 58.
    Entering SQLFetch function
    Value of LocalStat is 0
    Value of rowBuffer is 58
    rowBufferp is set at 3bd7fb0
    Value of colBytes is 50
    Top of SQLFetch, RowsInBuff is 0, BufferLeft is 8192
    moveRPDp is 3bd7ff8, ndbrpdln is 0
    Starting Do Forever loop
    rowBufferp storage initialized, moveBufferp is 3bd7fb0, and moveBuffer is 0
    SQL FETCH was successful and have formatted a row
    AnyRows is 1, RowsInBuff is 1, moveBuffer is 50, BufferLeft is 8142
    moveRPDp is 3bd802a, and ndbrpdln is 50
    (rowBuffer is 58, rowBufferp is 3bd7fb0)
    Starting Do Forever loop
    rowBufferp storage initialized, moveBufferp is 3bd7fb0, and moveBuffer is 0
    In SQLFetch, after SQL FETCH, sqlcode is 100...but rows were found
    Entering SQLClose function
    Exiting SQLClose function
    End of query, either by EOQ or by error.
    Before reinitializing:
    rowBufferp is 3bd7fb0, LocalStat is 0, AnyRows is 1
    rowBuffer is 58, colBytes is 50
    Exiting SQLFetch function
    Value of LocalStat is 0
    Value of rowBuffer is 0
    rowBufferp is set at 0
    Value of colBytes is 0
 25 Exiting SQLOpen function
    Value of Init_Done is 1
    Value of LocalStat is 0
    Value of rowBuffer is 0
    rowBufferp is set at 0
    Value of colBytes is 0
    Value of numEntries is 60
    Value of numBytes is 2656
    Back from SQLOpen. RC is  0. NDBSRC is 100
```

Figure 80. NDB port client trace with two NDB servers started and one client invoked (Part 6 of 7)

```
26 Exiting program NDBSRV
    Static var NewUser is 0
    NDBC contents at NDBSRV exit is:
    --------------------------------
    ndbrel   is 1
    ndbver   is 2
    ndbcb    is NDBC
    ndbsrc   is 100
    ndbappl  is 1
    ndbstat  is 5
    ndbsname is netdbsrv
    ndbusrid is USER1
    ndbpswd  is not echoed in trace
    ndbrqdln is 21
    ndbrqd   is 3bcbf90 (Hex)
    ndbrpdln is 50
    ndbrpd   is 3bd7ff8 (Hex)
 27 NDBSRV about to be called on behalf of NDB Client
    18:59:46 EZA4151I MVS NDB SERVER RECEIVED A CALL FROM HOST USERID user1
 19 Entering program NDBSRV
    Static var NewUser is 0
    NDBC contents at NDBSRV entry is:
    ---------------------------------
    ndbrel   is 1
    ndbver   is 2
    ndbcb    is NDBC
    ndbsrc   is 100
    ndbappl  is 1
    ndbstat  is 99
    ndbsname is netdbsrv
    ndbusrid is user1
    ndbpswd  is not echoed in trace
    ndbrqdln is 3
    ndbrqd   is 3bcbfa0 (Hex)
    ndbrpdln is 8192
    ndbrpd   is 3bd7ff8 (Hex)
    ndbrqd contents is:     end
20 NDBC Reply buffer has been initialized
    NDBC Host userid and password verified
    NDBC Control Block header fields verified
30 Processing NDB END command
    Entering DBClose function
    DBUTIL2:DB CLOSE: rtc is 0 and rsc is 0
    Exiting DBClose function
    In NDBSRV:Close: rtc is 0. rsc is 0(Hex).
    CAF CLOSE DB was successful.
    End of END command. NewUser is 1. Return.
 16 18:59:13 EZA4150I NDB SERVER STARTED WITH PROGNUM 20000021(HEX), 536870945(DEC)
```

Figure 80. NDB port client trace with two NDB servers started and one client invoked (Part 7 of 7)

Guidelines:

- NDB port client and NDB server tracing is off by default. It can be turned on by specifying the TRACE parameter at PORTC startup with the option ON or YES.
- NDB trace output is included in the job log output from the started NDB procedure.

Following are short descriptions of the numbered items in the trace.

1 This message indicates that the NDB port client is starting up.

2 This trace message indicates that tracing is now active.

3 The following 10 trace messages show the JCL startup parameters specified at PORTC startup.

4 The following two trace messages indicate that the NDB port client was successful at initiating remote procedure call (RPC) communication with the NDB port manager.

5 This trace message indicates that the NDB servers run with an RPC timeout value of five minutes.

6 The following eight trace messages show the input control block SSCB that are used when calling the NDB port manager through RPC.

7 The following two trace messages indicate that the NDB port client was able to obtain the DB2 subsystem name passed by the parameter DB2SSID= of the PORTC procedure and show what value was obtained.

8 This line shows the number of NDB servers specified on the parameter NUMSRV= of the PORTC procedure.

9 The following two trace messages indicate that initialization of NDB servers startup has successfully completed.

10 The following 10 trace messages indicate that NDB server one was successfully assigned a program number by the NDB port manager. The resulting SSCB contents is also shown.

11 The first NDB server has started up. Its assigned program number is shown in hexadecimal and decimal notations.

12 The following five trace messages indicate that NDB server one started and shows the values of the parameters passed to it.

13 The following 10 trace messages indicate that NDB server two was successfully assigned a program number by the NDB port manager. The resulting SSCB contents is also shown.

14 The second NDB server has started up. Its assigned program number is shown in hexadecimal and decimal notations.

15 The following five trace messages indicate that NDB server two started and shows the values of the parameters passed to it.

16 The NDB server has started up and is waiting to be assigned to an NDB client.

17 The following three trace messages indicate that the parameters passed to the NDB server were received and what the values of two of those parameters are. (The third parameter value, program number, is displayed in the previous message.)

18 The following two trace messages indicate the NDB server one was assigned to an NDB client and that it has received a request. The host userid the NDB server is to use when sending the user's request to DB2 is `user1`.

19 The following 19 trace messages show the contents of the input NDBC control block as received by NDB server one.

20 The following three trace messages indicate that the NDB server is ready to start processing the user's request. The host user ID and password supplied by the user with the NDB client have passed the security check and the NDBC control block was verified as valid.

21 The following nine trace messages indicate that this is the first call for this

NDB session. The NDB server must establish a connection with DB2. This is accomplished by opening the plan DBUTIL2 using the DB2 Call Attachment Facility (CAF). The open of DBUTIL2 was successful.

22 The following nine trace messages show the initial values of various internal control fields used in processing the user's request as they are set at the start of request processing.

23 The following five trace messages indicate that this is the first time this NDB server has been called since being started up. An SQLDA (a control block DB2 uses to pass information back to the NDB server about SQL statements sent to DB2) must be allocated and various initial values set.

24 The following three trace messages show the user's SQL statement that are sent to DB2.

25 The following nine trace messages indicate that the user's SQL statement was processed successfully. Also, they show the resulting values of various internal control fields used in processing user requests as they are set at the end of request processing.

26 The following 17 trace messages show the contents of the output NDBC control block that is being sent back to the NDB client.

27 The following two trace messages indicate that NDB server one has received another request from the NDB client.

28 The following five trace messages indicate that NDB server one has been called previously and so only needs to reinitialize certain fields of the SQLDA used by DB2 to pass information to the NDB server.

29 The following 33 trace messages show the path taken to process an SQL SELECT statement. The number of messages and their content vary according to the number of rows returned and columns retrieved by the SQL query. This sequence of messages shows that one row at a time is retrieved from DB2, is formatted and is placed in the NDBC reply buffer. The values of various internal fields used to control processing of the SQL query are displayed.

30 The following seven trace messages indicate that the user has entered the NDB END command to end this NDB session. NDB server one closes the connection with DB2 by issuing a DB2 CAF close call for the plan DBUTIL2. It was successful. An internal indicator is reset so that the next time this NDB server is invoked, it knows it is starting a new NDB session with a new or different assigned NDB client.

Chapter 23. Diagnosing X Window System and Motif problems

This chapter 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 548
- 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 548 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.

```
TRANS(OpenCOTSClient) (/9.2.104.56:0)
TRANS(Open) (1,/9.2.104.56:0)
TRANS(SocketOpenCOTSClient) (inet,9.2.104.56,0)
TRANS(Connect) (3,/9.2.104.56:0)
TRANS(SocketINETConnect) (3,9.2.104.56,0)
TRANS(GetPeerAddr) (3)
TRANS(ConvertAddress)(2,16,7f9d288)
TRANS(SetOption) (3,2,1)
TRANS(SocketWritev) (3,225bc,1)
TRANS(SetOption) (3,1,1)
TRANS(SocketRead) (3,22344,8)
TRANS(SocketRead) (3,22344,8)
TRANS(SocketRead) (3,7f9d368,224)
TRANS(SocketWrite) (3,7f9eb88,60)
TRANS(SocketRead) (3,2249c,32)
TRANS(SocketRead) (3,2249c,32)
TRANS(SocketRead) (3,2249c,32)
TRANS(SocketWrite) (3,7f9eb88,56)
TRANS(SocketRead) (3,22518,32)
TRANS(SocketRead) (3,22518,32)
TRANS(SocketWrite) (3,7f9eb88,80)
TRANS(SocketWrite) (3,7f9eb88,20)
TRANS(SocketRead) (3,223e8,32)
TRANS(SocketRead) (3,223e8,32)
TRANS(SocketDisconnect) (7f9d2c0,3)
TRANS(Close) (3)
TRANS(SocketINETClose) (7f9d2c0,3)
```

Figure 81. Example of X Application trace output when XWTRACE=2

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 XWTRACELC=2

Figure 82 on page 549 is a partial trace showing typical types of information displayed by locale-sensitive routines.

```
lcPubWrap:_XlcCreateLC(C)
 lcCT:_XlcAddCT(ISO8859-1:GL,"(B)
 lcCT:_XlcParseCT
 lcCT:_XlcGetCharSetFromEncoding( (B)
 lcCT:_XlcParseCT returning: 28 charset 0
 lcCharSet:_XlcCreateDefaultCharSet(ISO8859-1:GL,""a)
 lcCT:_XlcParseCharSet
 lcCT:_XlcParseCT
 lcCT:_XlcGetCharSetFromEncoding( (B)
 lcCT:_XlcParseCT returning: 28 charset 0
 lcCharSet:_XlcAddCharSet(ISO8859-1:GL)
 lcCharSet:_XlcGetCharSet(ISO8859-1:GL)
    returned NULL
 lcCT:_XlcAddCT   returning: 7f994d8
⋮
 (trace statements in this section have been deleted)
 lcCT:_XlcAddCT(CNS11643.1986-1:GL,"$(H)
 lcCT:_XlcParseCT
 lcCT:_XlcGetCharSetFromEncoding( $(H)
 lcCT:_XlcParseCT returning: 2428 charset 0
 lcCharSet:_XlcCreateDefaultCharSet(CNS11643.1986-1:GL,"""+)
 lcCT:_XlcParseCharSet
 lcCT:_XlcParseCT
lcCT:_XlcGetCharSetFromEncoding( $(H)
 lcCT:_XlcParseCT returning: 2428 charset 0
 lcCharSet:_XlcAddCharSet(CNS11643.1986-1:GL)
 lcCharSet:_XlcGetCharSet(CNS11643.1986-1:GL)
    returned NULL
 lcCT:_XlcAddCT   returning: 7f9c4e0
 lcCT:_XlcAddCT(TIS620.2533-1:GR,"-T)
 lcCT:_XlcParseCT
 lcCT:_XlcParseCT returning: 80 charset 0
 lcFile:_XlcResolveLocaleName(C,""," ""},"2h",)
 lcFile:_XlcResolveName(C,/usr/lib/X11/locale/locale.alias)
 lcFile:_XlcFileName(7f99420,locale)
 lcFile:_XlcResolveLocaleName(C,,"","","")
 lcFile:_XlcResolveName(C,/usr/lib/X11/locale/locale.alias)
 lcFile:_XlcResolveName(C,/usr/lib/X11/locale/locale.dir)
 lcDB:CreateDatabase(/usr/lib/X11/locale/C/XLC_LOCALE)
```

Figure 82. Example of X Application trace output when XWTRACELC=2 (Part 1 of 2)

```
  0: XLC_XLOCALE, cs0.ct_encoding,      1: ISO8859-1: GL,
  1: XLC_XLOCALE, cs0.wc_encoding,      1: \x00000000,
  2: XLC_XLOCALE, cs0.length,   1: 1,
  3: XLC_XLOCALE, cs0.side,     1: GL:Default,
  4: XLC_XLOCALE, wc_shift_bits,        1: 8,
  5: XLC_XLOCALE, wc_encoding_mask,     1: \x00008080,
  6: XLC_XLOCALE, state_depend_encoding,        1: False,
  7: XLC_XLOCALE, mb_cur_max,   1: 1,
  8: XLC_XLOCALE, encoding_name,        1: STRING,
  9: XLC_FONTSET, fs0.font,     1: ISO8859-1:GL,
 10: XLC_FONTSET, fs0.charset,          1: ISO8859-1:GL,
***
***
  lcDB:_XlcGetResource(7f99420,XLC_XLOCALE,mb_cur_max)
lcDB:_XlcGetLocaleDataBase(7f99420,XLC_XLOCALE,mb_cur_max)
lcDB:_XlcGetResource(7f99420,XLC_XLOCALE,state_dependent)
lcDB:_XlcGetLocaleDataBase(7f99420,XLC_XLOCALE,state_dependent)
   returning NULL
lcDB:_XlcGetResource(7f99420,XLC_XLOCALE,encoding_name)
lcDB:_XlcGetLocaleDataBase(7f99420,XLC_XLOCALE,encoding_name)
   returning lcd=7f99420
lcFile:_XlcResolveLocaleName(C,"",,"",)
lcFile:_XlcResolveName(C,/usr/lib/X11/locale/locale.alias)
```

Figure 82. Example of X Application trace output when XWTRACELC=2 (Part 2 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.

Chapter 24. Diagnosing Simple Network Management Protocol (SNMP) problems

This chapter explains SNMP-related concepts and terms, including information about how to diagnose SNMP problems and contains the following sections:

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 programs (Network SLAPM2 subagent or SLA subagent) for SLA-related MIB variables run as a separate application. The subagent program for TN3270 Telnet MIB variables executes as a separate subtask in the TCP/IP 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 (**osnmp**) and the NetView SNMP command. The **osnmp** 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 **osnmp** command runs in a user address space that is created and removed as **osnmp** 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 engine address space, which provides the protocol support for the SNMP PDUs.

The SNMPIUCV subtask in the NetView address space and the SNMP query engine 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
- SLA subagent that manages the **sla** MIB
- TN3270 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 OSNMP.CONF) 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*.

osnmp

To use **osnmp**, the following files might be needed:

OSNMP.CONF
Defines configuration data for sending SNMPv1, SNMPv2, and SNMPv3 requests to SNMP agents. You can name this file as either an HFS 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 an HFS file or an MVS data set (partitioned or sequential).

SNMP agent

The SNMP agent (osnmpd) 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 45.

Table 45. Configuration files and security types

Data set	SNMPv1 and SNMPv2c	SNMPv1, SNMPv2c, and SNMPv3
PW.SRC	Yes	No
SNMPTRAP.DEST	Yes	No
OSNMPD.DATA	Yes	Yes
SNMPD.CONF	No	Yes
SNMPD.BOOTS	No	Yes

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.

SLA subagent

The SLA subagent is controlled by start options specified when the subagent is started. The following statement options are particularly important:

PAGTSNMP start option -c *community*
Defines the community name to be used in connecting to the SNMP agent.

PAGTSNMP start option -P *port*
Defines the port to be used in connecting to the SNMP agent.

PAGTSNMP start option -d *n*
Specifies the level of tracing used by the SLA subagent. You can also use the MODIFY TRACE,LEVEL command.

TN3270 Telnet subagent

The TN3270 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"
- Connection problems
  - "Problems connecting the SNMP agent to the TCP/IP address space" on page 557
  - "Problems connecting SNMP agents to multiple TCP/IP stacks" on page 557
  - "Problems connecting subagents to the SNMP agent" on page 558
  - "Problems connecting to the SNMPIUCV subtask" on page 562
  - "Problems connecting the SNMP query engine to the TCP/IP address space" on page 563
- Incorrect output
  - "Unknown variable" on page 564
  - "Variable format incorrect" on page 567
  - "Variable value incorrect" on page 568
- "No response from the SNMP agent" on page 570
- "Report received from SNMP agent" on page 571
- "0.0.0.0 address in traps from the SNMP agent" on page 572
- "I/O error for SNMP PING" on page 572
- "Traps not forwarded by trap forwarder daemon" on page 573
- "Incorrect address in forwarded trap" on page 573

Note: A nonzero return code from the SNMP agent indicates an abnormal termination. For more information, use the SNMP agent traces sent to the SYSLOGD output.

Use the information provided in the following sections for problem determination and diagnosis of errors reported against SNMP.

For additional information, refer to the *z/OS Communications Server: IP Configuration Guide* and *z/OS Communications Server: IP Configuration Reference*.

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 MVS Diagnosis: Procedures* or Chapter 3, "Diagnosing abends, loops, and hangs," on page 23, 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
- TCPIP.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 574.

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.DATA information.
  - 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

  - SLA subagent syslogd output obtained by starting the SLA subagent with the -d 2 option or by issuing the MODIFY TRACE,LEVEL command to start SLA subagent tracing (if the subagent is the SLA subagent).
  - TN3270 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 /tmp/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?
   - Is SACONFIG disabled?

__ 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 TN3270 Telnet subagent? If so:
   - Is the TNSACONFIG statement configured correctly?
   - Is TNSACONFIG DISABLED specified?

__ 6. Is the subagent in question the SLA subagent?
   - Does the port number specified on the -P parameter of the SLA subagent match the port number specified by the SNMP agent?
   - Does the community name (or password) specified on the -c parameter of the SLA subagent match the community name (or password) specified by the SNMP agent?

__ 7. If you are using an *hlq*.HOSTS.SITEINFO file (or its HFS 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.

__ 8. 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.

__ 9. Is the port number correct?

__ 10. 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*.

__ 11. 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.

__ 12. 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.

__ 13. 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 chapter 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 the SLA subagent, get the SLA subagent traces. Turn these on by starting the SLA subagent with the -d 2 option or by issuing the MODIFY TRACE,LEVEL command to start SLA 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 TN3270 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 PW.SRC 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 SLA subagent, specify the community name by way of the -c parameter.
    - For the TN3270 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 /tmp/dpi_socket and provides an HFS 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 an onetstat -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` *`xxxx 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 TN3270 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 SLA subagent address space (for SLA subagent problems)
- Dump of user SNMP subagent address space
- Trace from subagent in syslogd

Information on obtaining a dump can be found in *z/OS MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP agent traces is discussed in “Starting SNMP agent traces” on page 575.

Problems connecting to the SNMPIUCV subtask

Problems in connecting the SNMPIUCV subtask of NetView to the SNMP query engine 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 engine:
- PROFILE.TCPIP data set
- SNMP query engine job output, including SYSPRINT output
- NetView log
- SNMPARMS member of DSIPARMS data set

Analysis: Check for problems connecting the SNMP query engine to the NetView SNMPIUCV subtask:
- Has the SNMP query engine job started successfully?
  – Check the SNMP query engine job output for errors. If the SNMP query engine is started successfully, you should see the message:
    ```
    SQEI001 -- SNMP Query Engine 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?

  ```
  SNM101W SNMP task (SNMPIUCV) found Query Engine (name) not ready
  ```

  - Is the *name* that the SNMPIUCV subtask is trying to connect to the correct name for the SNMP query engine 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 engine address space name.

If the problem still occurs after checking the preceding items and making any needed changes, obtain the following documentation:
- SNMP query engine level-two trace output
- SNMP query engine 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 engine 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 574.

Problems connecting the SNMP query engine to the TCP/IP address space

Problems connecting the SNMP query engine 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 engine 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 engine 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 engine 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 574.

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* for more information about syslogd.
- MIBS.DATA, when osnmp is used.
- SNMP query engine 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 **osnmp** command is the client being used, the **osnmp** command configuration file (OSNMP.CONF) might also be needed.
- Include all the configuration files described earlier under "Definitions" on page 553.

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 **osnmp** 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 **osnmp 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:
      - osnmp -v walk saDescription
      - osnmp -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 osnmp 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 **osnmp** command, use the `-d 4` flag to get level four manager traces.
- If the manager is the NetView SNMP command, obtain the SNMP query engine level two output. The SNMP query engine level two trace shows the information flowing between the SNMP query engine and the SNMPIUCV subtask of NetView. Verify in the trace that the variable name being requested is being passed correctly to the SNMP query engine.

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 TN3270 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 SLA subagent address space (for SLA 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 574.

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 osnmp 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 engine 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. osnmp 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 the SLA subagent, syslogd output obtained by the SLA subagent with the -d 2 option or the MODIFY TRACE,LEVEL command to start SLA 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 engine level four trace output or **osnmp** 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 TN3270 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 TN3270 Telnet subagent problems)
- Dump of SNMP agent address space
- Dump of SNMP query engine address space
- Dump of OMPROUTE address space (for OMPROUTE subagent problems)
- Dump of Network SLAPM2 subagent address space (for Network SLAPM2 subagent problems)
- Dump of SLA subagent address space (for SLA 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 574.

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 osnmp 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 engine?

The SNMP query engine 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 TN3270 Telnet subagent? The TN3270 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.

__ 7. Is the variable supported by the SLA subagent? If so, is it being cached? The SLA subagent caches MIB objects for five minutes by default, but the cache time can be overridden at subagent initialization time with the -t parameter.

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 the SLA subagent, syslogd output obtained by the SLA subagent with the -d 2 option or the MODIFY TRACE,LEVEL command to start SLA 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 TN3270 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 TN3270 Telnet subagent variables).
- Dump of SNMP query engine address space
- Dump of OMPROUTE address space (for OMPROUTE subagent problems)

- Dump of Network SLAPM2 subagent address space (for Network SLAPM2 subagent problems)
- Dump of SLA subagent address space (for SLA 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 engine might be useful. You can also use the **onetstat** command to verify that the TCP/IP subagent is reporting what the TCP/IP engine 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 574.

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:

- OMVS console output for any command responses and traces, if osnmp is being used.
- NetView console output or command responses if NetView SNMP is used.
- SNMP agent syslogd output.
- The OSNMP.CONF file (if the **osnmp** 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 **osnmp** 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 **osnmp** command defaults to sending SNMPv1 requests. To send an SNMPv2c request using the **osnmp** command, an entry is required in the OSNMP.CONF file and the **osnmp** 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 SNMPTRAP.DEST or SNMPD.CONF file, then set the snmpEnableAuthenTraps MIB variable to 1, signifying traps are enabled. For detailed information on enabling traps, refer to the *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 TN3270 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 SLA subagent address space (for SLA 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 574.

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 **osnmp** command, some reports occur during normal processing, such as the usmStatsUnknownEngineIDs condition, which occurs as the **osnmp** command performs discovery processing to learn the SNMP engineID of the agent with which it is communicating. Normal processing reports are not displayed by osnmp 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 580 shows the output received by an SNMP manager when the authentication key sent by an **osnmp** command did not match the key defined at the agent. The command issued in the z/OS UNIX shell was:

```
$ osnmp -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* 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."

Analysis

Obtain the following documentation:
- SNMP query engine job SYSPRINT output
- SNMP query engine level two trace output
- SNMP query engine 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 574.

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 572.

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 RECVFROM 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 **osnmp** command, issue osnmp 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 osnmp 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 engine protocol packets exchanged between the SNMP query engine and the SNMPIUCV subtask, with the exception of TRAP packets sent to NetView from the query engine.
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 engine protocol packets for TRAPs sent from the query engine 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 can start the SNMP agent in one of two ways:

- Through the start options in the JCL used to start the SNMP agent (more common). For example,
  ```
  //OSNMPD    EXEC PGM=EZASNMPD,REGION=4096K,TIME=1440,PARM='-d 8'
  ```
- Through OMVS, using the **osnmpd** command. For example:
  ```
  osnmpd -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
32	Internal trace (debug level 1)
64	Extended trace (debug level 2)
128	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 procname. However, if you start the agent from OMVS, 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.

Guideline: While most agent tracing can be modified dynamically, user-friendly formatting of requests in and responses out can be enabled only at agent initialization. This formatting also applies to traps received or forwarded by the Agent.

When this user-friendly formatting is enabled at initialization, it can only be disabled by recycling the agent with debug set to zero. Using the MODIFY command to disable tracing does not prevent inbound requests or outbound responses from being formatted.

For example, assume the procname 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*.

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 671 and "Starting OMPROUTE tracing and debugging using the MODIFY command" on page 672.

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 SLAMP2 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 SLA subagent traces

To start SLA subagent tracing, start the SLA subagent with the -d 2 option or by issuing the MODIFY TRACE,LEVEL command. Output is sent to syslogd.

The following are valid SLA subagent trace levels:

1	General subagent trace information.
2	General subagent trace information, plus extended dump trace and DPI traces. Extended dump trace provides storage dumps of useful information, such as storage returned by the IOCTL calls.

Starting TN3270 Telnet subagent traces

To start the TN3270 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*.

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:

```
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*.

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*.

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*.

Trace examples and explanations

The following examples are shown in this section:
- Agent trace
- 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 **osnmp 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:
Dec 19 15:55:39  snmpagent.9.: OBJECT_IDENTIFIER
Dec 19 15:55:39  snmpagent.9.: 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:
Dec 19 15:55:39  snmpagent.9.: 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 **onetstat -c** command to determine if TCP/IP is running and, if so, which ports are in use.

Figure 85 on page 580 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.S@AU1104.SOURCE.S@AGV123(1624): rc=-65 (SNMP_RC_USM_WRONG_DIGEST)
   from snmp_process_message()
```

Figure 85. SNMP messages and agent trace for nonmatching key

Figure 86 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:

```
osnmp -h v374a -v get usmUserStatus.12.0.0.0.2.0.0.0.0.9.67.35.37.2.117.49
```

```
OUTPUT RECEIVED BY THE MANAGER
usmUserStatus.12.0.0.0.2.0.0.0.0.9.67.35.37.2.117.49 = noSuchObject

AGENT TRACE
IDSTMVS.S@AU1104.SOURCE.S@AGV123(1624): RC=-30 (SNMP_RC_NOT_IN_VIEW)
from snmp_process_message()
```

Figure 86. SNMP messages and agent trace when data not in defined view

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.

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 on page 581 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.

```
Apr  4 16:28:17 MVS097 snmpagent[67108869]: EZZ6225I SNMP agent: Initialization complete
Apr  4 16:28:21 MVS097 M2SubA[50331651]: VS.2575 do_connect_and_open: DPIconnect_to_agent_UNIXstream rc -2.
Apr  4 16:28:21 MVS097 M2SubA[50331651]: VS.1320 do_open_and_register: Connect to SNMP agent failed, will
keep trying
Apr  4 16:28:21 MVS097 M2SubA[50331651]: VS.1340 do_open_and_register: issue selectex, interval = 10 seconds
Apr  4 16:28:31 MVS097 M2SubA[50331651]: VS.2543 do_connect_and_open .... getting agent info from config
Apr  4 16:28:31 MVS097 M2SubA[50331651]: 08B7A4A0  82818497  A6404040  40404040  40404040   *badpw        *
Apr  4 16:28:31 MVS097 M2SubA[50331651]: 08B7A4B0  40404040  40404040  40404040  40404040   *             *
Apr  4 16:28:31 MVS097 M2SubA[50331651]: 08B7A49C  000000A1                                 *....         *
Apr  4 16:28:31 MVS097 M2SubA[50331651]: VS.2556 do_connect_and_open: port 161
Apr  4 16:28:31 MVS097 M2SubA[50331651]: VS.2567 do_connect_and open: hostname_p => 9.67.35.37
Apr  4 16:28:31 MVS097 snmpagent[67108869]: IDSTMVS.SO064350.SOURCE.S@AGV123(1623): rc=-14
(SNMP_RC_NOT_AUTHENTICATED) from snmp_process_message()
Apr  4 16:28:34 MVS097 snmpagent[67108869]: IDSTMVS.SO064350.SOURCE.S@AGV123(1623): rc=-14
(SNMP_RC_NOT_AUTHENTICATED) from snmp_process_message()
```

Figure 87. SNMP subagent trace

SNMP query engine trace

This section discusses the output produced by the SNMP query engine trace.

Figure 88 on page 583 shows an example of the output produced by the SNMP query engine trace. This trace was produced by starting the SNMP query engine 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 engine address space

   Trace output lines in the range 1–3
2. Started the SNMPIUCV subtask at the NetView host (attempted connection to the query engine 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

```
trc  line
lvl  no.

 3    1  EZA6322I Using  'TCPCS.mibdesc.data'  as MIB description file
 0    2  EZA6275I SNMP Query Engine running and awaiting queries...
 2    3  EZA6276I There are 56 client connections possible
 0    4  EZA6290I Accepted new client connection
 4    5  EZA6292I Received following NVquery packet:
      6  EZA6305I dumping packet of 19 bytes:
      7  00 11 01 01 01 02 06 00 00 03 e9 00 00 00 00 00
      8  00 00 00
 2    9  EZA6359I major version:  1
     10  EZA6360I minor version:  1
     11  EZA6361I release:        1
     12  EZA6363I native set:     EBCDIC
     13  EZA6364I packet type:    TRAP REQUEST
     14  EZA6394I  filter id:         1001
     15  EZA6396I  network mask:      0.0.0.0
     16  EZA6397I  desired network:   0.0.0.0
 2   17  EZA6359I major version:  1
     18  EZA6360I minor version:  1
     19  EZA6361I release:        1
     20  EZA6363I native set:     EBCDIC
     21  EZA6364I packet type:    RESPONSE
     22  EZA6367I  sequence id:       1001
     23  EZA6388I  major error:       0
     24  EZA6389I  minor error:       0
     25  EZA6390I  error index:       0
     26  EZA6391I  error text len:    9
     27  EZA6392I  error text:        no error
 4   28  EZA6301I Received following SNMP_trap packet:
     29  EZA6305I dumping packet of 43 bytes:
     30           30 29 02 01 00 04 04 4d 56 53 4c a4 1e 06 0a 2b
     31           06 01 04 01 02 02 01 02 04 40 04 09 43 72 25 02
     32           01 04 02 01 00 43 02 25 80 30 00
 3   33  EZA6424I Decoded SNMP PDU :
     34  {
     35     version version-1,
     36     community '4d56534c'H,
     37     data {
     38        trap {
     39           enterprise 1.3.6.1.4.1.2.2.1.2.4,
     40           agent-addr {
     41              internet '09437225'H
     42           },
     43           generic-trap authenticationFailure,
     44           specific-trap 0,
     45           time-stamp 9600,
     46           variable-bindings {}
     47        }
     48     }
     49  }
 4   50  EZA6359I major version:  1
     51  EZA6360I minor version:  1
     52  EZA6361I release:        1
     53  EZA6363I native set:     EBCDIC
```

Figure 88. SNMP query engine traces (Part 1 of 10)

```
    EZA6364I packet type:        TRAP
    EZA6394I  filter id:         1001
    EZA6395I  agent address:     9.67.114.37
    EZA6399I  generic trap:      4     ( 0X4 )
    EZA6400I  specific trap:     0     ( 0X0 )
    EZA6401I  time stamp:        9600
    EZA6402I  enterprise len:    22
    EZA6403I  enterprise:        1.3.6.1.4.1.2.2.1.2.4
4   EZA6292I Received following NVquery packet:
    EZA6305I dumping packet of 15 bytes:
    00 0d 01 01 01 02 07 00 00 03 ea 00 00 03 e9
2   EZA6359I major version:  1
    EZA6360I minor version:  1
    EZA6361I release:        1
    EZA6363I native set:     EBCDIC
    EZA6364I packet type:    TRAP UN-REQUEST
    EZA6367I  sequence id:       1002
    EZA6394I  filter id:         1001
2   EZA6359I major version:  1
    EZA6360I minor version:  1
    EZA6361I release:        1
    EZA6363I native set:     EBCDIC
    EZA6364I packet type:    RESPONSE
    EZA6367I  sequence id:       1002
    EZA6388I  major error:       0
    EZA6389I  minor error:       0
    EZA6390I  error index:       0
    EZA6391I  error text len:    9
    EZA6392I  error text:        no error
4   EZA6301I Received following SNMP_trap packet:
    EZA6305I 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 40 04 09 43 72 25 02
             01 04 02 01 00 43 02 38 40 30 00
3   EZA6424I 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)

```
    101           variable-bindings {}
    102        }
    103     }
    104  }
 4  105  EZA6292I Received following NVquery packet:
    106  EZA6305I dumping packet of 42 bytes:
    107  00 28 01 01 01 02 01 00 00 03 eb 00 05 d4 e5 e2
    108  d3 00 00 05 e2 d5 d4 d7 00 00 03 01 ff 01 00 0a
    109  e2 e8 e2 e4 d7 e3 c9 d4 c5 00
 2  110  EZA6359I major version:  1
    111  EZA6360I minor version:  1
    112  EZA6361I release:        1
    113  EZA6363I native set:     EBCDIC
    114  EZA6364I packet type:    GET
    115  EZA6367I  sequence id:       1003
    116  EZA6368I  hostname len:      5
    117  EZA6370I  hostname:          MVSL
    118  EZA6371I  community len:     5
    119  EZA6373I  community:         SNMP
    120  EZA6374I  optional length:   3
    121  EZA6375I  max. retries:      1
    122  EZA6376I  initial timeout:   255
    123  EZA6377I  backoff exponent:  1
    124  EZA6380I   name length:      16
    125  EZA6381I   name:             1.3.6.1.2.1.1.3
 3  126  EZA6424I Decoded SNMP PDU :
    127  {
    128     version version-1,
    129     community '534e4d50'H,
    130     data {
    131        get-request {
    132           request-id 1,
    133           error-status noError,
    134           error-index 0,
    135           variable-bindings {
    136              {
    137                 name 1.3.6.1.2.1.1.3,
    138                 value {
    139                    simple {
    140                       empty {}
    141                    }
    142                 }
    143              }
    144           }
    145        }
    146     }
    147  }
    148  EZA6308I sending SNMP request to 9.67.114.37
 4  149  EZA6295I Received following SNMP_response packet:
    150  EZA6305I dumping packet of 39 bytes:
    151           30 25 02 01 00 04 04 53 4e 4d 50 a2 1a 02 01 01
    152           02 01 00 02 01 00 30 0f 30 0d 06 07 2b 06 01 02
    153           01 01 03 43 02 48 a8
 3  154  EZA6424I Decoded SNMP PDU :
```

Figure 88. SNMP query engine traces (Part 3 of 10)

```
    155  {
    156     version version-1,
    157     community '534e4d50'H,
    158     data {
    159        get-response {
    160           request-id 1,
    161           error-status noError,
    162           error-index 0,
    163           variable-bindings {
    164              {
    165                 name 1.3.6.1.2.1.1.3,
    166                 value {
    167                    application-wide {
    168                       ticks 18600
    169                    }
    170                 }
    171              }
    172           }
    173        }
    174     }
    175  }
 2  176  EZA6359I major version:  1
    177  EZA6360I minor version:  1
    178  EZA6361I release:        1
    179  EZA6363I native set:     EBCDIC
    180  EZA6364I packet type:    RESPONSE
    181  EZA6367I  sequence id:       1003
    182  EZA6388I  major error:       0
    183  EZA6389I  minor error:       0
    184  EZA6390I  error index:       0
    185  EZA6391I  error text len:    9
    186  EZA6392I  error text:        no error
    187  EZA6380I   name length:      16
    188  EZA6381I   name:             1.3.6.1.2.1.1.3
    189  EZA6382I   value type:       time ticks
    190  EZA6384I   value length:     4
    191  EZA6387I   value:            18600
 4  192  EZA6292I Received following NVquery packet:
    193  EZA6305I dumping packet of 42 bytes:
    194           00 28 01 01 01 02 02 00 00 03 ec 00 05 d4 e5 e2
    195           d3 00 00 05 e2 d5 d4 d7 00 00 03 01 ff 01 00 0a
    196           c9 c6 c4 c5 e2 c3 d9 4b f0 00
 2  197  EZA6359I major version:  1
    198  EZA6360I minor version:  1
    199  EZA6361I release:        1
    200  EZA6363I native set:     EBCDIC
    201  EZA6364I packet type:    GET-NEXT
    202  EZA6367I  sequence id:       1004
    203  EZA6368I  hostname len:      5
    204  EZA6370I  hostname:          MVSL
    205  EZA6371I  community len:     5
    206  EZA6373I  community:         SNMP
    207  EZA6374I  optional length:   3
    208  EZA6375I  max. retries:      1
    209  EZA6376I  initial timeout:   255
```

Figure 88. SNMP query engine traces (Part 4 of 10)

```
     EZA6377I  backoff exponent:  1
     EZA6380I   name length:      22
     EZA6381I   name:             1.3.6.1.2.1.2.2.1.2.0
3    EZA6424I Decoded SNMP PDU :
     {
        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
4    EZA6295I Received following SNMP_response packet:
     EZA6305I dumping packet of 47 bytes:
              30 2d 02 01 00 04 04 53 4e 4d 50 a2 22 02 01 02
              02 01 00 02 01 00 30 17 30 15 06 0a 2b 06 01 02
              01 02 02 01 02 01 04 07 49 42 4d 20 4c 43 53
3    EZA6424I Decoded SNMP PDU :
     {
        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
                       }
                    }
                 }
              }
           }
        }
     }
2    EZA6359I major version:  1
     EZA6360I minor version:  1
     EZA6361I release:        1
```

Figure 88. SNMP query engine traces (Part 5 of 10)

```
    266  EZA6363I native set:     EBCDIC
    267  EZA6364I packet type:    RESPONSE
    268  EZA6367I  sequence id:       1004
    269  EZA6388I  major error:       0
    270  EZA6389I  minor error:       0
    271  EZA6390I  error index:       0
    272  EZA6391I  error text len:    9
    273  EZA6392I  error text:        no error
    274  EZA6380I   name length:      22
    275  EZA6381I   name:             1.3.6.1.2.1.2.2.1.2.1
    276  EZA6382I   value type:       display string
    277  EZA6384I   value length:     7
    278  EZA6385I   value:            IBM LCS
 4  279  EZA6292I Received following NVquery packet:
    280  EZA6305I dumping packet of 57 bytes:
    281           00 37 01 01 01 02 03 00 00 03 ed 00 05 d4 e5 e2
    282           d3 00 00 05 e2 d5 d4 d7 00 00 03 01 ff 01 00 10
    283           c4 d7 c9 e2 c1 d4 d7 d3 c5 d5 e4 d4 c2 c5 d9 00
    284           00 00 06 f1 f2 f3 f4 f5 00
2   285  EZA6359I major version:  1
    286  EZA6360I minor version:  1
    287  EZA6361I release:        1
    288  EZA6363I native set:     EBCDIC
    289  EZA6364I packet type:    SET
    290  EZA6367I  sequence id:       1005
    291  EZA6368I  hostname len:      5
    292  EZA6370I  hostname:          MVSL
    293  EZA6371I  community len:     5
    294  EZA6373I  community:         SNMP
    295  EZA6374I  optional length:   3
    296  EZA6375I  max. retries:      1
    297  EZA6376I  initial timeout:   255
    298  EZA6377I  backoff exponent:  1
    299  EZA6380I   name length:      22
```

Figure 88. SNMP query engine traces (Part 6 of 10)

```
  300  EZA6381I   name:             1.3.6.1.4.1.2.2.1.4.1
    301  EZA6382I   value type:       number
    302  EZA6384I   value length:     4
    303  EZA6386I   value:            12345
 3  304  EZA6424I Decoded SNMP PDU :
    305  {
    306     version version-1,
    307     community '534e4d50'H,
    308     data {
    309        set-request {
    310           request-id 3,
    311           error-status noError,
    312           error-index 0,
    313           variable-bindings {
    314              {
    315                 name 1.3.6.1.4.1.2.2.1.4.1,
    316                 value {
    317                    simple {
    318                       number 12345
    319                    }
    320                 }
    321              }
    322           }
    323        }
    324     }
    325  }
    326  EZA6308I sending SNMP request to 9.67.114.37
 4  327  EZA6295I Received following SNMP_response packet:
    328  EZA6305I dumping packet of 42 bytes:
    329           30 28 02 01 00 04 04 53 4e 4d 50 a2 1d 02 01 03
    330           02 01 00 02 01 00 30 12 30 10 06 0a 2b 06 01 04
    331           01 02 02 01 04 01 02 02 30 39
 3  332  EZA6424I Decoded SNMP PDU :
    333  {
    334     version version-1,
    335     community '534e4d50'H,
    336     data {
    337        get-response {
    338           request-id 3,
    339           error-status noError,
    340           error-index 0,
    341           variable-bindings {
    342              {
    343                 name 1.3.6.1.4.1.2.2.1.4.1,
    344                 value {
    345                    simple {
    346                       number 12345
```

Figure 88. SNMP query engine traces (Part 7 of 10)

```
                      }
                   }
                }
             }
          }
       }
    }
 2  EZA6359I major version:  1
    EZA6360I minor version:  1
    EZA6361I release:        1
    EZA6363I native set:     EBCDIC
    EZA6364I packet type:    RESPONSE
    EZA6367I  sequence id:       1005
    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:      22
    EZA6381I   name:             1.3.6.1.4.1.2.2.1.4.1
    EZA6382I   value type:       number
    EZA6384I   value length:     4
    EZA6386I   value:            12345
 4  EZA6292I Received following NVquery packet:
    EZA6305I dumping packet of 29 bytes:
             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
 2  EZA6359I major version:  1
    EZA6360I minor version:  1
    EZA6361I release:        1
    EZA6363I native set:     EBCDIC
    EZA6364I packet type:    VAR_NAME
    EZA6367I  sequence id:       1006
    EZA6405I  object id len:     16
    EZA6406I  object id:         1.3.6.1.2.1.1.1
 2  EZA6359I major version:  1
    EZA6360I minor version:  1
    EZA6361I release:        1
    EZA6363I native set:     EBCDIC
    EZA6364I packet type:    RESPONSE
    EZA6367I  sequence id:       1006
    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:             1.3.6.1.2.1.1.1
    EZA6382I   value type:       display string
    EZA6384I   value length:     9
    EZA6385I   value:            sysDescr
 4  EZA6292I Received following NVquery packet:
    EZA6305I dumping packet of 23 bytes:
             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)

```
   401           d3 00 00 03 01 ff 01
2  402  EZA6359I major version:  1
   403  EZA6360I minor version:  1
   404  EZA6361I release:        1
   405  EZA6363I native set:     EBCDIC
   406  EZA6364I packet type:    PING REQUEST
   407  EZA6367I  sequence id:       1007
   408  EZA6368I  hostname len:      5
   409  EZA6370I  hostname:          MVSL
   410  EZA6374I  optional length:   3
   411  EZA6375I  max. retries:      1
   412  EZA6376I  initial timeout:   255
   413  EZA6377I  backoff exponent:  1
2  414  EZA6359I major version:  1
   415  EZA6360I minor version:  1
   416  EZA6361I release:        1
   417  EZA6363I native set:     EBCDIC
   418  EZA6364I packet type:    RESPONSE
   419  EZA6367I  sequence id:       1007
   420  EZA6388I  major error:       0
   421  EZA6389I  minor error:       0
   422  EZA6390I  error index:       0
   423  EZA6391I  error text len:    9
   424  EZA6392I  error text:        no error
   425  EZA6380I   name length:      34
   426  EZA6381I   name:             1.3.6.1.4.1.2.2.1.3.2.9.67.114.37
   427  EZA6382I   value type:       number
   428  EZA6384I   value length:     4
   429  EZA6386I   value:            76
4  430  EZA6292I Received following NVquery packet:
   431  EZA6305I dumping packet of 39 bytes:
   432           00 25 01 01 01 02 01 00 00 03 f0 00 05 d4 e5 e2
   433           d3 00 00 05 e2 d5 d4 d7 00 00 03 01 ff 01 00 07
   434           c2 c1 c4 e5 c1 d9 00
1  435  EZA6356E error code 7: unknown MIB variable
2  436  EZA6359I major version:  1
   437  EZA6360I minor version:  1
   438  EZA6361I release:        1
   439  EZA6363I native set:     EBCDIC
   440  EZA6364I packet type:    GET
   441  EZA6367I  sequence id:       1008
   442  EZA6368I  hostname len:      5
   443  EZA6370I  hostname:          MVSL
   444  EZA6371I  community len:     5
   445  EZA6373I  community:         SNMP
   446  EZA6374I  optional length:   3
   447  EZA6375I  max. retries:      1
   448  EZA6376I  initial timeout:   255
   449  EZA6377I  backoff exponent:  1
   450  EZA6380I   name length:      2
   451  EZA6381I   name:             ?
```

Figure 88. SNMP query engine traces (Part 9 of 10)

```
2  452  EZA6359I major version:  1
    453  EZA6360I minor version:  1
    454  EZA6361I release:        1
    455  EZA6363I native set:     EBCDIC
    456  EZA6364I packet type:    RESPONSE
    457  EZA6367I  sequence id:       1008
    458  EZA6388I  major error:       2
    459  EZA6389I  minor error:       7
    460  EZA6390I  error index:       0
    461  EZA6391I  error text len:    17
    462  EZA6392I  error text:        unknown variable
 0  463  EZA6293I Terminated client connection
```

Figure 88. SNMP query engine traces (Part 10 of 10)

The following is an explanation of the traces in Figure 88 on page 583.

- 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 engine has been successfully started.
- Line 3 is an informational message indicating the number of client connections the query engine allows. (A client connection is a connection from a program using the query engine protocol to communicate with the SNMP query engine 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 engine.
- Lines 5–8 are the encoded packet received from the client (the SNMPIUCV subtask) by the query engine. 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 engine. 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 engine.
- 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 engine 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 engine packet sent from SNMPIUCV to the query engine.
- 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 engine 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 engine receives the trap but does not forward the information to SNMPIUCV.
- Lines 105–109 indicate another request from SNMPIUCV being received by the query engine.
- Lines 110–125 are the decoded query engine 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 engine 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 engine 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 engine protocol requests received from SNMPIUCV by the query engine.
- Lines 197–212 are the decoded version of the query engine request. This is a GetNext request (line 201) to host MVSL (line 204) for variable 1.3.6.1.2.1.2.2.1.2.0 (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 engine 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 engine 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 engine protocol request from SNMPIUCV to the query engine.
- Lines 285–303 are the decoded query engine request. It is a SET request (line 289) to host MVSL to set variable 1.3.6.1.4.1.2.2.1.4.1 (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 engine 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 engine request packet from SNMPIUCV.
- Lines 374–381 are the decoded query engine 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 engine 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 engine request packet from SNMPIUCV.
- Lines 402–413 are the decoded query engine 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 engine 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 engine 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 engine request packet received from SNMPIUCV.
- Line 435 indicates that an error occurred while the query engine was decoding the request packet. The MIB variable name in the request was unknown to the query engine.
- Lines 435–451 are the decoded query engine 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 engine response (line 456) to SNMPIUCV request 1008 (line 457). The request was unsuccessful. The query engine 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 engine 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.

SNMP query engine IUCV communication trace

Figure 89 on page 595 shows an example of the output produced by the IUCV communication trace. This trace was produced by starting the SNMP query engine 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=A0000,
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=2 (Connection Complete)
sock_request_inet entry parms:
   f=0 d=-254 rl=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, path=1, iprcode=0, ipmsgid=C, 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 000054bc
     IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
   f=0 d=-254 rl=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 rl=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, path=1, iprcode=0, ipmsgid=D, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 000054e4
     IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
   f=25 d=3 rl=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 rl=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, path=1, iprcode=0, ipmsgid=E, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=3 buf (ipbfadr2) is at 0000550c
     IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
   f=2 d=3 rl=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
sock_request_inet entry parms:
   f=25 d=4 rl=00000000 rd=0005db44 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, path=1, iprcode=0, ipmsgid=F, 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 00005534
     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 rl=00000000 rd=0005db44 rdl=16 pdh=0 pdl=0
   rc=4 err=0 rpl=00000000 rpb=00000000 rpbl=0
sock_request_inet entry parms:
   f=2 d=4 rl=00000000 rd=0005da9c 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, path=1, iprcode=0, ipmsgid=10, 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 0000555c
     IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
   f=2 d=4 rl=00000000 rd=0005da9c 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
sock_request_inet entry parms:
   f=25 d=5 rl=00000000 rd=0005db64 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=5, path=1, iprcode=0, ipmsgid=11, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
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 rl=00000000 rd=0005db64 rdl=16 pdh=0 pdl=0
   rc=5 err=0 rpl=00000000 rpb=00000000 rpbl=0
sock_request_inet entry parms:
   f=2 d=5 rl=00000000 rd=0005daa8 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=5, path=1, iprcode=0, ipmsgid=12, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
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=2 d=5 rl=00000000 rd=0005daa8 rdl=16 pdh=0 pdl=0
   rc=0 err=0 rpl=00000000 rpb=00000000 rpbl=0
sock_request_inet entry parms:
   f=13 d=5 rl=00000000 rd=00000000 rdl=0 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=13, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 00000000
Rc=1 on IUCV_NEXTBUFF, fd=5 buf (ipbfadr2) is at 000055d4
     IUCV interrupt from TCPIP, fd=-254, path=1 type=7 (Incoming Reply)
sock_request_inet return parms:
   f=13 d=5 rl=00000000 rd=00000000 rdl=0 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 rl=00000000 rd=0005db2c rdl=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=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 rl=00000000 rd=0005db2c rdl=16 pdh=0 pdl=0
   rc=7 err=0 rpl=00000000 rpb=00000000 rpbl=0
 SQEI001 -- SNMP Query Engine running and awaiting queries...
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=15, iucvname=00032508
wait ecblist=5dc5c, ecbcount=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 rl=00000000 rd=00000000 rdl=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=4, 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 rl=00000000 rd=00000000 rdl=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 ecblist=5dc5c, ecbcount=2
iucvposted=1073741824, waitposted=0, callposted=0
in iucvposted
```

Figure 89. SNMP IUCV communication traces (Part 3 of 5)

```
Rc=1 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00005674
     IUCV interrupt, fd=6, path=2 type=1 (Pending Connection)
iucvcomp is now TRUE
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
in iucvcom && iucvselect
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
Rc=0 on IUCV_PURGE to TCPCS   , fd=-254, path=1, iprcode=0, ipmsgid=17, iucvname=00032508
Rc=0 on IUCV_NEXTBUFF, fd=6 buf (ipbfadr2) is at 00000000
Rc=0 on IUCV_ACCEPT to CNMR3X  , fd=8, path=2, iprcode=0, ipmsgid=10000, iucvname=SNMPQE
 SQEI002 -- Accepted new client connection
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
fd=8 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=18, iucvname=00032508
wait ecblist=5dc5c, ecbcount=2
iucvposted=1073741824, waitposted=0, callposted=0
in iucvposted
Rc=1 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 0000569c
     IUCV interrupt, fd=8, path=2 type=3 (Connection Severed)
iucvcomp is now TRUE
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
in iucvcom && iucvselect
fd=3 inetselect now TRUE
fd=6 iucvselect now TRUE
Rc=0 on IUCV_PURGE to TCPCS   , fd=-254, path=1, iprcode=0, ipmsgid=18, iucvname=00032508
fd=8 in callers rmask
Rc=0 on IUCV_NEXTBUFF, fd=-254 buf (ipbfadr2) is at 00000000
fd=8 iucvselect now TRUE
in iucvselect, iucvnfds=1
Rc=0 on IUCV_SEVER to CNMR3X  , fd=8, path=2, iprcode=0, ipmsgid=0, iucvname=SNMPQE
 SQEI003 -- Terminated client connection
```

Figure 89. SNMP IUCV communication traces (Part 4 of 5)

```
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=19, iucvname=00032508
wait ecblist=5dc5c, ecbcount=2
iucvposted=0, waitposted=0, callposted=1073741824
callers ECB posted
Rc=0 on IUCV_PURGE to TCPCS   , fd=-254, path=1, iprcode=0, ipmsgid=19, iucvname=00032508
Rc=0 on IUCV_CLR to , fd=6, path=0, iprcode=0, ipmsgid=0, iucvname=SNMPQE
     ciucv_data area (ipbfadr2) is at 00005480
Rc=0 on IUCV_CLR to TCPCS   , fd=-254, path=0, iprcode=0, ipmsgid=0, iucvname=00032508
     ciucv_data area (ipbfadr2) is at 00000000
```

Figure 89. SNMP IUCV communication traces (Part 5 of 5)

The following sequence of events occurred to create the trace output:
1. Started the SNMP query engine
2. Connected to the query engine from the SNMPIUCV subtask
3. Disconnected the SNMPIUCV subtask from the query engine

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 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]: EZZ8409I TRAPFWD: INITIALIZATION COMPLETE
Oct 15 14:06:07  trapfwd[16777250]: Ready to receive and forward traps....
```

Figure 90. TRAPFWD trace

Chapter 25. Diagnosing Policy Agent problems

The z/OS UNIX Policy Agent (PAGENT) provides administrative control for policies. This chapter provides information and guidance to diagnose Policy Agent problems, and it contains the following sections:

Overview

Policy Agent reads policies defined in local 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 27, "Diagnosing intrusion detection problems," on page 631 for more information about diagnosing IDS policies.
- IPSec

  See Chapter 29, "Diagnosing IP security problems," on page 649 for more information about diagnosing IPSec policies.
- Application Transparent Transport Layer Security (AT-TLS)

  See Chapter 28, "Diagnosing Application Transparent Transport Layer Security (AT-TLS)," on page 637 for more information about diagnosing AT-TLS 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.

TR The policy applies to Traffic Regulation Management. QoS scope TR policies can be defined only in the Policy Agent configuration file. The TR function is part of the Intrusion Detection Services (IDS) and is fully supported by IDS TR policies defined in LDAP. For compatibility, QoS scope TR policies encountered in the Policy Agent configuration file are converted internally by Policy Agent to IDS TR policies.

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 26, "Diagnosing RSVP agent problems," on page 617 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 SLA Subagent to display service policy and mapped application information, as well as to manage and display SLA 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, Network SLAPM2 Subagent, and the SLA Subagent.

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.

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.

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

For details on debug options, refer to *z/OS Integrated Security Services LDAP Client Programming*.

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
- SNMP output from walks of the SLA 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 606
- "LDAP object retrieval problems" on page 611
- "LDAP object storage problems" on page 612
- "Policy Agent/Sysplex distribution problems" on page 614
- "Memory allocation/leakage problems" on page 615

Initialization problems

If Policy Agent does not complete initialization, 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 46 lists some common Policy Agent initialization problems.

Table 46. Common Policy Agent initialization problems

Problem	Cause/action	Symptom
Policy Agent started from a user ID without superuser authority	Policy Agent must be started from a superuser	EZZ8434I message, along with messages in the log file indicating that superuser authority is required and showing an exit code value of 27.
Policy Agent not authorized to security product	Policy Agent must be authorized to a security product profile. Refer to the *z/OS Communications Server: IP Configuration Guide* for details about setting up the proper authorization.	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.
Unable to read configuration file	• 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. • The file must exist. • The permission bits must be correctly set for an HFS file. • Because multiple configuration files might be configured, you might need to check these files also.	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.

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 procname, REFRESH command to restart Policy Agent from the beginning of its configuration files, or MODIFY procname, 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 chapters listed below:

- Intrusion Detection Services (IDS) policy definition problems

  See Chapter 27, “Diagnosing intrusion detection problems,” on page 631 for more information about diagnosing IDS policy definition problems.
- IPSec policy definition problems

  See Chapter 29, “Diagnosing IP security problems,” on page 649 for more information about diagnosing IPSec policy definition problems.
- Application Transparent Transport Layer Security (AT-TLS) policy definition problems

  See Chapter 28, “Diagnosing Application Transparent Transport Layer Security (AT-TLS),” on page 637 for more information about diagnosing AT-TLS policy definition problems.

You might encounter some of the policy definition problems listed in Table 47 on page 607.

Table 47. Policy definition problems

Problem	Cause/action	Symptom
GSKCMS31 DLL not found	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. Policy Agent accesses the GSKCMS31 DLL using the LIBPATH environment variable. 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.	Policy Agent logs a system error message and object error message.
GSKSSL DLL not found	Policy Agent must have access to the GSKSSL DLL at run time. This is needed for AT-TLS policies. 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. 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.	Policy Agent logs a system error message and warning message.
Version 1 QoS policies to version 2 QoS policies conversion Semantic differences exist between version 1 and version 2 policy definitions. **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. See Note 1.	The following circumstances might lead to problems: • 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. • 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. • 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.	Discrepancies between version 1 and version 2 policy definitions.
Policy groups or rules are discarded when defined on an LDAP server.	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. Specify the correct reference Distinguished Names on LDAP objects that reference other objects.	Discarded policy groups or rules

Table 47. Policy definition problems (continued)

Problem	Cause/action	Symptom
Policies with complex conditions (using CNF or DNF) are not mapping correctly.	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 **pasearch** 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 pasearch -o option to also display the original set of specified conditions. This helps to show how the working set was derived.	Difficultly configuring complex policy conditions using CNF or DNF.
Wrong policy being mapped to traffic	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.	Policy rule priority settings are inadequate to control situations where multiple rules map to the same set of traffic.

Table 47. Policy definition problems (continued)

Problem	Cause/action	Symptom
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. 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: 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* 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*. • See Chapter 27, "Diagnosing intrusion detection problems," on page 631 for more information about diagnosing IDS policy definition problems. • See Chapter 29, "Diagnosing IP security problems," on page 649 for more information about diagnosing IP security problems. • See Chapter 28, "Diagnosing Application Transparent Transport Layer Security (AT-TLS)," on page 637 for more information about diagnosing AT-TLS policy definition problems.	Unexpected or missing set of policies.

Table 47. Policy definition problems (continued)

Problem	Cause/action	Symptom
| QoS policies not mapping to the expected traffic	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.	A blank policy rule name is displayed for an active connection using the NETSTAT ALL or **netstat -A** command.
Timing windows when switching policies based on time	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.	Different sets of policies are activated at different times (for example at each shift).
Policies defined in an MVS data set are not being installed.	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.	Parsing errors occur.

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.

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 48, select actions as indicated according the problem you are experiencing.

Table 48. LDAP object retrieval problems

Problem	Cause/action	Symptom
Unable to connect to the LDAP server	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. Also, check that the specified protocol version matches the protocol version being run on the server	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). **Tip:** If a backup LDAP server is configured, the EZZ8440I message is only issued if neither the primary or backup server can be connected.
No objects, or incorrect objects, retrieved from the LDAP server	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 *z/OS Communications Server: IP Configuration Guide* for supported schema object classes.	Missing or incorrect policies are displayed by the **pasearch** command, or the NETSTAT SLAP or **netstat -j** commands.
Wrong set of objects retrieved from the LDAP server	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 SearchPolicyBaseDN, SearchPolicyGroupKeyword, SearchPolicyKeyword, and SearchPolicyRuleKeyword attributes.	Missing or incorrect policies are displayed by the **pasearch** command or the NETSTAT SLAP or **netstat -j** commands.
LDAP DLL not found **Restriction:** Policy Agent must have access to the LDAP DLL at run time.	Check that the LIBPATH environment variable is specified, and that it contains the directory in which the LDAP DLL (GLDCLDAP) resides. This is normally /usr/lib. | Policy Agent accesses the LDAP DLL using the LIBPATH environment variable.	Policy Agent terminates unexpectedly with a CEEDUMP. The reason for termination in the CEEDUMP indicates that the LDAP DLL (GLDCLDAP) was not found.

Table 48. LDAP object retrieval problems (continued)

Problem	Cause/action	Symptom
Version 1 policies not shared among multiple TCP/IP stacks	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.	Version 1 policies not shared among multiple TCP/IP stacks

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 49 on page 613, select actions as indicated according to the problem you are experiencing.

Table 49. LDAP object storage problems

Problem	Cause/action	Symptom

Unable to initialize an LDAPv2 server with the Policy Agent schema definition files	The Policy Agent LDAPv2 schema definition files are shipped as 2 sample files:
	• pagent_at.conf
	• pagent_oc.conf
	
	These files need to be included in the LDAP server's initial configuration file, for example slapd.conf. Check that these files are included using *include* statements like the following:
	`include /path/pagent_at.conf`
	`include /path/pagent_oc.conf`
	
	where */path/* defines the location of the schema definition files on the LDAP server. Verify that the files are readable text 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.

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 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. For more information about installing the schema definition files, refer to *z/OS Communications Server: IP Configuration Guide*.	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 49. LDAP object storage problems (continued)

Problem	Cause/action	Symptom
Unable to add policy objects to an LDAP server	Check the following: 1. Are the Policy Agent schema definitions installed on the LDAP server? See above for more information. 2. Are the proper 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?	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 symptoms listed below correspond to the numbered actions to the left. 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, particularly LDAPv2 implementations, 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.

Policy Agent/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 about Sysplex Distributor Policy Performance Monitoring Configuration in the *z/OS Communications Server: IP Configuration Guide*.

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: rexmit: 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 9, "Diagnosing dynamic VIPA and sysplex problems," on page 311.

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 26. Diagnosing RSVP agent problems

The z/OS UNIX RSVP Agent provides end-to-end resource reservation services on behalf of applications. This chapter provides information and guidance to diagnose z/OS UNIX RSVP Agent problems and contains the following sections:

- "Overview"
- "Policies and RSVP processing" on page 619
- "Gathering diagnostic information" on page 620
- "Diagnosing RSVP agent problems" on page 620

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
- SNMP output from walks of the SLA 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 50:

Table 50. Common RSVP initialization problems

Problem	Cause or action
RSVP Agent not authorized to security product	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.
Unable to read configuration file	Is the correct configuration file specified? Refer to the *z/OS Communications Server: IP Configuration Guide* for the search order used to locate the configuration file. Does the file exist? Are the permission bits correctly set for an HFS file?
Unable to associate with the TCP/IP stack	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.
Unable to initialize interfaces	The RSVP Agent needs to initialize each interface for which it is configured. A pair of ″mailboxes″ are created for each interface. Check for error messages while creating the ″rsvp″ and ″rsvp-udp″ 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.`

Application problems

Before you begin: Determine if a Qos-aware application using RAPI is experiencing problems

If so, check the items listed in Table 51.

Table 51. RSVP application problems

Problem	Cause or action
RAPI DLL not found	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.
Error RAPI_ERR_NORSVP received	If the application receives a RAPI_ERR_NORSVP error code when calling a RAPI function, ensure that the RSVP Agent has been successfully started.

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 52.

Table 52. RSVP policy problems

Problem	Cause or action
RSVP policies not being applied to data flows	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.
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 If the Service Policy Unit is: • 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 25, “Diagnosing Policy Agent problems,” on page 601 for more information about Policy Agent.

Example log file

Figure 91 on page 623 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.

```
 01
03/22 08:51:01 INFO   :.main: *************** RSVP Agent started ***************
 02
03/22 08:51:01 INFO   :...locate_configFile: Specified configuration file: /u/user10/rsvpd1.conf
03/22 08:51:01 INFO   :.main: Using log level 511
03/22 08:51:01 INFO   :..settcpimage: Get TCP images rc - EDC8112I Operation not supported on socket.
 03
03/22 08:51:01 INFO   :..settcpimage: Associate with TCP/IP image name = TCPCS
03/22 08:51:02 INFO   :..reg_process: registering process with the system
03/22 08:51:02 INFO   :..reg_process: attempt OS/390 registration
03/22 08:51:02 INFO   :..reg_process: return from registration rc=0
 04
03/22 08:51:06 TRACE  :...read_physical_netif: Home list entries returned = 7
03/22 08:51:06 INFO   :...read_physical_netif: index #0, interface VLINK1 has address 129.1.1.1, ifidx 0
03/22 08:51:06 INFO   :...read_physical_netif: index #1, interface TR1 has address 9.37.65.139, ifidx 1
03/22 08:51:06 INFO   :...read_physical_netif: index #2, interface LINK11 has address 9.67.100.1, ifidx 2
03/22 08:51:06 INFO   :...read_physical_netif: index #3, interface LINK12 has address 9.67.101.1, ifidx 3
03/22 08:51:06 INFO   :...read_physical_netif: index #4, interface CTCD0 has address 9.67.116.98, ifidx 4
03/22 08:51:06 INFO   :...read_physical_netif: index #5, interface CTCD2 has address 9.67.117.98, ifidx 5
03/22 08:51:06 INFO   :...read_physical_netif: index #6, interface LOOPBACK has address 127.0.0.1, ifidx 0
03/22 08:51:06 INFO   :....mailslot_create: creating mailslot for timer
03/22 08:51:06 INFO   :...mailbox_register: mailbox allocated for timer
 05
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
 06
03/22 08:51:06 WARNING:.....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address not available.
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 129.1.1.1, entity for rsvp allocated and initialized
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 WARNING:.....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address not available.
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 9.37.65.139, entity for rsvp allocated and
initialized
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 WARNING:.....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address not available.
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 9.67.100.1, entity for rsvp allocated and initialized
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 WARNING:.....mailslot_create: setsockopt(MCAST_ADD) failed - EDC8116I Address not available.
03/22 08:51:06 INFO   :....mailbox_register: mailbox allocated for rsvp-udp
03/22 08:51:06 TRACE  :..entity_initialize: interface 9.67.101.1, entity for rsvp allocated and initialized
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 9.67.116.98, entity for rsvp allocated and
initialized
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 9.67.117.98, entity for rsvp allocated and
initialized
```

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
 07
03/22 08:51:11 INFO   :.main: rsvpd initialization complete
 08
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
 09
03/22 08:52:50 EVENT  :.....api_reader: api request SESSION
 10
03/22 08:52:50 TRACE  :......rsvp_event_establishSession: local node will send
03/22 08:52:50 INFO   :........router_forward_getOI: Ioctl to get route entry successful
03/22 08:52:50 TRACE  :........router_forward_getOI:         source address:   9.67.116.98
03/22 08:52:50 TRACE  :........router_forward_getOI:         out inf:   9.67.116.98
03/22 08:52:50 TRACE  :........router_forward_getOI:         gateway:   0.0.0.0
03/22 08:52:50 TRACE  :........router_forward_getOI:         route handle:   7f5251c8
 11
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
 12
03/22 08:52:50 EVENT  :.....api_reader: api request SENDER
 13
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 = 98BDFB0,  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: RSVPFindActionName:  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: RSVPFindActionName:  Result = 0

03/22 08:52:51 INFO   :......rpapi_getPolicyData: RSVPFindActionName:  Exiting

 14
03/22 08:52:51 INFO   :......rpapi_getPolicyData: found action name CLCat2 for
flow[sess=9.67.116.99:1047:6,source=9.67.116.98:8000]
03/22 08:52:51 INFO   :......rpapi_getPolicyData: RSVPFindServiceDetailsOnActName:  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: RSVPFindServiceDetailsOnActName:  Result = 0

03/22 08:52:51 INFO   :......rpapi_getPolicyData: RSVPFindServiceDetailsOnActName:  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   :.......rsvp_flow_stateMachine: state SESSIONED, event PATHDELTA
 15
03/22 08:52:51 TRACE  :........rsvp_action_nHop: constructing a PATH
03/22 08:52:51 TRACE  :........flow_timer_start: started T1
 16
03/22 08:52:51 TRACE  :.......rsvp_flow_stateMachine: entering 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)
 17
03/22 08:52:51 TRACE  :.....rsvp_event: received event from RAW-IP on interface 9.67.116.98
03/22 08:52:51 TRACE  :......rsvp_explode_packet: v=1,flg=0,type=2,cksm=54875,ttl=255,rsv=0,len=84
03/22 08:52:51 TRACE  :.......rsvp_parse_objects: STYLE is WF
03/22 08:52:51 INFO   :.......rsvp_parse_objects: obj RSVP_HOP hop=9.67.116.99, lih=0
03/22 08:52:51 TRACE  :......rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
03/22 08:52:51 INFO   :.......rsvp_flow_stateMachine: state PATHED, event RESVDELTA
 18
03/22 08:52:51 TRACE  :........traffic_action_oif: is to install filter
03/22 08:52:51 INFO   :.........qosmgr_request: src-9.67.116.98:8000 dst-9.67.116.99:1047 proto-6
rthdl-7f5251c8
 19
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)

```
03/22 08:52:51 INFO   :..........rpapi_Reg_UnregFlow: ReadBuffer:  Entering

03/22 08:52:52 INFO   :..........rpapi_Reg_UnregFlow: ReadBuffer:  Exiting

03/22 08:52:52 INFO   :..........rpapi_Reg_UnregFlow: RSVPPutActionName:  Result = 0

03/22 08:52:52 INFO   :..........rpapi_Reg_UnregFlow: RSVPPutActionName:  Exiting

03/22 08:52:52 INFO   :..........rpapi_Reg_UnregFlow: flow[sess=9.67.116.99:1047:6,
source=9.67.116.98:8000] registered with CLCat2
03/22 08:52:52 EVENT  :..........qosmgr_response: RESVRESP from qosmgr, reason=0, qoshandle=8b671d0
03/22 08:52:52 INFO   :..........qosmgr_response: src-9.67.116.98:8000 dst-9.67.116.99:1047 proto-6
03/22 08:52:52 TRACE  :...........traffic_reader: tc response msg=1, status=1
03/22 08:52:52 INFO   :...........traffic_reader: Reservation req successful[session=9.67.116.99:1047:6,
source=9.67.116.98:8000, qoshd=8b671d0]
 20
03/22 08:52:52 TRACE  :........api_action_sender: constructing a RESV
03/22 08:52:52 TRACE  :........flow_timer_stop: stopped T1
03/22 08:52:52 TRACE  :........flow_timer_stop: Stop T4
03/22 08:52:52 TRACE  :........flow_timer_start: started T1
03/22 08:52:52 TRACE  :........flow_timer_start: Start T4
 21
03/22 08:52:52 TRACE  :.......rsvp_flow_stateMachine: entering state RESVED
 22
03/22 08:53:07 EVENT  :..mailslot_sitter: process received signal SIGALRM
03/22 08:53:07 TRACE  :.....event_timerT1_expire: T1 expired
03/22 08:53:07 INFO   :......router_forward_getOI: Ioctl to query route entry successful
03/22 08:53:07 TRACE  :......router_forward_getOI:         source address:   9.67.116.98
03/22 08:53:07 TRACE  :......router_forward_getOI:         out inf:   9.67.116.98
03/22 08:53:07 TRACE  :......router_forward_getOI:         gateway:   0.0.0.0
03/22 08:53:07 TRACE  :......router_forward_getOI:         route handle:   7f5251c8
03/22 08:53:07 INFO   :......rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:53:07 TRACE  :.......rsvp_action_nHop: constructing a PATH
03/22 08:53:07 TRACE  :.......flow_timer_start: started T1
03/22 08:53:07 TRACE  :......rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:07 TRACE  :.......mailslot_send: sending to (9.67.116.99:0)
 23
03/22 08:53:22 TRACE  :.....rsvp_event: received event from RAW-IP on interface 9.67.116.98
03/22 08:53:22 TRACE  :......rsvp_explode_packet: v=1,flg=0,type=2,cksm=54875,ttl=255,rsv=0,len=84
03/22 08:53:22 TRACE  :.......rsvp_parse_objects: STYLE is WF
03/22 08:53:22 INFO   :.......rsvp_parse_objects: obj RSVP_HOP hop=9.67.116.99, lih=0
03/22 08:53:22 TRACE  :......rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
03/22 08:53:22 INFO   :.......rsvp_flow_stateMachine: state RESVED, event RESV
03/22 08:53:22 TRACE  :........flow_timer_stop: Stop T4
03/22 08:53:22 TRACE  :........flow_timer_start: Start T4
03/22 08:53:22 TRACE  :.......rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:22 EVENT  :..mailslot_sitter: process received signal SIGALRM
03/22 08:53:22 TRACE  :.....event_timerT1_expire: T1 expired
03/22 08:53:22 INFO   :......router_forward_getOI: Ioctl to query route entry successful
03/22 08:53:22 TRACE  :......router_forward_getOI:         source address:   9.67.116.98
03/22 08:53:22 TRACE  :......router_forward_getOI:         out inf:   9.67.116.98
03/22 08:53:22 TRACE  :......router_forward_getOI:         gateway:   0.0.0.0
03/22 08:53:22 TRACE  :......router_forward_getOI:         route handle:   7f5251c8
03/22 08:53:22 INFO   :......rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:53:22 TRACE  :.......rsvp_action_nHop: constructing a PATH
03/22 08:53:22 TRACE  :.......flow_timer_start: started T1
03/22 08:53:22 TRACE  :......rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:22 TRACE  :.......mailslot_send: sending to (9.67.116.99:0)
03/22 08:53:38 EVENT  :..mailslot_sitter: process received signal SIGALRM
03/22 08:53:38 TRACE  :.....event_timerT1_expire: T1 expired
03/22 08:53:38 INFO   :......router_forward_getOI: Ioctl to query route entry successful
```

Figure 91. RSVP Agent processing log (Part 4 of 6)

```
03/22 08:53:38 TRACE  :......router_forward_getOI:         source address:   9.67.116.98
03/22 08:53:38 TRACE  :......router_forward_getOI:         out inf:   9.67.116.98
03/22 08:53:38 TRACE  :......router_forward_getOI:         gateway:   0.0.0.0
03/22 08:53:38 TRACE  :......router_forward_getOI:         route handle:   7f5251c8
03/22 08:53:38 INFO   :......rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:53:38 TRACE  :.......rsvp_action_nHop: constructing a PATH
03/22 08:53:38 TRACE  :.......flow_timer_start: started T1
03/22 08:53:38 TRACE  :......rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:38 TRACE  :.......mailslot_send: sending to (9.67.116.99:0)
03/22 08:53:52 TRACE  :.....rsvp_event: received event from RAW-IP on interface 9.67.116.98
03/22 08:53:52 TRACE  :......rsvp_explode_packet: v=1,flg=0,type=2,cksm=54875,ttl=255,rsv=0,len=84
03/22 08:53:52 TRACE  :.......rsvp_parse_objects: STYLE is WF
03/22 08:53:52 INFO   :.......rsvp_parse_objects: obj RSVP_HOP hop=9.67.116.99, lih=0
03/22 08:53:52 TRACE  :......rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
03/22 08:53:52 INFO   :.......rsvp_flow_stateMachine: state RESVED, event RESV
03/22 08:53:52 TRACE  :........flow_timer_stop: Stop T4
03/22 08:53:52 TRACE  :........flow_timer_start: Start T4
03/22 08:53:52 TRACE  :.......rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:53 EVENT  :..mailslot_sitter: process received signal SIGALRM
03/22 08:53:53 TRACE  :.....event_timerT1_expire: T1 expired
03/22 08:53:53 INFO   :......router_forward_getOI: Ioctl to query route entry successful
03/22 08:53:53 TRACE  :......router_forward_getOI:         source address:   9.67.116.98
03/22 08:53:53 TRACE  :......router_forward_getOI:         out inf:   9.67.116.98
03/22 08:53:53 TRACE  :......router_forward_getOI:         gateway:   0.0.0.0
03/22 08:53:53 TRACE  :......router_forward_getOI:         route handle:   7f5251c8
03/22 08:53:53 INFO   :......rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:53:53 TRACE  :.......rsvp_action_nHop: constructing a PATH
03/22 08:53:53 TRACE  :.......flow_timer_start: started T1
03/22 08:53:53 TRACE  :......rsvp_flow_stateMachine: reentering state RESVED
03/22 08:53:53 TRACE  :.......mailslot_send: sending to (9.67.116.99:0)
03/22 08:54:09 EVENT  :..mailslot_sitter: process received signal SIGALRM
03/22 08:54:09 TRACE  :.....event_timerT1_expire: T1 expired
03/22 08:54:09 INFO   :......router_forward_getOI: Ioctl to query route entry successful
03/22 08:54:09 TRACE  :......router_forward_getOI:         source address:   9.67.116.98
03/22 08:54:09 TRACE  :......router_forward_getOI:         out inf:   9.67.116.98
03/22 08:54:09 TRACE  :......router_forward_getOI:         gateway:   0.0.0.0
03/22 08:54:09 TRACE  :......router_forward_getOI:         route handle:   7f5251c8
03/22 08:54:09 INFO   :......rsvp_flow_stateMachine: state RESVED, event T1OUT
03/22 08:54:09 TRACE  :.......rsvp_action_nHop: constructing a PATH
03/22 08:54:09 TRACE  :.......flow_timer_start: started T1
03/22 08:54:09 TRACE  :......rsvp_flow_stateMachine: reentering state RESVED
03/22 08:54:09 TRACE  :.......mailslot_send: sending to (9.67.116.99:0)
03/22 08:54:22 TRACE  :.....rsvp_event: received event from RAW-IP on interface 9.67.116.98
03/22 08:54:22 TRACE  :......rsvp_explode_packet: v=1,flg=0,type=2,cksm=54875,ttl=255,rsv=0,len=84
03/22 08:54:22 TRACE  :.......rsvp_parse_objects: STYLE is WF
03/22 08:54:22 INFO   :.......rsvp_parse_objects: obj RSVP_HOP hop=9.67.116.99, lih=0
03/22 08:54:22 TRACE  :......rsvp_event_mapSession: Session=9.67.116.99:1047:6 exists
03/22 08:54:22 INFO   :.......rsvp_flow_stateMachine: state RESVED, event RESV
03/22 08:54:22 TRACE  :........flow_timer_stop: Stop T4
03/22 08:54:22 TRACE  :........flow_timer_start: Start T4
03/22 08:54:22 TRACE  :.......rsvp_flow_stateMachine: reentering state RESVED
03/22 08:54:24 EVENT  :..mailslot_sitter: process received signal SIGALRM
03/22 08:54:24 TRACE  :.....event_timerT1_expire: T1 expired
03/22 08:54:24 INFO   :......router_forward_getOI: Ioctl to query route entry successful
03/22 08:54:24 TRACE  :......router_forward_getOI:         source address:   9.67.116.98
03/22 08:54:24 TRACE  :......router_forward_getOI:         out inf:   9.67.116.98
03/22 08:54:24 TRACE  :......router_forward_getOI:         gateway:   0.0.0.0
03/22 08:54:24 TRACE  :......router_forward_getOI:         route handle:   7f5251c8
```

Figure 91. RSVP Agent processing log (Part 5 of 6)

```
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
 24
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
 25
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

03/22 08:54:35 INFO   :..........rpapi_Reg_UnregFlow: RSVPRemActionName:  Result = 0

03/22 08:54:35 INFO   :..........rpapi_Reg_UnregFlow: RSVPRemActionName:  Exiting

03/22 08:54:35 INFO   :..........rpapi_Reg_UnregFlow: flow[sess=9.67.116.99:1047:6,
source=9.67.116.98:8000] unregistered from CLCat2
03/22 08:54:35 EVENT  :..........qosmgr_response: DELRESP from qosmgr, reason=0, qoshandle=0
03/22 08:54:35 INFO   :..........qosmgr_response: src-9.67.116.98:8000 dst-9.67.116.99:1047 proto-6
03/22 08:54:35 TRACE  :...........traffic_reader: tc response msg=3, status=1
 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)
03/22 08:54:35 TRACE  :......rsvp_event_propagate: flow[session=9.67.116.99:1047:6,
source=9.67.116.98:8000] ceased
 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 PROTERR:.......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)

Following are short descriptions of the numbered items in the trace:

01 The RSVP Agent is started.

02 The configuration file being used is reported.

03 The name of the TCP/IP stack that the RSVP Agent associates itself with is reported.

04 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.

05 The interfaces are initialized one by one.

06 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.

07 RSVP Agent initialization is complete.

08 An application makes its first RAPI call, initializing the RAPI interface with the RSVP Agent.

09 The type of RAPI request is SESSION, meaning a rapi_session() call was made.

10 The RSVP Agent determines what the application sends based on the specified destination address not being a local interface.

11 The outbound interface to use for the session is returned from the stack.

12 The application issues a rapi_sender() call, passing the Tspec.

13 The Policy Agent interface is initialized.

14 The policy action ″CLCat2″ is obtained from the Policy Agent for the specified flow.

15 The RSVP Agent constructs an RSVP PATH packet to be sent to the destination.

16 The flow enters the pathed stated (PATHED), meaning a PATH packet has been sent for the flow.

17 An RSVP RESV packet is received from the RSVP Agent at the receiver node, specifying the reservation parameters.

18 The RSVP Agent installs the reservation request into the TCP/IP stack and registers the flow with the Policy Agent.

19 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).

20 The RESV packet values are passed to the sender application.

21 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).

22 A T1 timeout occurs, meaning a PATH refresh packet is sent. This occurs every 15 seconds.

23 A refreshed RESV packet is received from the RSVP Agent at the receiver node. This occurs every 30 seconds.

24 The application issues a rapi_release() call to end the RAPI session.

25 The reservation is removed from the TCP/IP stack and unregistered from the Policy Agent.

26 A PATHTEAR packet is constructed and sent, to tear down the flow along the data path.

27 The flow enters the sessioned state (SESSIONED), meaning that the flow has been torn down.

28 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.

29 A SIGTERM signal is received (due to a **kill** command issued from the UNIX shell), and the RSVP Agent shuts itself down.

Chapter 27. Diagnosing intrusion detection problems

This chapter 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 632
- "Diagnosing TRMD problems" on page 635
- "Documentation for the IBM Software Support Center" on page 636

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 -i** (refer to *z/OS Communications Server: IP System Administrator's Commands*) to see what IDS policies are active in Policy Agent.

See Chapter 25, "Diagnosing Policy Agent problems," on page 601 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* and the section about the IDS attribute matrix in the *z/OS Communications Server: IP Configuration Reference*.

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 idsMaxEventMessage attribute. If specified, this limits the number of times the same idsAttackType 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 idsTRtcpTotalConnections limit, is reported during each port constrained period.
- Only the first connection denied, when a source host exceeds the idsTRtcpPercentage 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 idsTRtcpPercentage 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 136 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 89 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.

```
EZZ8761I IDS EVENT DETECTED
EZZ8762I EVENT TYPE: SCAN STORAGE CONSTRAINED
EZZ8763I CORRELATOR 0 - PROBEID 0300FFF3
EZZ8766I IDS RULE N/A
EZZ8767I 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
    ICMPRULENAME: 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: SCAN INTERVAL OVERRUN
EZZ8763I CORRELATOR 0 - PROBEID 0300FFF5
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 20 - PROBEID 04070015
.EZZ8770I INTERFACE OSAQDIO4L
EZZ8765I DESTINATION IP ADDRESS 5.72.107.78 - PORT 0
......EZZ8766I IDS RULE AttackFlood-rule
.EZZ8767I IDS ACTION AttackLog-action
```

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
IEA995I 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=0000
   NAME=CEEBINIT
   DATA AT PSW  0000AA74 - 00181610  0A0D47F0  B10A1811
   GR 0: 84000000   1: 84000FFD
      2: 00000090   3: 00000001
        4: 0001C2A0   5: 0001C144
        6: 00016560   7: 000169D0
        8: 00000016   9: 098E374E
        A: 00000004   B: 8000A9A8
        C: 00017AC0   D: 0001C018
        E: 00000000   F: 00000090
   END OF SYMPTOM DUMP
  IEF450I TRMD TRMD - ABEND=S000 U4093 REASON=00000090
          TIME=12.48.58
  IEF404I TRMD - ENDED - TIME=12.48.58
  $HASP395 TRMD     ENDED
CEE5101C During initialization, the OpenEdition callable service BPX1MSS
         failed. The system return code was 0000000156 , the reason code was
         0B0C00F9 . The application will be terminated.
```

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 635.
- Start CTRACE in the stack to gather related information. See "Component trace" on page 41.
- 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 28. 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 chapter 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 639
- "AT-TLS return codes" on page 642
- "SIOCTTLSCTL ioctl return codes" on page 646

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 a TTLSConfig statement.
  - 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* 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* 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*. Ensure that your AT-TLS policies are correctly defined. Refer to the AT-TLS information in *z/OS Communications Server: IP Configuration Guide* and the AT-TLS Policy statements in *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

TTLSConnectionAction statement that has the Trace statement set to 255 (All). If configuring using the z/OS Network Security Configuration Assistant, 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 TTLSGroupAdvancedParms 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 z/OS Network Security Configuration Assistant, 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 TTLSGroupAction, TTLSEnvironmentAction or TTLSConnectionAction 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 z/OS Network Security Configuration Assistant, 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 53 lists the trace level, the generated AT-TLS messages, and the syslog priority.

Table 53. AT-TLS trace levels

Trace level	Traced information	Syslog priority
1 Error (to Joblog)	EZD1287I	NA
2 Error	EZD1286I	err
4 - Info	EZD1281I, EZD1283I	info
8 - Event	EZD1282I, EZD1283I	debug
16 - Flow	EZD1282I, EZD1283I, EZD1284I	debug
32 - Data	EZD1285I	debug

Tip: Setting the Trace level to 6 enables both error messages and info messages.

The information messages trace when a 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 641 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
                           JOBNAME: FTPD2 USERID: FTPD TYPE: InBound STATUS: Enabled RULE: ftp_serv_21
                           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: 00000002 CONNID: 00000025  RC:    0 Call
                           GSK_ENVIRONMENT_OPEN - 7F1DB058
11:10:28 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000002 CONNID: 00000025  RC:    0 Set
                           GSK_KEYRING_FILE -  FTPDsafkeyring   3
11:10:28 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000002 CONNID: 00000025  RC:    0 Set
                           GSK_CLIENT_AUTH_TYPE -  FULL
11:10:28 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000002 CONNID: 00000025  RC:    0 Set
                           GSK_SESSION_TYPE -  SERVER
11:10:28 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000002 CONNID: 00000025  RC:    0 Set
                           GSK_PROTOCOL_SSLV2 -  ON
11:10:28 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000002 CONNID: 00000025  RC:    0 Set
                           GSK_PROTOCOL_SSLV3 -  ON
11:10:28 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000002 CONNID: 00000025  RC:    0 Set
                           GSK_PROTOCOL_TLSV1 -  ON
11:10:28 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000002 CONNID: 00000025  RC:    0 Set
                           GSK_IO_CALLBACK -
11:10:28 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000002 CONNID: 00000025  RC:    0 Set
                           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
11:10:28 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000002 CONNID: 00000025  RC:    0 Set
                           GSK_SSL_HW_DETECT_MESSAGE -  NULL
11:10:28 TCPCS3   EZD1283I TTLS Event GRPID: 00000001 ENVID: 00000002 CONNID: 00000000  RC:    0
  Environment Master
                           Init 7F1DB058
11:10:28 TCPCS3   EZD1283I TTLS Event GRPID: 00000001 ENVID: 00000001 CONNID: 00000000  RC:    0
  Environment
                           Link 7F1DB058 00000002
11:10:28 TCPCS3   EZD1282I TTLS Start GRPID: 00000001 ENVID: 00000001 CONNID: 00000025 Initial Handshake
                           ACTIONS: grp_act1 env_act_serv **N/A** HS-Server    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
```

```
11:10:28 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 RECV CIPHER 807A010301  5
11:10:28 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 RECV CIPHER
   00510000002000000401008000000500002F000033000032
   00000A0700C000001600001300000906004000001500001200000302008000000800001400001100000010000
   020000180000340001B00001A00001700001941E69D75F7DCB55234895D884B271253A522E4BE211250F546
                          4FE5C5AB980FBD
11:10:28 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 SEND CIPHER
   16030102900200004603014 1E69D753469372857A71168D9
   9D7B93AD6CC30F6F6BDF7F774929CD4D2E8E2A2000000026091B9AAB04F7000000000000000000000000000
   41E69D75000000010005000B00023E00023B000238308202343082019DA003020102020100300D06092A8648
   86F70D0101050500302E310B3009060355040613027573311030 0E060355040B130774657374696E67310D30
   0B0603550403130446545044301E170D303430383039303430 30305A170D3035303831303   6
11:10:28 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 RECV CIPHER 1603010086
11:10:28 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 RECV CIPHER
   10000082008037A6573A4C160A8C0810C542A1CEB73A9FF5
   899D767711EF3BF86D4C2D2743837AA4D5E247DE35F79C8A71A9E6A18DF8CC845D5E0F8F386DF84D746A4004
   B641C14DD7A002FAC5538ED52E3194C2ADE6010381BFC70D1CA6D9F34EDC0F345F0A015575A6C9D85602B1BF
   2877760BA91FC6296625A16A274426112C65DB7A2685
11:10:29 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 RECV CIPHER 1403010001
11:10:29 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 RECV CIPHER 01
11:10:29 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 RECV CIPHER 1603010024
11:10:29 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 RECV CIPHER
   789DBBACAE9D6F19F62B1AF2B529B1850F7057A6EDDE64CD 2301D91CA43C4EBBB5A3DFE5
11:10:29 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 SEND CIPHER 140301000101
11:10:29 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 SEND CIPHER
   603010024FE6548CCBA0D820D73FF439A6B475B4116BCE4
   6FF225DAE1A0F7EC2AEA4690595E63F036
11:10:29 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000001 CONNID: 00000025  RC:    0 Call
                          GSK_SECURE_SOCKET_INIT - 7F0CA118
11:10:29 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000001 CONNID: 00000025  RC:    0 Get
                          GSK_CONNECT_SEC_TYPE -  TLSV1
11:10:29 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000001 CONNID: 00000025  RC:    0 Get
                          GSK_CONNECT_CIPHER_SPEC -  05
11:10:29 TCPCS3   EZD1283I TTLS Event GRPID: 00000001 ENVID: 00000001 CONNID: 00000025  RC:    0
   Initial Handshake
                          7F0CA118 7F1DB058 TLSV1 05 7
11:11:05 TCPCS3   EZD1285I TTLS Data  CONNID: 00000025 SEND CIPHER
   1503010016D47A7AEC70D317976ACEEF3418CDCC8B2DF7
   D3491D     8
11:11:13 TCPCS3   EZD1283I TTLS Event GRPID: 00000001 ENVID: 00000001 CONNID: 00000025  RC: 0 Receive
   Reset
11:11:13 TCPCS3   EZD1282I TTLS Start GRPID: 00000001 ENVID: 00000001 CONNID: 00000025 Connection Close
                          ACTIONS: grp_act1 env_act_serv **N/A**   9
11:11:13 TCPCS3   EZD1284I TTLS Flow  GRPID: 00000001 ENVID: 00000001 CONNID: 00000025  RC: 0 Call
                          GSK_SECURE_SOCKET_CLOSE - 7F0CA118
11:11:13 TCPCS3   EZD1283I TTLS Event GRPID: 00000001 ENVID: 00000001 CONNID: 00000025  RC: 0 Connection
                          Close 7F0CA118 7F1DB058
```

Figure 92. Example trace of a generic server processing

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 below 5000 are defined by System SSL. Refer to *z/OS Cryptographic Service System Secure Sockets Layer Programming* for additional information on these return codes. Return codes above 5000 are defined by AT-TLS. See Table 55 on page 644 for information about these return codes.

Table 54 lists some common System SSL return codes and possible causes.

Table 54. Common System SSL return codes

Return code	Event	Possible cause and solution
202	Environment Init	The key ring cannot be opened because the user does not have permission. Check the following: • 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 z/OS Network Security Configuration Assistant, you can specify the key ring on either the AT-TLS: Image Level Settings panel or on each Traffic Descriptor. • Ensure that the correct key ring has been specified. • If using a RACF key ring, verify that all the steps in *z/OS Communications Server: IP Configuration Guide* have been followed for this user ID.

Table 54. Common System SSL return codes (continued)

Return code	Event	Possible cause and solution
402	Connection Init	A SSL cipher suite could not be agreed upon between the client and server. Check the following: • If V2Ciphers or V3Ciphers are coded, verify that the remote end supports at least one of the cipher suites coded. If configuring using the z/OS Network Security Configuration Assistant, the ciphers are selected for each Security Level. • 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. • For ciphers defined as exportable, verify that the proper FMIDs to support the encryption level are installed.
406	Connection Init	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: • Ensure that the remote partner is enabled for secure connections. • Determine whether the secure negotiation has completed. Use the AT-TLS Data trace level to determine this. • Verify that the TCP data flows were sent by the remote partner. Use a TCP/IP packet trace to verify this.
412	Connection Init	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: • Determine the protocols supported by the remote partner. • Code a TTLSEnvironmentAdvancedParms statement, which enables the common protocols. If configuring using the z/OS Network Security Configuration Assistant, use a Security Level with cipher levels supported by the remote partner.
422	Connection Init	A v3Cipher that is not valid has been found. Check the following: • Determine whether the v3Cipher statement has been coded. • Verify that the proper SSL FMIDs are installed to support the ciphers specified.

Table 54. Common System SSL return codes (continued)

Return code	Event	Possible cause and solution
434	Connection Init	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 z/OS Network Security Configuration Assistant, the ciphers are selected in each Security Level.

Table 55 lists some common AT-TLS return codes and possible causes.

Table 55. AT-TLS return codes

Return code	Event	Possible cause and solution
5001	Connection Init	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.
5002	Connection Init	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*.
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 z/OS Network Security Configuration Assistant, the Application Controlled setting is done in each Traffic Descriptor.

Table 55. AT-TLS return codes (continued)

Return code	Event	Possible cause and solution
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 TTLSEnvironmentAction or the TTLSConnectionAction statement. If configuring using the z/OS Network Security Configuration Assistant, 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 z/OS Network Configuration Assistant the Handshake, 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 55. AT-TLS return codes (continued)

Return code	Event	Possible cause and solution
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.

SIOCTTLSCTL ioctl return codes

The SIOCTTLSCTL ioctl provides the interface for an application to query and control AT-TLS. Table 56 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 56. SIOCTTLSCTL error codes

Errno	Errnojr	IOCTL request specified (1)	Condition causing Error	Valid Data? (2)
EAcces	JrConnDeniedPolicy	INIT_CONNECTION, RESET_SESSION, RESET_CIPHER	Mapped policy indicates that the application cannot request AT-TLS security for the connection (ApplicationControlled Off)	Yes
EAlready	JrAlreadyActive	INIT_CONNECTION	An INIT_CONNECTION request was previously received for the connection	Yes

Table 56. SIOCTTLSCTL error codes (continued)

Errno	Errnojr	IOCTL request specified (1)	Condition causing Error	Valid Data? (2)
EConnReset	JrTTLSHandshakeFailed	Any	Initial handshake was in progress and socket is a blocking socket. Request blocked for handshake to complete. Handshake failed.	No
EInProgress	JrOK	INIT_CONNECTION	Initial handshake has been started and socket is a non-blocking socket. (3)	Yes
EInval	JrInvalidVersion	Any	Bad ioctl version number specified.	No
EInval	JrSocketCallParmError	Any	Length of input data is not length of ioctl structure.	No
EInval	JrSocketCallParmError	Not valid	Request type specified is not valid.	No
EInval	JrSocketCallParmError	RETURN_CERTIFICATE	Certificate buffer pointer = 0 or certificate buffer length = 0.	No
EInval	JrSocketCallParmError	! RETURN_CERTIFICATE	Certificate buffer pointer != 0 or certificate buffer length != 0.	No
EMVSErr	JrUnexpectedErr	Any	Policy was not mapped prior to ioctl call and an error was encountered upon policy map during ioctl call.	No
ENoBufs	JrBuffTooSmall	RETURN_CERTIFICATE	The certificate buffer provided is too small.	Yes (4)
ENotConn	JrGetConnError	Any	TCP connection is not yet in Established state or has been reset.	No
EOpNotSupp	JrOptNotSupported	INIT_CONNECTION, RESET_SESSION, RESET_CIPHER	Mapped policy indicates that AT-TLS is not enabled for the connection (TTLSEnabled Off).	Yes
EPerm	JrSocketCallParmError	INIT_CONNECTION with RESET_SESSION, INIT_CONNECTION with RESET_CIPHER	Combination of requests specified is not permitted.	No
EPipe	JrUnexpectedErr	INIT_CONNECTION, RESET_CIPHER	TCP connection is no longer in Established state. Two-way communication is not possible.	Yes
EProto	JrGetConnErr	RESET_SESSION, RESET_CIPHER	An INIT_CONNECTION request has not been received for the connection.	Yes

Table 56. SIOCTTLSCTL error codes (continued)

Errno	**Errnojr**	**IOCTL request specified (1)**	**Condition causing Error**	**Valid Data? (2)**
EProto	JrInvalidVersion	RESET_CIPHER	Connection is secured using SSL version 2.	Yes
EProtoType	JrSocketTypeNotSupported	Any	Socket is not a TCP socket.	No
EWouldBlock	JrOK	Any	Initial handshake is in progress and socket is a non-blocking socket. (3)	Yes

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[, request_type]
   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.

Chapter 29. Diagnosing IP security problems

This chapter describes how to diagnose IP security problems and contains the following sections:

Overview of diagnosing IP security 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 z/OS Network Security Configuration Assistant, which provides wizards and a set of reusable objects (at a different level of abstraction than if configured manually). The z/OS Network Security Configuration Assistant 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 57 provides a brief mapping of these objects.

Table 57. GUI-level object mapping

Policy Agent Object	**z/OS Network Security Configuration Assistant Object**
IpDataOffer	Configured in security levels implementing dynamic tunnels
IpDynVpnAction	Security level implementing dynamic tunnels A numeric suffix is appended to the Security Level name to guarantee uniqueness.
IpFilterRule	Connectivity rule A numeric suffix is appended to the connectivity rule name to guarantee uniqueness.
IpManVpnAction	Security level implementing manual tunnels A numeric suffix is appended to the security level name to guarantee uniqueness.
IpService	Configured in traffic descriptors A numeric suffix is appended to the traffic descriptor name to guarantee uniqueness.
IpTimeCondition	Defined within either Connectivity Rules or Security Levels implementing Manual Tunnels

Table 57. GUI-level object mapping (continued)

Policy Agent Object	z/OS Network Security Configuration Assistant Object
KeyExchangeAction	Configured in connectivity rules A numeric suffix is appended to the connectivity rule name to guarantee uniqueness.
KeyExchangeRule	Configured in Connectivity Rules A numeric suffix is appended to the Connectivity Rule name to guarantee uniqueness.
LocalDynVpnRule	Configured in connectivity rules Names are user specified.

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.

Problems can occur in the following areas:
- IP security policy installation
- IP security output to syslogd
- IP security operation

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 25, "Diagnosing Policy Agent problems," on page 601 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.

2. Issue **ipsec -f** display to see how the stack mapped your IpFilterPolicy statement. Refer to *z/OS Communications Server: IP System Administrator's Commands* for more information about the **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 *z/OS Communications Server: IP Configuration Guide*.

Perform the following steps to determine the cause for missing IP security syslogd output.

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. See "Diagnosing TRMD problems" on page 635 for more information.

   ---

3. Ensure that syslogd is running on this system.

   ---

4. Ensure that syslogd is configured for IP security output. TRMD always writes IP security log records to the syslog local4 facility and uses the info priority level.

   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.

   Tips:

   - 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 IDS output by stack.

   Guidelines:

   - Ensure that syslogd is configured to write TRMD and IKED messages. For example, the following two lines could be added to the syslogd configuration file to organize TRMD and IKED messages:

     ```
     *.*.local4.* /tmp/logs/iked.log
     *.trmd*.*.* /tmp/logs/trmd.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.

Steps for verifying IP security operation

Figure 93 on page 652 shows the decisions involved for IP security operation.

Verify IP Security Operation

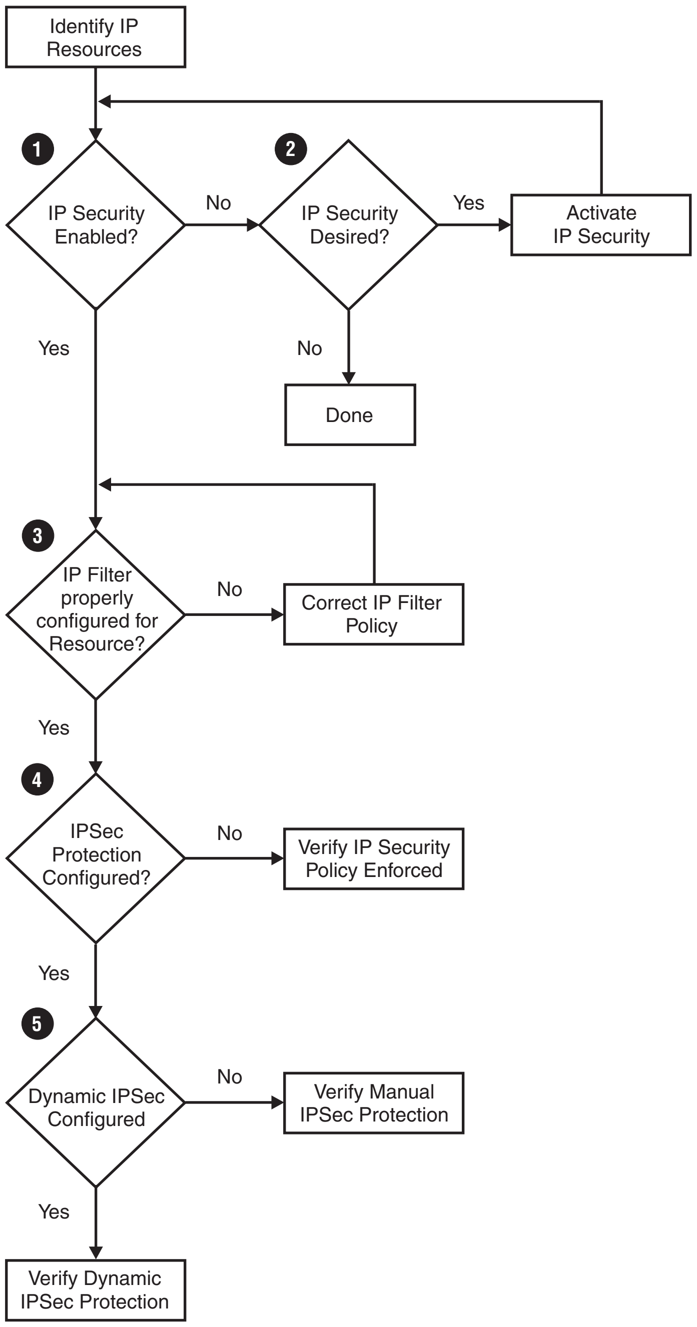

Identify IP Resources
1
IP Security Enabled?
No
2
IP Security Desired?
Yes
Activate IP Security
Yes
No
Done
3
IP Filter properly configured for Resource?
No
Correct IP Filter Policy
Yes
4
IPSec Protection Configured?
No
Verify IP Security Policy Enforced
Yes
5
Dynamic IPSec Configured
No
Verify Manual IPSec Protection
Yes
Verify Dynamic IPSec Protection

Figure 93. Overview of verifying 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 statement. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about the IpFilterRule and IPSECRULE statements.

Perform the following steps to verify IP security operation.

1. Use the Netstat CONFIG/-f command to determine whether the TCP/IP stack is configured for IPSECURITY or FIREWALL. 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 configured for FIREWALL, refer to *z/OS Integrated Security Services Firewall Technologies*.
   - If the stack is not configured for IPSECURITY, proceed to step 2.
   - If the stack is configured for IPSECURITY, proceed to step 3.

2. If IP security is desired but not enabled, configure the stack for IP security using the IPCONFIG IPSECURITY statement in the TCP/IP profile. Refer to *z/OS Communications Server: IP Configuration Reference* for more information about the IPCONFIG IPSECURITY statement. Refer to *z/OS Communications Server: IP Configuration Guide* for general information about IP security concepts, including IP filtering.

3. If IP security is enabled, 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 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.

   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 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 *z/OS Communications Server: IP Configuration Guide* for general information about configuring IP filters.

4. Locate the Type field in the **ipsec -t** command output to determine whether IP security protection is configured for the identified IP packet. If the Type field indicates `Generic`, then IPSec protection is not configured for the identified IP packet. See "Steps for verifying IP security policy enforcement" on page 658 to verify that the configured policy is enforced for the IP traffic characterized by the identified IP packet.

5. 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 655.

Steps for verifying manual IPSec protection

Figure 94 on page 654 shows the decisions involved for verifying manual IPSec protection.

Verify Manual IPSec Protection

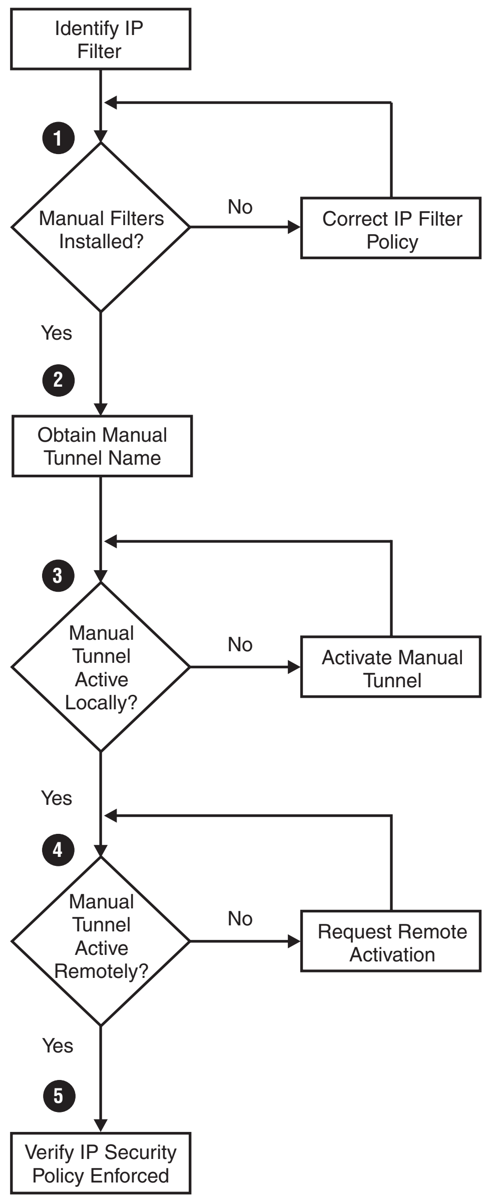

Identify IP Filter
1
Manual Filters Installed?
No
Correct IP Filter Policy
Yes
2
Obtain Manual Tunnel Name
3
Manual Tunnel Active Locally?
No
Activate Manual Tunnel
Yes
4
Manual Tunnel Active Remotely?
No
Request Remote Activation
Yes
5
Verify IP Security Policy Enforced

Figure 94. Overview of verifying manual IPSec protection

Before you begin: Complete the steps in "Steps for verifying IP security operation" on page 651 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*

Communications Server: IP Configuration Reference for more information about the IpManVpnAction and IpTimeCondition statements.

If IP filter rules are not installed, also verify that Policy Agent is active.

---

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 z/OS Network Security Configuration Assistant 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 enforcement" on page 658.

Steps for verifying dynamic IPSec protection

Figure 95 on page 656 shows the decisions involved for verifying dynamic IPSec protection.

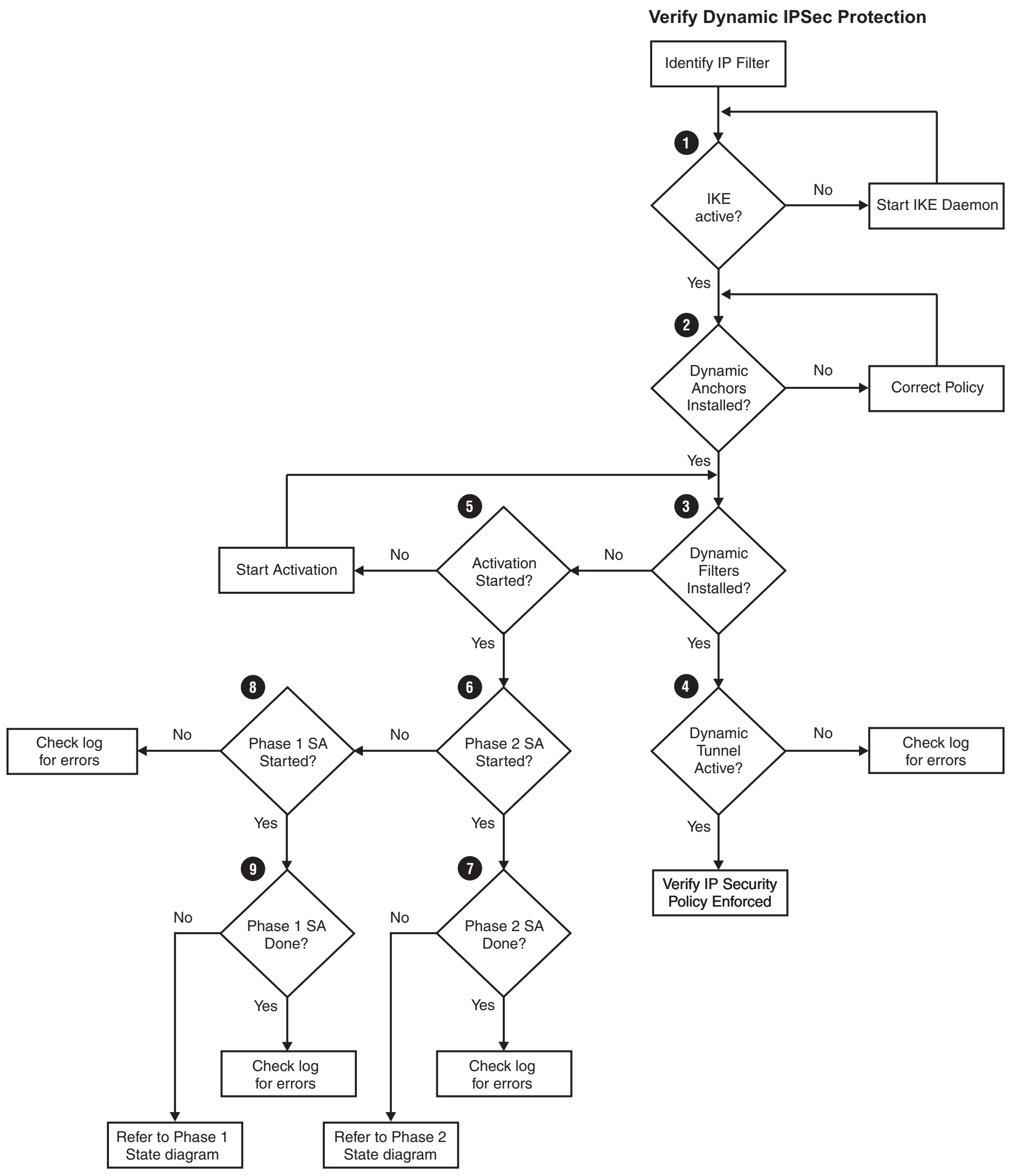


Figure 95. Overview of verifying dynamic IPSec protection

Before you begin: Complete the steps in "Steps for verifying IP security operation" on page 651 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 29.

   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

an IpTimeCondition. For information about the **ipsec** command, refer to *z/OS Communications Server: IP System Administrator's Commands*. Refer to *z/OS Communications Server: IP Configuration Reference* 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 syslogd output for errors. Otherwise, see "Steps for verifying IP security policy enforcement" on page 658.

---

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 **Yes** 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 656, 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 817. 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 812. Otherwise, check the syslogd output for errors.

Steps for verifying IP security policy enforcement

Figure 96 on page 659 shows the decisions involved for verifying IP security policy enforcement.

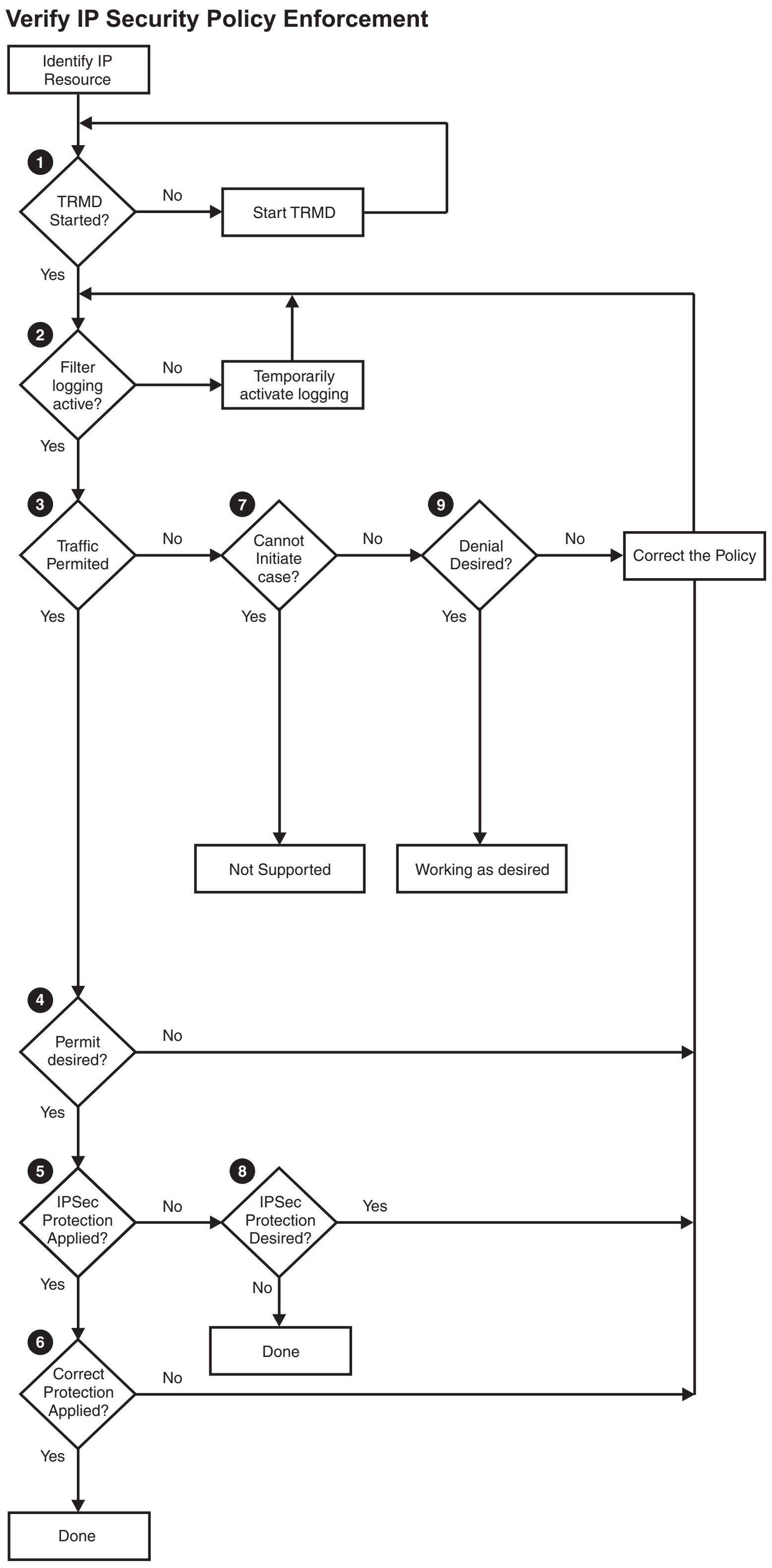


Figure 96. Overview of verifying IP security policy enforcement

Before you begin: Complete the steps in "Steps for verifying IP security operation" on page 651 in order to identify the name of an IpFilterRule or IPSECRULE for which IP security policy enforcement is to be verified.

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. Use the instructions in the following lists to temporarily activate logging for the filter if it is not already active. Refer to *z/OS Communications Server: IP Configuration Reference* for information about the IpFilterPolicy, IpFilterRule, IpGenericFilterAction, IPSEC and IPSECRULE statements.
   - If the **ipsec** command header output indicates Stack Policy, do the following:
     - If the **ipsec** command header output indicates `Logging NO`, temporarily specify FilterLogging Yes on the IpFilterPolicy statement. If you are using the z/OS Network Security Configuration Assistant to configure, select **Enable filter logging** on the IPSec: Stack Level Settings panel.
     - If the displayed filter Logging field is not `ALL`, specify IpFilterLogging **Yes** on the IpGenericFilterAction referenced by the IpFilterRule. If you are using the z/OS Network Security Configuration Assistant 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 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, or EZD0822I 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 z/OS Network Security Configuration Assistant 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 z/OS Network Security Configuration Assistant 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. Because data traffic cannot be initiated to the remote data endpoint under certain conditions, check to determine whether all of the following conditions are true:
   - NAT Traversal support is being utilized
   - A NAT was detected between the local security endpoint and the remote security endpoint.
   - The remote security endpoint is acting as a security gateway.

   If an attempt is made to initiate data traffic when these conditions are true, message EZD0832I is issued.

8. If the IP traffic is not being protected with IPSec, but IPSec protection is desired, 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, correct the filter configuration to change this situation. Refer to *z/OS Communications Server: IP Configuration Guide* for information about configuring IP filtering.

Tools for diagnosing IP security problems

This section describes tools used to diagnose IP security 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

ipsec -f display

The **ipsec -f** display command displays information about the current set of filter rules in use by a stack.

You can use the options listed in Table 58 to define the display.

Table 58. ipsec -f display command options

Option	Use
-p	Directs the command to a stack other than the default stack.
-c profile	Displays information about the set of filter rules defined on the IPSEC statement in the TCP/IP profile.
-c policy	Display information about the set of filter rules defined in the Policy Agent IPSec Configuration file.

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.

Several different types of filter rules exist. By default, the **ipsec -f** display output includes information about generic, 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. NRF (NAT resolution filter) rules can be present when at least one NAT device exists between the local security endpoint and a remote security endpoint that might be acting as a gateway. Refer to *z/OS Communications Server: IP Configuration Guide* for an explanation filter types.

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. 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. 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. 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. The SECCLASS option of LINK statement is used to assign a security classification to an interface. The LINK 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.

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. Port translation is performed as needed for TCP and UDP connections that utilize a dynamic security association with a remote security gateway that resides behind a NAT.

Using the pasearch command

You can use the **pasearch** commands listed in Table 59 on page 664 to display information about the IPSec policy loaded by the Policy Agent for the stack:

Table 59. pasearch commands

Command	Description
pasearch -v a	Displays all IPSec policy
pasearch -v f	Displays IpFilterPolicy
pasearch -v k	Displays KeyExchangePolicy
pasearch -v l	Displays LocalDynVpnPolicy

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 304 for more details and sample IKE daemon syslog output.

If you are using the z/OS Network Security Configuration Assistant to configure, the IKE Syslog level and the Policy Agent API Syslog level can be set from the IPSec: IKE Daemon Settings panel.

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 enforcement" on page 658 for details about enabling IP filter logging.

Chapter 30. Diagnosing OMPROUTE problems

This chapter provides information and guidance to diagnose OMPROUTE problems, and contains the following sections:

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 Hierarchical File System (HFS) 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 HFS 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 684 and Chapter 5, "TCP/IP services traces and IPCS support," on page 41.
- 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 HFS. 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.

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 chapter 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

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. Generally, an inability to connect to TCP/IP is caused by an error in the configuration or definitions in TCP/IP. An inability to connect to one of the required ports is generally caused by an error in the configuration or definitions in TCP/IP or by attempting to start OMPROUTE when OMPROUTE is already connected to the specified stack.

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 556.

Routing failures

Routing problems are usually the result of outages in a network and a lack of 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 60.

Table 60. OMPROUTE command terms

Term	Description
PING	Refers to z/OS UNIX ping, TSO PING, and the ping commands used on other platforms.

Table 60. OMPROUTE command terms (continued)

Term	Description
Traceroute	Refers to z/OS UNIX **otracert|traceroute**, TSO TRACERTE, and the **traceroute** commands used on other platforms.
NETSTAT ROUTE	Refers to the z/OS UNIX **netstat -r**, TSO NETSTAT ROUTE, and the **netstat route** commands used on other platforms.
OMPROUTE RTTABLE (IPv4)	Refers to D TCPIP,tcpipjobname,OMPROUTE,RTTABLE (for IPv4).
OMPROUTE RT6TABLE (IPv6)	Refers to D TCPIP,tcpipjobname,OMPROUTE,RT6TABLE (for IPv6).
NETSTAT GATE (IPv4 only)	Refers to the z/OS UNIX **netstat -g** and TSO NETSTAT GATE commands. This command is available only on the z/OS platform.
NETSTAT ARP	Refers to the z/OS UNIX netstat -R ALL, TSO NETSTAT ALL, and the **netstat arp** commands used on other platforms.

If a PING or Traceroute command fails on a system where OMPROUTE is being used, a client is unable to get a positive response to a PING or TRACERT command. Before doing anything else, do the following:

1. If using a name for the destination host, try using an IP address. If the IP address works, then the problem is related to DNS configuration. See Chapter 37, "Diagnosing resolver problems," on page 761 for problem determination.
2. Issue the NETSTAT ROUTE and OMPROUTE RTTABLE or RT6TABLE commands on the local and remote hosts to get the routing table information for both the TCP/IP stack and OMPROUTE.
3. Issue the TRACERT command to the destination host. If the destination host is located on a local multi-access network, issue the NETSTAT ARP command to determine if there is an ARP entry for the remote host. If there is a TRANSLATE statement defined in the TCP/IP profile for this host, make sure that its MAC address is correct.

From the NETSTAT ROUTE outputs, determine the route that is used to reach the destination and determine the route active state. For IPv4, a routing table is searched in the following order, starting with the most specific to the least specific:

1. Host Routes
2. Subnet Routes
3. Network Routes
4. Supernet Routes
5. Default Routes

For IPv6, a routing table is searched in the following order, starting with the most specific to the least specific:

1. Host Routes
2. Prefix Routes
3. Default Routes

If the local or remote is a z/OS host, the NETSTAT GATE command can be issued to display the MTUs and subnet masks for the IPv4 routes in the routing table.

If there are no active routes available to reach the destination or if there are improperly configured channel-attached or network-attached routers along the routing path, the PING and Traceroute commands fail. To function correctly, PING requires active routes in both directions between the Ping origin and the Ping destination. If the routes are shown to be active at the local and remote hosts, the problem is most likely to be caused by a router along the routing path. Use the output from the Traceroute command to locate the suspect router.

Documenting routing failures

The following documentation should be available for initial diagnosis of routing failures:

- MVS system log.
- Output from onetstat -r or TSO NETSTAT ROUTE command.
- 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 671.
- TCP/IP and OMPROUTE CTRACE. For information about generating an OMPROUTE Component Trace, see "TCP/IP services component trace for OMPROUTE" on page 684.
- Output from appropriate OMPROUTE DISPLAY commands as described in the *z/OS Communications Server: IP System Administrator's Commands*.
- Output from NETSTAT ROUTE and OMPROUTE RTTABLE or RT6TABLE commands for both local and remote hosts.
- Output from NETSTAT GATE command if the local or remote is a z/OS host running IPv4.

  Restriction: NETSTAT GATE/-g is not enhanced to support IPv6 routes.
- Outputs from PING and Traceroute commands.
- Outputs from NETSTAT ARP commands.

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.
- If the output from the onetstat -r or TSO NETSTAT ROUTE command does not show the expected results relative to the desired destination, do one or more of the following:
  - 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. So if the netstat done at the origin shows correct routing, you must also do netstat route displays at the destination to verify that it can send replies back to the origin.

    Also, this is affected by SOURCEVIPA. If SOURCEVIPA is enabled at the origin of the communication, then the destination must be able to route back to the VIPA address that the origin uses as its source address. 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.

– Make sure that the router or routers involved in providing information relative to this destination are operational and participating in the correct routing protocol.
– Make sure that the physical connections involved in reaching the destination are active.
– Use the OMPROUTE DISPLAY commands described in the *z/OS Communications Server: IP System Administrator's Commands* to determine if anything in the configuration or current state of OMPROUTE has resulted in the absence of a route to the destination.

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 684). 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 671.

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 HFS 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 HFS 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 97 on page 675 is a sample OMPROUTE initialization and IPv4 routing protocol trace with descriptions for some of the trace entries:

```
1    EZZ7800I OMPROUTE starting
         EZZ7845I Established affinity with TCPCS8
    EZZ7817I Using defined OSPF protocol 89
    EZZ7838I Using configuration file: /u/user146/omproute/omproute.conf
2   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
2.5  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
3   EZZ8057I Added network 9.67.100.0 to interface 9.67.100.8
                on net 0 interface CTC1
    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
    EZZ8057I Added network 9.67.100.7 to interface 9.67.100.8 on net 0
          interface CTC1
    EZZ7827I Adding stack route to 9.67.100.7, mask 255.255.255.255 via
          0.0.0.0, link CTC1, metric 1, type 129
4   EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
                 interface CTC1
    EZZ7879I Joining multicast group 224.0.0.5 on interface 9.67.100.8
5   EZZ7913I State change, interface 9.67.100.8, new state 16,
                 event 1

:
    EZZ7875I No IPv4 Default Route Installed
    EZZ81001 OMPROUTE subagent Starting
    EZZ7898I OMPROUTE Initialization Complete
    EZZ81011 OMPROUTE subagent Initialization Completed
    EZZ7908I Received packet type 1 from 9.167.100.13
6   EZZ8011I send request to address 9.67.100.7
    EZZ8015I sending packet to 9.67.100.7
    EZZ8011I send request to address 9.169.100.14
    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
7   EZZ7908I Received packet type 1 from 9.67.100.7
    EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
          interface CTC1
8   EZZ7919I State change, neighbor 9.67.100.7, new state 4, event 1
9   EZZ7919I 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
10  EZZ7919I State change, neighbor 9.67.100.7, new state 16,
                event 14
11  EZZ7910I Sending multicast, type 2, destination 224.0.0.5 net
                0 interface CTC1
12  EZZ7908I Received packet type 2 from 9.67.100.7
13  EZZ7919I State change, IPv4neighbor 9.67.100.7, new state 32, event 5
14  EZZ7910I Sending multicast, type 3, destination 224.0.0.5 net 0
                interface CTC1
    EZZ7908I Received packet type 2 from 9.67.100.7
15  EZZ7908I Received packet type 4 from 9.67.100.7
```

Figure 97. 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
17 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
18 EZZ7919I State change, neighbor 9.67.100.7, new state 128,
           event 6
19 EZZ7908I Received packet type 5 from 9.67.100.7
20 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 97. Sample OMPROUTE Trace Output (Part 2 of 6)

```
21 EZZ7949I Dijkstra calculation performed, on 2 area(s)
   EZZ7935I New OMPROUTE route to destination Net 9.67.100.7,
          type SPF cost 1
   EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.7 org
          9.67.100.8
   EZZ7908I Received packet type 5 from 9.67.100.7
   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
   EZZ7908I Received packet type 4 from 9.67.100.7
   EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id
          9.167.100.13 org 9.167.100.13
   EZZ7928I from 9.67.100.7, new LS advertisement: typ 4 id
          9.67.100.8 org 9.167.100.13
   EZZ7928I from 9.67.100.7, new LS advertisement: typ 3 id
          9.67.100.7 org 9.167.100.13
   EZZ7908I Received packet type 5 from 9.67.100.7
   EZZ7910I Sending multicast, type 5, destination 224.0.0.5 net 0
          interface CTC1
   EZZ7949I Dijkstra calculation performed, on 2 area(s)
22 EZZ7827I Adding stack route to 9.167.100.13, mask 255.255.
                255.255 via 9.67.100.7, link CTC1, metric 2, type 129
   EZZ7935I New OMPROUTE route to destination Net 9.167.100.13,
          type SPF cost 2
   EZZ7935I New OMPROUTE route to destination Net 9.67.100.8,
          type SPF cost 2
   EZZ7913I State change, interface 9.67.100.8, new state 16, event 1
   EZZ7935I New OMPROUTE route to destination BR 9.167.100.13,
          type SPF cost 2
   EZZ7827I Adding stack route to 9.167.100.17, mask 255.255.255.255
          via 9.67.100.7, link CTC1, metric 3, type 129
   EZZ7935I New OMPROUTE route to destination Net 9.167.100.17,
          type SPF cost 3
   EZZ7934I Originating LS advertisement: typ 3 id 9.167.100.13 org
          9.67.100.8
   EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.8 org
          9.67.100.8
   EZZ7934I Originating LS advertisement: typ 3 id 9.167.100.17 org
          9.67.100.8
23 EZZ7909I Sending unicast type 1 dst 9.167.100.13
   EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
          interface CTC1
   EZZ7908I Received packet type 1 from 9.167.100.13
   EZZ7919I State change, neighbor 9.167.100.13, new state 4, event 1
   EZZ7919I State change, neighbor 9.167.100.13, new state 8, event 3
   EZZ7919I State change, neighbor 9.167.100.13, new state 16, event 14
   EZZ7909I Sending unicast type 2 dst 9.167.100.13
   EZZ7908I Received packet type 4 from 9.67.100.7
   EZZ7928I from 9.67.100.7, new LS advertisement: typ 4 id
          9.67.100.8 org 9.167.100.13
   EZZ7928I from 9.67.100.7, new LS advertisement: typ 3 id
          9.67.100.7 org 9.167.100.13
   EZZ7908I Received packet type 2 from 9.167.100.13
   EZZ7919I State change, neighbor 9.167.100.13, new state 32, event 5
   EZZ7909I Sending unicast type 2 dst 9.167.100.13
   EZZ7910I Sending multicast, type 5, destination 224.0.0.5 net 0
          interface CTC1
   EZZ7908I Received packet type 2 from 9.167.100.13
   EZZ7909I Sending unicast type 3 dst 9.167.100.13
   EZZ7908I Received packet type 4 from 9.167.100.13
```

Figure 97. Sample OMPROUTE Trace Output (Part 3 of 6)

```
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 1
       interface CTC2
EZZ7928I from 9.167.100.13, new LS advertisement: typ 1 id
       9.67.100.8 org 9.67.100.8
EZZ7927I from 9.167.100.13, self update: typ 1 id 9.67.100.8 org
       9.67.100.8
```

⋮

```
 EZZ7909I Sending unicast type 4 dst 9.167.100.13
EZZ7919I State change, neighbor 9.167.100.13, new state 128, event 6
EZZ7909I Sending unicast type 2 dst 9.167.100.13
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
EZZ7933I Flushing advertisement: typ 3 id 9.67.100.7 org 9.167.100.13
EZZ7933I Flushing advertisement: typ 4 id 9.67.100.8 org 9.167.100.13
EZZ7909I Sending unicast type 5 dst 9.167.100.13
EZZ8015I sending packet to 9.67.100.7
EZZ8012I sending broadcast response to address 9.67.100.255 in
       1 packets with 1 routes
EZZ7908I Received packet type 5 from 9.67.100.7
EZZ7908I Received packet type 1 from 9.67.100.7
EZZ8004I response received from host 9.67.100.7
EZZ7908I Received packet type 5 from 9.167.100.13
EZZ7949I Dijkstra calculation performed, on 2 area(s)
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 9.167.100.13
EZZ7928I from 9.167.100.13, new LS advertisement: typ 1 id
       9.167.100.13 org 9.167.100.13
EZZ7908I Received packet type 4 from 9.67.100.7
EZZ7928I from 9.67.100.7, new LS advertisement: typ 1 id
       9.167.100.13 org 9.167.100.13
EZZ7910I Sending multicast, type 5, destination 224.0.0.5 net 0
       interface CTC1
EZZ7909I Sending unicast type 5 dst 9.167.100.13
EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8 org
       9.67.100.8
EZZ7909I Sending unicast type 4 dst 9.167.100.13
EZZ7908I Received packet type 5 from 9.167.100.13
EZZ8062I Subnet 9.0.0.0 defined
EZZ7949I Dijkstra calculation performed, on 2 area(s)
EZZ7935I New OMPROUTE route to destination BR 9.167.100.13,
       type SPF cost 2
```
24
```
EZZ7895I Processing DISPLAY command - OSPF,LIST,INTERFACES
EZZ7809I    EZZ7833I INTERFACE CONFIGURATION
EZZ7809I IP ADDRESS      AREA    COST  RTRNS  TRNSDLY PRI  HELLO   DEAD
EZZ7809I 9.169.100.18    0.0.0.0    1     10      1     1     20    80
EZZ7809I 9.67.100.8      2.2.2.2    1     10      1     1     20    80
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
       interface CTC1
EZZ7908I Received packet type 1 from 9.167.100.13
EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 1
       interface CTC2
EZZ7909I Sending unicast type 1 dst 9.167.100.13
EZZ7908I Received packet type 1 from 9.67.100.7
EZZ8015I sending packet to 9.67.100.7
```

Figure 97. Sample OMPROUTE Trace Output (Part 4 of 6)

```
     EZZ8012I sending broadcast response to address 9.67.100.255 in
            1 packets with 1 routes
     EZZ8004I response received from host 9.67.100.7
     EZZ8015I sending packet to 9.169.100.14
     EZZ8012I sending broadcast response to address 9.169.100.255 in
            1 packets with 1 routes
25   EZZ7895I Processing MODIFY command - TRACE=2
25.5
     EZZ7895I Processing MODIFY command -TRACE6=2
     EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
            interface CTC1
26   EZZ7876I -- OSPF Packet Sent ------ Type: Hello
     EZZ7878I OSPF Version: 2            Packet Length: 48
     EZZ7878I Router ID: 9.67.100.8      Area: 2.2.2.2
     EZZ7878I Checksum: 1dcf             Authentication Type: 0
     EZZ7878I Hello_Interval: 20         Network mask: 255.255.255.0
     EZZ7878I Options: E
     EZZ7878I Router_Priority: 1         Dead_Router_Interval: 80
     EZZ7878I Backup DR: 0.0.0.0         Designated Router: 0.0.0.0
     EZZ7878I Neighbor: 9.67.100.7
     EZZ7877I -- OSPF Packet Received -- Type: Hello
     EZZ7878I OSPF Version: 2            Packet Length: 48
     EZZ7878I Router ID: 9.67.100.7      Area: 2.2.2.2
     EZZ7878I Checksum: 1dcf             Authentication Type: 0
     EZZ7878I Hello_Interval: 20         Network mask: 255.255.255.0
     EZZ7878I Options: E
     EZZ7878I Router_Priority: 1         Dead_Router_Interval: 80
     EZZ7878I Backup DR: 0.0.0.0         Designated Router: 0.0.0.0
     EZZ7878I Neighbor: 9.67.100.8
     EZZ7908I Received packet type 1 from 9.67.100.7
27              -- RIP Packet Received -- Type: Response (V1)
            Destination_Addr: 9.169.100.0     metric: 2
     EZZ8004I response received from host 9.67.100.7
            -- RIP Packet Sent ------ Type: Response (V1)
            Destination_Addr: 9.169.100.0     metric: 1
     EZZ8015I sending packet to 9.67.100.7
     EZZ8012I sending broadcast response to address 9.67.100.255 in
            1 packets with 1 routes
28   EZZ7895I Processing MODIFY command - TRACE=1
     EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 1
            interface CTC2
     EZZ7909I Sending unicast type 1 dst 9.167.100.13
     EZZ7908I Received packet type 1 from 9.67.100.7
     EZZ8004I response received from host 9.67.100.7
     EZZ8015I sending packet to 9.67.100.7
     EZZ8004I response received from host 9.67.100.7
     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
     EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 0
            interface CTC1

⋮
     EZZ7909I Sending unicast type 1 dst 9.167.100.13
     EZZ7908I Received packet type 1 from 9.67.100.7
29   EZZ7862I Received update interface CTC1
```

Figure 97. Sample OMPROUTE Trace Output (Part 5 of 6)

```
30 EZZ8061I Deleted net 9.67.100.0 route via 9.67.100.8 net 0
            interface CTC1
   EZZ7864I Deleting all stack routes to 9.67.100.0, mask 255.255.255.0
31 EZZ7919I State change, neighbor 9.67.100.7, new state 1, event 11
   EZZ7879I Leaving multicast group 224.0.0.5 on interface 9.67.100.8
32 EZZ7913I State change, interface 9.67.100.8, new state 1, event 7
   EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8 org
       9.67.100.8
   EZZ7934I Originating LS advertisement: typ 1 id 9.67.100.8 org
       9.67.100.8
   EZZ7909I Sending unicast type 4 dst 9.167.100.13
   EZZ8015I sending packet to 9.169.100.14
   EZZ8012I sending broadcast response to address 9.169.100.255 in
       1 packets with 1 routes
   EZZ7934I Originating LS advertisement: typ 5 id 9.67.100.0 org
       9.67.100.8
   EZZ7933I Flushing advertisement: typ 5 id 9.67.100.0 org 9.67.100.8
   EZZ7949I Dijkstra calculation performed, on 1 area(s)
   EZZ7801I Deleting stack route to 9.67.100.7, mask 255.255.255.255
       via 0.0.0.0, link CTC1, metric 1, type 129
   EZZ7935I New OMPROUTE route to destination Net 9.67.100.7,
       type SPIA cost 5
   EZZ7943I Destination Net 9.167.100.13 now unreachable
   EZZ7864I Deleting all stack routes to 9.167.100.13, mask
       255.255.255.255
   EZZ7935I New OMPROUTE route to destination Net 9.67.100.8,
       type SPIA cost 4
   EZZ7919I State change, neighbor 9.167.100.13, new state 1, event 11
   EZZ7913I State change, interface 9.67.100.8, new state 1, event 7
   EZZ7943I Destination BR 9.167.100.13 now unreachable
   EZZ7943I Destination Net 9.167.100.17 now unreachable
   EZZ7864I Deleting all stack routes to 9.167.100.17, mask
       255.255.255.255
   EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.7
       org 9.67.100.8
   EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.8 org
       9.67.100.8

 ⋮
   EZZ7933I Flushing advertisement: typ 3 id 9.167.100.17 org 9.67.100.8
   EZZ7933I Flushing advertisement: typ 3 id 9.67.100.8 org 9.67.100.8
   EZZ7933I Flushing advertisement: typ 3 id 9.167.100.13 org 9.67.100.8
   EZZ7933I Flushing advertisement: typ 3 id 9.67.100.7 org 9.67.100.8
   EZZ8015I sending packet to 9.169.100.14
   EZZ8012I sending broadcast response to address 9.169.100.255 in 1
       packets with 1 routes
   EZZ7949I Dijkstra calculation performed, on 1 area(s)
   EZZ7943I Destination Net 9.67.100.7 now unreachable
   EZZ7943I Destination Net 9.67.100.8 now unreachable
   EZZ7943I Destination BR 9.167.100.13 now unreachable
   EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.7 org
       9.67.100.8
   EZZ7934I Originating LS advertisement: typ 3 id 9.67.100.8 org
       9.67.100.8
   EZZ7933I Flushing advertisement: typ 3 id 9.67.100.8 org 9.67.100.8
   EZZ7933I Flushing advertisement: typ 3 id 9.67.100.7 org 9.67.100.8
   EZZ7910I Sending multicast, type 1, destination 224.0.0.5 net 1
       interface CTC2
   EZZ7804I OMPROUTE exiting
```

Figure 97. Sample OMPROUTE Trace Output (Part 6 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.
2.5	IPv6 tracing is in the file pointed to by the OMPROUTE_IPV6_DEBUG_FILE environment variable
3	Direct routes are added for each TCP/IP stack IPv4 interface.
4	OSPF Hello packet sent out OSPF interface.
5	OSPF Interface transitions to state "point-to-point."
6	RIP Requests & Responses begin being sent out RIP interface.
7	OSPF Hello packet received from OSPF neighbor.
8	OSPF neighbor transitions to state "Init."
9	OSPF neighbor transitions to state "2-Way."
10	OSPF neighbor transitions to state "ExStart."
11	OSPF Database Description packet sent out OSPF interface.
12	OSPF Database Description received from OSPF neighbor.
13	OSPF neighbor transitions to state "Exchange."
14	OSPF Link State Request packet sent out OSPF interface.
15	OSPF Link State Update packet received from OSPF neighbor.
16	Link State Advertisements from received Update packet are processed.
17	OSPF Link State Update packet sent out OSPF interface.
18	OSPF neighbor transitions to state "Full."
19	OSPF Link State Acknowledgment packet received from OSPF neighbor.
20	OSPF Link State Acknowledgment packet sent out OSPF interface.
21	OSPF Dijkstra calculation is performed.
22	Learned route is added to TCP/IP stack IPv4 route table.
23	Adjacency establishment begins with router at other end of OSPF Virtual Link.
24	Request received to display OSPF Interface configuration information.
25	Request received to change IPv4 tracing level to 2 (adds formatted packets).
25.5	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).
26	Formatted OSPF packet.
27	Formatted RIP packet.
28	Request received to change tracing level back to 1(-t1).
29	OMPROUTE learns of stopped TCP/IP IPv4 interface.
30	Routes over stopped interface are deleted.
31	Neighbor over stopped interface transitions to state "Down."
32	Stopped interface transitions to state "Down."

The following sample shows OMPROUTE IPv6 routing protocol trace with descriptions for some of the trace entries:

```
1  EZZ7977I Processing IPv6 interface from stack, address 1977::7,
name MPCPTPV67, index 16, flags 811, flags2 0
EZZ7977I Processing IPv6 interface from stack, address
fe80::542c:ed1e:1362:4d26, name MPCPTPV67, index 16, flags 811,
flags2 2
EZZ7977I Processing IPv6 interface from stack, address
7:7:7:7:7:7:7:7, name VIPA16, index 18, flags 4001, flags2 0
2  EZZ8057I Added network 1977::7 to interface
fe80::542c:ed1e:1362:4d26 on net 16 interface MPCPTPV67
EZZ7879I Joining multicast group ff02::9 on interface MPCPTPV67
EZZ8057I Added network 7:7:7:7:7:7:7:7 to interface
7:7:7:7:7:7:7:7 on net 18 interface VIPA16
EZZ8057I Added network 7:7:7:: to interface 7:7:7:7:7:7:7:7 on
net 18 interface VIPA16
EZZ7827I Adding stack route to ::, prefixlen 0 via
fe80::7cb6:c5d5:6593:c076, link MPCPTPV67, metric 0, type 136
3  EZZ8011I send request to address ff02::9
EZZ8015I sending packet to ff02::9
EZZ8021I sending IPv6RIP response to address ff02::9 from
fe80::542c:ed1e:1362:4d26 in 1 packets with 6 routes
4  EZZ8004I response received from host fe80::846e:70a6:8ca6:48b7
EZZ7827I Adding stack route to ::, prefixlen 0 via
fe80::846e:70a6:8ca6:48b7, link MPCPTPV67, metric 0, type 136
EZZ7801I Deleting stack route to ::, prefixlen 0 via
fe80::7cb6:c5d5:6593:c076, link MPCPTPV67, metric 0, type 136
EZZ8010I update route to net :: at metric 9 hops via router
fe80::846e:70a6:8ca6:48b7
EZZ7806I Changing stack route to ::, prefixlen 0 via
fe80::846e:70a6:8ca6:48b7, link MPCPTPV67, metric 9, type 136
EZZ8010I update route to net 1967::6 at metric 2 hops via router
fe80::846e:70a6:8ca6:48b7
EZZ7827I Adding stack route to 1967::6, prefixlen 128 via
fe80::846e:70a6:8ca6:48b7, link MPCPTPV67, metric 2, type 1
EZZ8010I update route to net 6:6:6:6:6:6:6:6 at metric 2 hops
via router fe80::846e:70a6:8ca6:48b7
EZZ7827I Adding stack route to 6:6:6:6:6:6:6:6, prefixlen 128
via fe80::846e:70a6:8ca6:48b7, link MPCPTPV67, metric 2
EZZ8010I update route to net 6:6:6:: at metric 2 hops via router
fe80::846e:70a6:8ca6:48b7
EZZ7827I Adding stack route to 6:6:6::, prefixlen 48 via
fe80::846e:70a6:8ca6:48b7, link MPCPTPV67, metric 2, type 12
EZZ8010I update route to net 1946::6 at metric 2 hops via router
fe80::846e:70a6:8ca6:48b7
EZZ7827I Adding stack route to 1946::6, prefixlen 128 via
fe80::846e:70a6:8ca6:48b7, link MPCPTPV67, metric 2, type 1
EZZ8010I update route to net 9::67:120:4 at metric 3 hops via
router fe80::846e:70a6:8ca6:48b7
EZZ7827I Adding stack route to 9::67:120:4, prefixlen 128 via
fe80::846e:70a6:8ca6:48b7, link MPCPTPV67, metric 3, type 1
EZZ8010I update route to net 1946::4 at metric 3 hops via router
fe80::846e:70a6:8ca6:48b7
EZZ7827I Adding stack route to 1946::4, prefixlen 128 via
fe80::846e:70a6:8ca6:48b7, link MPCPTPV67, metric 3, type 1
...
5  EZZ8015I sending packet to ff02::9
6   -- 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: 7:7:7:7:7:7:7:7
     Prefix Length: 128   metric: 1
   Destination_Addr: 7:7:7::
     Prefix Length: 48   metric: 1
   Destination_Addr: 9::67:120:7
     Prefix Length: 128   metric: 1
   Destination_Addr: 1967::
     Prefix Length: 16   metric: 1
   Destination_Addr: 1967::6
     Prefix Length: 128   metric: 16
   Destination_Addr: 6:6:6:6:6:6:6:6
     Prefix Length: 128   metric: 16
   Destination_Addr: 6:6:6::
     Prefix Length: 48   metric: 16
   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: 1946::
     Prefix Length: 16   metric: 2
   Destination_Addr: 1111::
     Prefix Length: 16   metric: 16
   Destination_Addr: 50c9:c2d4::
     Prefix Length: 64   metric: 3
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
   Destination_Addr: 6:6:6:6:6:6:6:6
     Prefix Length: 128   metric: 1
   Destination_Addr: 6:6:6::
     Prefix Length: 48   metric: 1
   Destination_Addr: 1946::6
     Prefix Length: 128   metric: 1
   Destination_Addr: 50c9:c2d4::
     Prefix Length: 64   metric: 16
   Destination_Addr: 1111::
     Prefix Length: 16   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: 128   metric: 16
   Destination_Addr: 7:7:7::
     Prefix Length: 48   metric: 16
   Destination_Addr: 9::67:120:7
     Prefix Length: 128   metric: 16
   Destination_Addr: 1976::
     Prefix Length: 16   metric: 2
   Destination_Addr: f000::
     Prefix Length: 4   metric: 2
   Destination_Addr: 9::67:120:4
     Prefix Length: 128   metric: 2
   Destination_Addr: 1946::4
     Prefix Length: 128   metric: 2
```

Figure 98. Sample IPv6 OMPROUTE Trace Output

Following are brief explanations of numbered items in the trace:

1 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.

2 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.

3 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.

4 IPv6 RIP Response received and associated routes added to IPv6 route table. Note that source address is always link-local.

5 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.

6 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 41. Also, see "Commands to enable, disable, and display the status of the OMPROUTE CTRACE" on page 688.

For detailed descriptions, refer to the following books:

- *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

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.

```
/*********************************************************************/
/*                                                                   */
/*  IBM Communications Server for z/OS                               */
/*  SMP/E Distribution Name: CTIORA00                                */
/*                                                                   */
/*  PART Name:  CTIORA00                                             */
/*                                                                   */
/*                                                                   */
/*  Copyright:                                                       */
/*              Licensed Materials - Property of IBM                 */
/*              5694-A01                                             */
/*              (C) Copyright IBM Corp. 1998,2003                    */
/*                                                                   */
/*                                                                   */
/*  Status:     CSV1R5                                               */
/*                                                                   */
/*                                                                   */
/*  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(1M)
 /*        WTR(wtr_procedure)                                        */
 /* ---------------------------------------------------------------- */
 /*   OPTIONS: NAMES OF FUNCTIONS TO BE TRACED, OR "ALL"             */
 /* ---------------------------------------------------------------- */
 /*        OPTIONS(            */
 /*                'ALL     '  */
```

Figure 99. SYS1.PARMLIB member CTIORA00 (Part 1 of 2)

```
/*                ,'MINIMUM '   */
/*                ,'ROUTE   '   */
/*                ,'PACKET  '   */
/*                ,'OPACKET '   */
/*                ,'RPACKET '   */
/*                ,'IPACKET '   */
/*                ,'SPACKET '   */
/*                ,'DEBUGTRC'   */
/*                 )            */
```

Figure 99. SYS1.PARMLIB member CTIORA00 (Part 2 of 2)

Table 61 describes the available trace options.

Table 61. OMPTRACE options

Trace event	Description
ALL	Select all types of records. Be aware that this option slows performance.
MINIMUM	Select OMPROUTE's minimum level of tracing. Specifying MINIMUM is the same as specifying ROUTE.
ROUTE	Select information exchange and routing updates between the OMPROUTE application and the z/OS TCP/IP Services stack.
PACKET	Select all inbound and outbound packet flows. This is the same as specifying OPACKET, RPACKET, and IPACKET.
RPACKET	Select inbound and outbound packet flows for the IPv4 RIP and IPv6 RIP protocols.
OPACKET	Select inbound and outbound packet flows for the IPv4 OSPF and IPv6 OSPF protocols.
IPACKET	Select inbound packets sent from z/OS TCP/IP with information regarding route or interface changes.
SPACKET	Trace inbound and outbound packets sent between the SNMP agent and the OMPROUTE subagent.
DEBUGTRC	Redirects IPv4 trace (-t), IPv4 debug (-d), IPv6 trace (-6t) and IPv6 debug (-6d) output to the CTRACE facility.

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 41.

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 41.) 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:

```
TRACE CT,ON,COMP=SYSTCPRT,SUB=(omproute_jobname)
R xx,OPTIONS=(ctrace options),END
```

- Issue the following console commands to enable a CTRACE to an external writer:

  ```
  TRACE CT,WTRSTART=writer_proc
  TRACE CT,ON,COMP=SYSTCPRT,SUB=(omproute_jobname)
  R xx,OPTIONS=(ctrace options),WTR=writer_proc,END
  ```

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=x
   ```

Step for displaying the CTRACE status

To display the CTRACE status, issue the following console command:

```
D TRACE,COMP=SYSTCPRT,SUB=(omproute_jobname)
```

Chapter 31. 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 chapter contains the following sections:

Figure 100 shows the NCPROUTE environment.

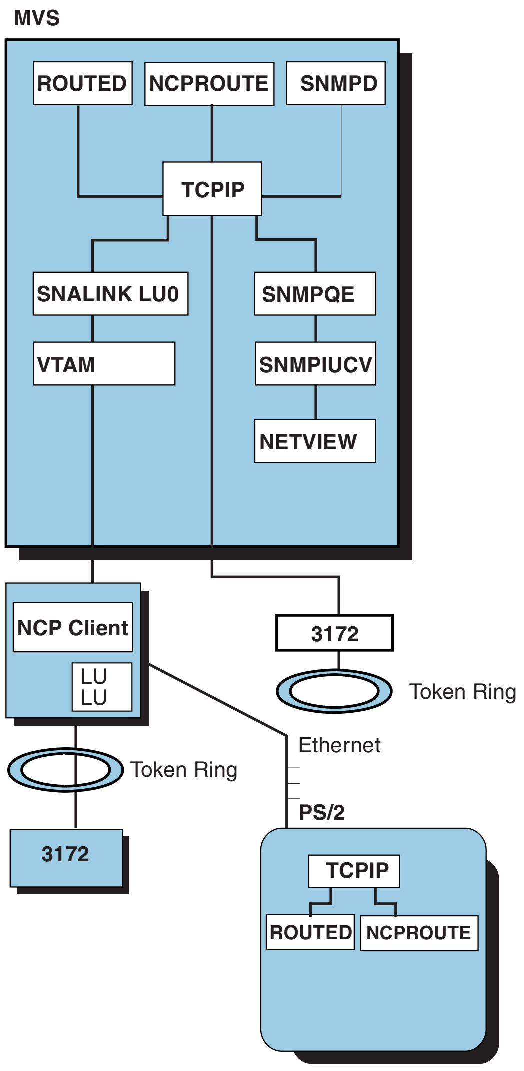


Figure 100. NCPROUTE environment

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 62 on page 693

Table 62. Types of RIP messages

Message type	Description
Request	There are two types of Request messages: **REQUEST TO** Sent from NCP by NCPROUTE over a network interface to request routing table from one or more neighboring RIP routers. **REQUEST FROM** 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.
Response	There are two types of Response messages: **RESPONSE TO** 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. **RESPONSE FROM** 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.
TraceOn	NCP received a request from a neighboring RIP router to enable the actions trace provided by NCPROUTE.
TraceOff	NCP received a request to a neighboring RIP router to disable tracing provided by NCPROUTE.

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 63 describes the MIB variables registered for use by NCPROUTE:

Table 63. MIB variables registered for use by NCPROUTE

MIB variable	Description
ipRouteDest	Destination IP address of this route
ipRouteMetric1	Primary routing metric for this route
ipRouteMetric2	Alternative routing metric for this route
ipRouteMetric3	Another alternative routing metric for this route
ipRouteMetric4	Another alternative routing metric for this route
ipRouteNextHop	IP address of the next hop of this route
ipRouteType	Type of route
ipRouteProto	Routing mechanism by which this route was learned
ipRouteMask	Mask value for this route

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 the *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 the *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 MVS Diagnosis: Procedures* or to Chapter 3, "Diagnosing abends, loops, and hangs," on page 23 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

– NetView log (if the SNMP client is on an MVS host)

More documentation that might be needed is discussed in the analysis section.

Analysis

Table 64 shows symptoms of connection problems and refers to the steps needed for initial diagnosis of the error.

Table 64. NCPROUTE connection problems

Connection Problem	Analysis Steps
NCP client connection failure	1, 2, 7, 8, 10, 13
NCPROUTE connection failure	1, 3, 5, 6, 7, 8, 10, 11, 13
SNALINK LU0 connection failure	1, 3, 7, 8, 10, 12
SNMP Agent connection failure	4, 9, 10, 13

"Steps for NCPROUTE connection problems" gives the diagnostic steps referred to in Table 64.

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 (INTFACE 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 (INTFACE 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 27.

12. For SNMP problems, see Chapter 24, "Diagnosing Simple Network Management Protocol (SNMP) problems," on page 551.

13. For OMPROUTE problems, see Chapter 30, "Diagnosing OMPROUTE problems," on page 665. 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 *z/OS Communications Server: 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 65.

Table 65. NCPROUTE command terms

Term	Description
PING	Refers to z/OS UNIX oping, TSO PING, and the ping commands used on other platforms.
Traceroute	Refers to z/OS UNIX **otracert|traceroute**, TSO TRACERTE, and the **traceroute** commands used on other platforms.
NETSTAT ROUTE	Refers to the z/OS UNIX **onetstat -r**, TSO NETSTAT ROUTE, and the **netstat route** commands used on other platforms.
NETSTAT GATE	Refers to the z/OS UNIX **onetstat -g** and TSO NETSTAT GATE commands. This command is available only on the z/OS platform.
NETSTAT ARP	Refers to the z/OS UNIX onetstat -R ALL, TSO NETSTAT ALL, and the **netstat arp** commands used on other platforms.

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 66 on page 700 shows symptoms of PING failures and refers to the steps needed for initial diagnosis of the error.

Table 66. NCPROUTE routing failures

Routing Failure	Analysis Steps
Incorrect response	1, 2, 3, 4, 5, 6, 7, 8
Timeouts	2, 9

Steps for analyzing routing failures: This section gives the diagnostic steps referred to in Table 66.

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 36.

---

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 27.

8. For more information about diagnosing PING problems, refer to "Using the Ping command" on page 34.

9. For more information about diagnosing PING timeouts, refer to "Correcting timeout problems" on page 35.

"Steps for analyzing routing failures" on page 700 gives the diagnostic steps referred to in Table 66 on page 700.

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 67 on page 702 shows types of incorrect output and refers to the steps needed for initial diagnosis of the error.

Table 67. NCPROUTE incorrect output

Incorrect output	Analysis steps
TCP/IP incorrect output	1
SNALINK LU0 incorrect output	2
NCPROUTE incorrect output	3
SNMP agent or client incorrect output	4

Steps for diagnosing incorrect output: This section gives the diagnostic steps referred to in Table 67.

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 23.

| 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 17, "Diagnosing SNALINK LU0 problems," on page 473.
|

---

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 24, "Diagnosing Simple Network Management Protocol (SNMP) problems," on page 551.

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 68 shows symptoms of session outages and refers to the steps needed for initial diagnosis of the error.

Table 68. Symptoms of session outages

If this is the outage type	Then perform these steps
TCP/IP session outage	If the TCP/IP console shows a TCP/IP error message, refer to *z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)* and follow the directions for system programmer response for the message. If TCP/IP abended, see Chapter 3, "Diagnosing abends, loops, and hangs," on page 23.
SNALINK LU0 session outage	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 17, "Diagnosing SNALINK LU0 problems," on page 473.
NCPROUTE-to-NCP client session outage	If the NCPROUTE console shows an NCPROUTE error message, refer to *z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)* and follow the directions for system programmer response for the message.
NCPROUTE-to-SNMP agent session outage	If the SNMP agent console shows a SNMP error message, refer to *z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)* and follow the directions for system programmer response for the message. For more information about diagnosing SNMP problems, see Chapter 24, "Diagnosing Simple Network Management Protocol (SNMP) problems," on page 551.

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:

```
//NCPROUT 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 her 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.

Value	Meaning
0	Does not activate any traces.
1	Activates tracing of actions by the NCPROUTE server.
2	Activates tracing of all packets sent or received.
3	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.
4	Activates tracing of actions, packets sent or received, packet history, and packet contents. The RIP network routing information is included in the trace output.

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* 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 101 on page 706 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.

```
Trace
level
0      1 15:29:48 EZB3826I Port 580 assigned to ncprout
0        15:29:49 EZB3885I Input parameter(s): -t -t -dp
1        15:29:49 EZB4159I Global tracing actions started
2        15:29:49 EZB4160I Global tracing packets started
0        15:29:49 EZB3834I ************************************************
0      2 15:29:49 EZB4196I * Opening NCPROUTE profile dataset (DD:NCPRPROF)
0        15:29:49 EZB3834I ************************************************
0      3 15:29:49 EZB4055I ** Attempting to (re)start SNMP connection
0        15:29:49 EZB4059I Connecting to agent 9.67.116.66 on DPI port 1141
0        15:29:49 EZB4062I SNMP DPI connection established
0        15:29:50 EZB4064I 1.3.6.1.4.1.2.6.17. registered with SNMP agent0
1        15:29:50 EZB3829I Waiting for incoming packets
d      4 =============== Received datagram from NCP client (length=32)
d        0000    0100 0000 c1f0 f4d5
d        0008    f7f1 f1d7 f0f5 61f0
d        0010    f661 f9f4 40f1 f07a
d        0018    f1f0 7af4 f200 0000
d        0020(32)
0        15:29:51 EZB3834I ************************************************
0        15:29:51 EZB3876I * Hello from new client 9.67.116.65
0        15:29:51 EZB3877I * RIT dataset name: A04N711P
0        15:29:51 EZB3878I * RIT ID:  05/06/94 10:10:42
0        15:29:51 EZB3834I ************************************************
0        15:29:51 EZB3867I Acknowledge to 9.67.116.65: Hello Received
0        15:29:51 EZB3999I Establishing session with client 9.67.116.65
0        15:29:51 EZB3868I Acknowledge to 9.67.116.65: RIT Loaded OK
0        15:29:51 EZB4166I Session with client 9.67.116.65 started
1        15:29:51 EZB3829I Waiting for incoming packets
1        =============== Received datagram from NCP client (length=8)
d        0000    0800 0000 0a44 005c
d        0008(8)
0        15:29:51 EZB3834I ************************************************
0      5 15:29:51 EZB3898I * Recv:  Inactive Interface List from 9.67.116.65
0                          * 1 interface(s) found:
0        15:29:51 EZB3899I *   10.68.0.92 - TR92
0        15:29:51 EZB3834I ************************************************
0        15:29:51 EZB3834I ************************************************
0      6 15:29:51 EZB3956I * Processing interface NCSTALU1
0        15:29:51 EZB3834I ************************************************
0        15:29:51 EZB3959I Point-to-point interface, using dstaddr
0        15:29:51 EZB3962I Adding (sub)network address for interface
1        15:29:51 EZB3912I ifwithnet:  compare with NCSTALU1
1        15:29:51 EZB3915I netmatch 9.67.116.65 and 9.67.116.65
1        15:29:51 EZB4029I Tue Jun 28 15:29:51:
1      7 15:29:52 EZB4030I ADD destination 9.67.116.66, router 9.67.116.66, metric 1
1                          flags UP|HOST state INTERFACE|CHANGED|INTERNAL|PERM|SUBNET timer 0
```

Figure 101. NCPROUTE trace (Part 1 of 10)

```
0        15:29:52 EZB3834I ******************************************************
0        15:29:52 EZB3956I * Processing interface TR88
0        15:29:52 EZB3834I ******************************************************
0        15:29:52 EZB3960I This interface is not point-to-point
0        15:29:52 EZB3962I Adding (sub)network address for interface
1        15:29:52 EZB3912I ifwithnet:  compare with NCSTALU1
1        15:29:52 EZB3912I ifwithnet:  compare with TR88
1        15:29:52 EZB3915I netmatch 10.68.0.88 and 10.68.0.88
1        15:29:52 EZB4030I ADD destination 10.0.0.0, router 10.68.0.88, metric 1
1                          flags UP state INTERFACE|CHANGED|INTERNAL|SUBNET|PERM timer 0
1        15:29:52 EZB3912I ifwithnet:  compare with NCSTALU1
1        15:29:52 EZB3912I ifwithnet:  compare with TR88
1        15:29:52 EZB3915I netmatch 10.68.0.88 and 10.68.0.88
1        15:29:52 EZB4030I ADD destination 10.68.0.0, router 10.68.0.0, metric 1
1                          flags UP state INTERFACE|CHANGED|SUBNET|PERM timer 0
0        15:29:52 EZB3834I ******************************************************
0        15:29:52 EZB3956I * Processing interface TR92
0        15:29:52 EZB3834I ******************************************************
0        15:29:52 EZB3960I This interface is not point-to-point
0        15:29:52 EZB3948I Interface TR92 not up
0        15:29:52 EZB3834I ******************************************************
0      8 15:29:52 EZB3973I * Opening GATEWAYS dataset for client 9.67.116.65
0                          *   'TCPCS.NCPROUTE.GATEWAYS(A04N711P)'
0        15:29:52 EZB3834I ******************************************************
0        15:29:52 EZB3968I Start of GATEWAYS processing:
0        15:29:52 EZB4195I Option(s): trace.level 4 supply on default.router no
1        15:29:52 EZB4015I Client tracing actions started
2        15:29:52 EZB4016I Client tracing packets started
3        15:29:52 EZB4017I Client tracing history started
4        15:29:52 EZB4018I Client tracing packet contents started
0        15:29:52 EZB4198I (no etc.gateway definitions)
0        15:29:52 EZB4150I End of GATEWAYS processing
1        15:29:52 EZB3829I Waiting for incoming packets
d      9 =============== Received datagram from NCP client (length=80)
d     10 0000    0700 0000 9200 004a
         0008    4500 0048 09c0 0000
d        0010    3c11 79dc 0943 7442
d        0018    0943 7441 0208 0208
d        0020    0034 079e 0201 0000
d        0028    0002 0000 0943 7441
d        0030    0000 0000 0000 0000
d        0038    0000 0001 0002 0000
d        0040    0943 7000 0000 0000
d        0048    0000 0000 0000 0001
d        0050(80)
d        =============== Transport PDU header (length=8)
d        0000    0700 0000 9200 004a
d        0008(8)
d        =============== IP header (length=20)
d        0000    4500 0048 09c0 0000
d        0008    3c11 79dc 0943 7442
d        0010    0943 7441 8002 c12c
d        0018(24)
```

Figure 101. NCPROUTE trace (Part 2 of 10)

```
d        =============== UDP header (length=8)
d        0000    0208 0208 0034 079e
d        0008(8)
d        =============== UDP data (length=44)
d        0000    0201 0000 0002 0000
d        0008    0943 7441 0000 0000
d        0010    0000 0000 0000 0001
d        0018    0002 0000 0943 7000
d        0020    0000 0000 0000 0000
d        0028    0000 0001 0001 6e68
d        0030(48)
1     11 15:30:04 EZB3894I Transport from 9.67.116.65:  44 bytes of RIP data
2        15:30:04 EZB4045I     RESPONSE from 9.67.116.66 -> 520:
d     12 =============== RIP net info (length=20)
d        0000    0002 0000 0943 7441
d        0008    0000 0000 0000 0000
d        0010    0000 0001 8002 c12c
d        0018(24)
4        15:30:04 EZB4049I         destination 9.67.116.65 metric 1
d        =============== RIP net info (length=20)
d        0000    0002 0000 0943 7000
d        0008    0000 0000 0000 0000
d        0010    0000 0001 8002 c12c
4        0018(24)
1        15:30:04 EZB4049I         destination 9.67.112.0 metric 1
1        15:30:04 EZB4029I Tue Jun 28 15:30:04:
1     13 15:30:04 EZB4030I ADD destination 9.67.112.0, router 9.67.116.66, metric 2
1                          flags UP|GATEWAY state CHANGED|SUBNET timer 0
1     14 15:30:04 EZB3855I NCP_Add out to 9.67.116.65
1                          Route to: 9.67.112.0 via interface 9.67.116.65 to 9.67.116.66
1                          Metric: 2, Type Subnet
1        15:30:04 EZB3829I Waiting for incoming packets
1     15 15:30:20 EZB4011I client 9.67.116.65:  30 second timer expired (broadcast)
1        15:30:20 EZB3951I client 9.67.116.65:  supply 9.67.116.66 -> 0 via NCSTALU1
4     16 15:30:20 EZB4045I     RESPONSE to 9.67.116.66 -> 0:
d        =============== RIP net info (length=20)
d        0000    0002 0000 0943 7442
d        0008    0000 0000 0000 0000
d        0010    0000 0001 8002 c12c
d        0018(24)
4        15:30:20 EZB4049I         destination 9.67.116.66 metric 1
d        =============== RIP net info (length=20)
d        0000    0002 0000 0a00 0000
d        0008    0000 0000 0000 0000
d        0010    0000 0001 8002 c12c
d        0018(24)
4        15:30:20 EZB4049I         destination 10.0.0.0 metric 1
d        =============== RIP net info (length=20)
d        0000    0002 0000 0943 7000
d        0008    0000 0000 0000 0000
d        0010    0000 0002 8002 c12c
d        0018(24)
4        15:30:20 EZB4049I         destination 9.67.112.0 metric 2
```

Figure 101. NCPROUTE trace (Part 3 of 10)

```
d        =============== UDP data (length=64)
d        0000    0201 0000 0002 0000
d        0008    0943 7442 0000 0000
d        0010    0000 0000 0000 0001
d        0018    0002 0000 0a00 0000
d        0020    0000 0000 0000 0000
d        0028    0000 0001 0002 0000
d        0030    0943 7000 0000 0000
d        0038    0000 0000 0000 0002
d        0040(64)
d        =============== UDP header (length=8)
d        0000    0208 0208 0048 fd70
d        0008(8)
d        =============== IP header (length=20)
d        0000    4500 005c 0000 0000
d        0008    0411 bb88 0943 7441
d        0010    0943 7442 8002 c12c
d        0018(24)
d        =============== Transport PDU header (length=8)
d        0000    0700 0000 0943 7441
d        0008(8)
d        =============== Sending Transport PDU to NCP client (length=100)
d        0000    0700 0000 0943 7441
d        0008    4500 005c 0000 0000
d        0010    0411 bb88 0943 7441
d        0018    0943 7442 0208 0208
d        0020    0048 fd70 0201 0000
d        0028    0002 0000 0943 7442
d        0030    0000 0000 0000 0000
d        0038    0000 0001 0002 0000
d        0040    0a00 0000 0000 0000
d        0048    0000 0000 0000 0001
d        0050    0002 0000 0943 7000
d        0058    0000 0000 0000 0000
d        0060    0000 0002 0000 0001
d        0068(104)
:
1    17 15:30:20 EZB3948I Interface TR92 not up
1       15:30:20 EZB3829I Waiting for incoming packets
                            .
1       15:30:20 EZB3894I Transport from 9.67.116.65:  64 bytes of RIP data
2       15:30:20 EZB4045I     RESPONSE from 10.68.0.88 -> 520:
                            .
4       15:30:20 EZB4049I         destination 9.67.116.66 metric 1
                            .
4       15:30:20 EZB4049I         destination 10.68.0.0 metric 1
                            .
4       15:30:20 EZB4049I         destination 9.67.112.0 metric 2
1       15:30:20 EZB3829I Waiting for incoming packets
                            .
                            .
                            .
```

Figure 101. NCPROUTE trace (Part 4 of 10)

```
1        15:30:34 EZB3829I Waiting for incoming packets
1        15:30:50 EZB4011I client 9.67.116.65:  30 second timer expired (broadcast)
1        15:30:50 EZB3951I client 9.67.116.65:  supply 9.67.116.66 -> 0 via NCSTALU1
2        15:30:50 EZB4045I     RESPONSE to 9.67.116.66 -> 0:
                               .
4        15:30:50 EZB4049I         destination 9.67.116.66 metric 1
                               .
4        15:30:50 EZB4049I         destination 10.0.0.0 metric 1
                               .
4        15:30:50 EZB4049I         destination 9.67.112.0 metric 2
                               .
1        15:30:50 EZB3951I client 9.67.116.65:  supply 10.68.15.255 -> 0 via TR88
2        15:30:50 EZB4045I     RESPONSE to 10.68.15.255 -> 0:
                               .
4        15:30:50 EZB4049I         destination 9.67.116.66 metric 1
                               .
4        15:30:50 EZB4049I         destination 10.68.0.0 metric 1
                               .
4        15:30:50 EZB4049I         destination 9.67.112.0 metric 2
:
1        15:32:35 EZB3894I Transport from 9.67.116.65:  64 bytes of RIP data
2        15:32:35 EZB4045I     RESPONSE from 9.67.116.66 -> 520:
                               .
4        15:32:35 EZB4049I         destination 9.67.116.65 metric 1
                               .
4        15:32:35 EZB4049I         destination 10.0.0.0 metric 2
                               .
4        15:32:35 EZB4049I         destination 9.67.112.0 metric 16
1        15:32:35 EZB4029I Tue Jun 28 15:32:35:
1     18 15:32:35 EZB4036I CHANGE metric destination 9.67.112.0, router 9.67.116.66, from 2 to 16
      19 15:32:35 EZB3862I NCP_Delete out to 9.67.116.65:
                           Route to 9.67.112.0, type = Subnet
1        15:32:35 EZB3943I Send dynamic update
1        15:32:35 EZB3950I toall:  requested to skip interface NCSTALU1
1        15:32:35 EZB3951I client 9.67.116.65:  supply 10.68.15.255 -> 0 via TR88
2        15:32:35 EZB4045I     RESPONSE to 10.68.15.255 -> 0:
                               .
4        15:32:35 EZB4049I         destination 9.67.112.0 metric 16
                               .
1        15:32:35 EZB3948I Interface TR92 not up
1        15:32:35 EZB3945I Inhibit dynamic update for 2017537 usec
1        15:32:35 EZB3829I Waiting for incoming packets
                               .
1        15:32:35 EZB3894I Transport from 9.67.116.65:  24 bytes of RIP data
1        15:32:35 EZB4045I     RESPONSE from 10.68.0.88 -> 520:
                               .
4        15:32:35 EZB4049I         destination 9.67.112.0 metric 16
1        15:32:35 EZB3829I Waiting for incoming packets
```

Figure 101. NCPROUTE trace (Part 5 of 10)

```
1        15:32:50 EZB4011I client 9.67.116.65:  30 second timer expired (broadcast)
1        15:32:50 EZB3951I client 9.67.116.65:  supply 9.67.116.66 -> 0 via NCSTALU1
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
         15:32:50 EZB3951I client 9.67.116.65:  supply 10.68.15.255 -> 0 via TR88
2        15:32:50 EZB4045I     RESPONSE to 10.68.15.255 -> 0:
                                  .
1        15:32:50 EZB3948I Interface TR92 not up
1        15:32:50 EZB3829I Waiting for incoming packets
:
1        15:36:15 EZB3829I Waiting for incoming packets
1     20 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:
1     21 15:36:39 EZB4030I DELETE destination 9.67.112.0, router 9.67.116.66, metric 16
1                          flags UP|GATEWAY state SUBNET timer 300
1        15:36:39 EZB4011I client 9.67.116.65:  30 second timer expired (broadcast)
1        15:36:39 EZB3951I client 9.67.116.65:  supply 9.67.116.66 -> 0 via NCSTALU1
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 EZB3951I client 9.67.116.65:  supply 10.68.15.255 -> 0 via TR88
2        15:36:39 EZB4045I     RESPONSE to 10.68.15.255 -> 0:
                                  .
4        15:36:39 EZB4049I         destination 9.67.116.66 metric 1
                                  .
4        15:36:39 EZB4049I         destination 10.68.0.0 metric 1
:
1     22 15:43:01 EZB3895I Transport from 9.67.116.65:  43 bytes of SNMP data
1     23 15:43:01 EZB4182I SNMP request received from NCP client 9.67.116.65
d        =============== Object data (length=13)
d        0000    2b06 0102 0104 1501
d        0008    0709 4374 4207 39f8
d        0010(16)
d        =============== prefix + address (length=12)
d        0000    2b06 0104 0102 0611
d        0008    0943 7441 4207 39f8
d        0010(16)
```

Figure 101. NCPROUTE trace (Part 6 of 10)

```
d    =============== Inbound SNMP packet (post edit) (length=55)
d    0000 3035 0201 0004 0473
d    0008    6e6d 70a0 2a02 0115
d    0010    0201 0002 0100 301f
d    0018    301d 0619 2b06 0104
d    0020    0102 0611 0943 7441
d    0028    2b06 0102 0104 1501
d    0030    0709 4374 4205 0000
d    0038(56)
d    =============== Sending SNMP request to agent (length=55)
d    0000    3035 0201 0004 0473
d    0008    6e6d 70a0 2a02 0115
d    0010    0201 0002 0100 301f
d    0018    301d 0619 2b06 0104
d    0020    0102 0611 0943 7441
d    0028    2b06 0102 0104 1501
d    0030    0709 4374 4205 00f3
d    0038(56)
1    15:43:01 EZB3829I Waiting for incoming packets
1    15:43:01 EZB4194I SNMP sub-agent received DPI request
d    =============== Received DPI request from SNMP agent (length=69)
d    0000    0043 0201 0101 f14b
d    0008    f34b f64b f14b f44b
d    0010    f14b f24b f64b f1f7
d    0018    4bf9 4bf6 f74b f1f1
d    0020    f64b f6f5 4bf4 f34b
d    0028    f64b f14b f24b f14b
d    0030    f44b f2f1 4bf1 4bf7
d    0038    4bf9 4bf6 f74b f1f1
d    0040    f64b f6f6 0007 2b30
d    0048(72)
1    15:43:01 EZB4072I SNMP sub-agent:DPI GET request
                       (1.3.6.1.4.1.2.6.17.9.67.116.65.43.6.1.2.1.4.21.1.7.9.67.116.66) received
1    15:43:01 EZB4083I iproutenexthop.9.67.116.66
d    =============== Sending DPI response to SNMP agent (length=77)
d    0000    004b 0201 0105 00f1
d    0008    4bf3 4bf6 4bf1 4bf4
d    0010    4bf1 4bf2 4bf6 4bf1
d    0018    f74b f94b f6f7 4bf1
d    0020    f1f6 4bf6 f54b f4f3
d    0028    4bf6 4bf1 4bf2 4bf1
d    0030    4bf4 4bf2 f14b f14b
d    0038    f74b f94b f6f7 4bf1
d    0040    f1f6 4bf6 f600 8500
d    0048    0409 4374 4149 5f3c
d    0050(80)
1    15:43:01 EZB3829I Waiting for incoming packets
1    15:43:01 EZB4068I SNMP response received from agent 9.67.116.66
```

Figure 101. NCPROUTE trace (Part 7 of 10)

```
d        =============== Received SNMP response from agent (length=59)
d        0000    3039 0201 0004 0473
d        0008    6e6d 70a2 2e02 0115
d        0010    0201 0002 0100 3023
d        0018    3021 0619 2b06 0104
d        0020    0102 0611 0943 7441
d        0028    2b06 0102 0104 1501
d        0030    0709 4374 4240 0409
d        0038    4374 4196 95a2 8540
d        0040(64)
d        =============== Object data (length=25)
d        0000    2b06 0104 0102 0611
d        0008    0943 7441 2b06 0102
d        0010    0104 1501 0709 4374
d        0018    4240 2910 0000 0001
d        0020(32)
d        =============== prefix + address (length=12)
d        0000    2b06 0104 0102 0611
d        0008    0943 7441 2b06 0102
d        0010(16)
d        =============== Outbound SNMP packet (post edit) (length=47)
d        0000    302d 0201 0004 0473
d        0008    6e6d 70a2 2202 0115
d        0010    0201 0002 0100 3017
d        0018    3015 060d 2b06 0102
d        0020    0104 1501 0709 4374
d        0028    4240 0409 4374 4100
d        0030(48)
1        15:43:01 EZB4172I SNMP reply sent to NCP client 9.67.116.66
d        =============== UDP data (length=47)
d        0000    302d 0201 0004 0473
d        0008    6e6d 70a2 2202 0115
d        0010    0201 0002 0100 3017
d        0018    3015 060d 2b06 0102
d        0020    0104 1501 0709 4374
d        0028    4240 0409 4374 4168
d        0030(48)
d        =============== UDP header (length=8)
d        0000    00a1 040e 0037 ec9f
d        0008(8)
d        =============== IP header (length=20)
d        0000    4500 004b 0034 0000
d        0008    0411 a18e 0a44 0058
d        0010    0a44 0001 8002 c12c
d        0018(24)
d        =============== Transport PDU header (length=8)
d        0000    0700 0000 0a44 0058
d        0008(8)
```

Figure 101. 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 a18e 0a44 0058
d        0018    0a44 0001 00a1 040e
d        0020    0037 ec9f 302d 0201
d        0028    0004 0473 6e6d 70a2
d        0030    2202 0115 0201 0002
d        0038    0100 3017 3015 060d
d        0040    2b06 0102 0104 1501
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     24 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     25 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 101. NCPROUTE trace (Part 9 of 10)

```
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
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
⋮
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 EZB4039I *** End packet history ***
         15:44:31 EZB3829I Waiting for incoming packets
                             .
                             .
                             .
1        15:44:41 EZB3948I Interface TR88 not up
1        15:44:41 EZB3948I Interface TR92 not up
1        15:44:41 EZB3829I Waiting for incoming packets
                             .
                             .
                             .
```

Figure 101. NCPROUTE trace (Part 10 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.

13 NCPROUTE is adding a new route to its tables from the information received in the transport datagram.

14 NCPROUTE is issuing a request to the NCP client to add the route to its tables.

15 The NCP client 30-second timer has expired, so NCPROUTE supplies its routing tables to other routers.

16 NCPROUTE is responding to the request by sending its routing tables to the requesting router for the NCP client.

17 This line shows an inactive state for interface TR92.

18 The NCP client 3-minute timer expired. The client was broadcast as a network unreachable route (in the range metric 16—infinite), so NCPROUTE updates its routing tables for the NCP client.

19 NCPROUTE is deleting the NCP client from its tables.

20 The NCP client five-minute timer has expired for the route to 9.67.112.0.

21 NCPROUTE is deleting the route to 9.67.112.0 from its tables for the NCP client.

22 NCPR received a transport datagram from the SNMP client through NCP client 9.67.116.65.

23 NCPROUTE is processing the SNMP request.

24 NCPROUTE has received a status notification from the NCP client. The interface TR88 has become inactive.

25 The packet history for the interface TR88 is included in the trace because the interface has become inactive.

Chapter 32. Diagnosing X.25 NPSI problems

This chapter discusses how to diagnose X.25 NPSI problems and includes the following sections:

- "Operation" on page 718
- "Configuration requirements" on page 719
- "Sources of diagnostic information" on page 720
- "X.25 trace examples" on page 720
- "Steps for diagnosing logon problems" on page 723
- "Session hangs" on page 724

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 102 on page 718 shows the X.25 NPSI environment.

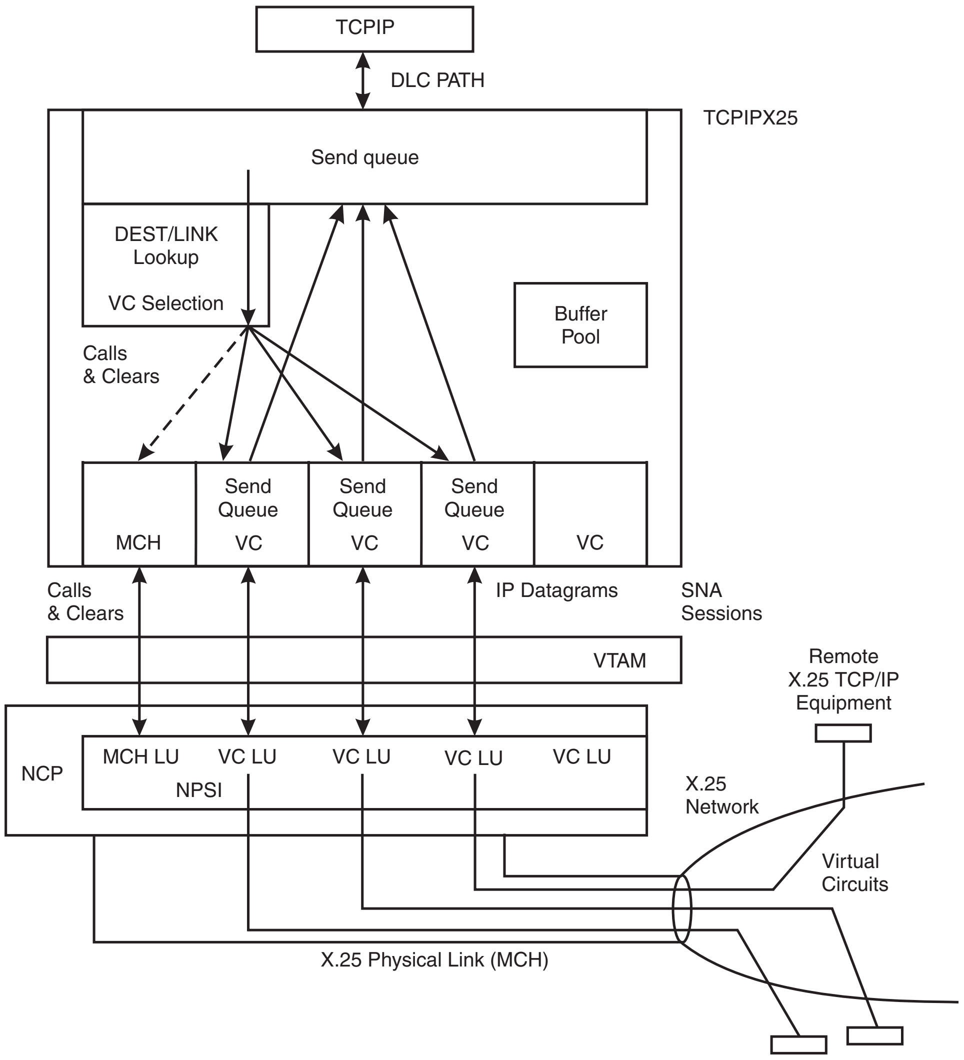


Figure 102. X.25 NPSI environment

Operation

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 102 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.
  - SSCPFM=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).

- X25.OUFT definition

  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 41.

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 41, 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	Irrecoverable error

The following example shows normal initialization:

```
EZB2111I VTAM ACB X25IPI1  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.
EZB2302I VC F001XU038    call accept packet sent (length=6)
EZB2000I 0000 .0....          0FF00102 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 0800AA6B 00BAF778 72ADA88E
EZB2000I 0020 0}.f9kq.,.PF;._n 307D0C66 B96BF118 AC085046 3B83DF6E
          ....data omitted for brevity...
EZB2000I 0110 =_3.            BD5F339D
EZB2331I VC F001XU038    data sent (length=277)
EZB2000I 0000 .E.......<.....} 00450001 14001000 003C017F 82820FFD
EZB2000I 0010 ...}&;.2k.:wxr-( 11820FFD 260000B2 6B00BAF7 7872ADA8
EZB2000I 0020 .0}.f9kq.,.PF;._ 8E307D0C 66B96BF1 18AC0850 463B83DF
          ....data omitted for brevity...
EZB2000I 0110 _=_3.           5FBD5F33 9D

EZB2336I VC F001XU038    inactivity timer expired
EZB2353I VC F001XU038    clear request packet sent (length=5)
```

```
EZB2000I 0000 .....            00011300 00
EZB2365I VC F001XU038    clear sent
EZB2333I VC F001XU038    packet received (length=1)
EZB2000I 0000 .                17
EZB2358I VC F001XU038    clear confirmed
EZB2351I VC F001XU038    connection terminated for 00000039: sent 1
                         received 1 dropped 0
EZB2352I VC 010 closed
```

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......... 0B081002 04008800 00003900 00003803
EZB2000I 0010 ....             420A0ACC
EZB2230I MCH XU038    packet received (length=5)
EZB2000I 0000 ...0.            0F0810F0 01
EZB2314I VC 0810XU038    call accepted by  user data
EZB2320I VC 0810XU038    NPSI logon LU VL038001
EZB2330I VC 0810XU038    call complete

EZB2336I VC 0810XU038    inactivity timer expired
EZB2353I VC 0810XU038    clear request packet sent (length=5)
EZB2000I 0000 .....            00011300 00
EZB2365I VC 0810XU038    clear sent
EZB2333I VC 0810XU038    packet received (length=1)
EZB2000I 0000 .                17
EZB2358I VC 0810XU038    clear confirmed
EZB2351I VC 0810XU038    connection terminated for 00000039: sent 5
                         received 5 dropped 0
EZB2352I VC 010 closed
```

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
              ...
EZB2021R VC 001 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:

```
EZB2091I HALT notice accepted, type 0
EZB2250I MCH XU038    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 720. 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:

Code	Meaning
00	DTE clearing. The remote system cleared the call.
01	Number busy. The called number cannot accept another call.
03	Invalid facility request. A facility requested by the caller is not subscribed or conflicts with a subscribed option.
05	Network congestion. Congestion conditions or some other problem within the network temporarily prevent the requested virtual circuit from being established.
09	Out of order. The called number is out of order.

0B	Access barred. The caller is not permitted to obtain a connection to the called number.
0D	Not obtainable. The called number is not assigned or is no longer assigned.
11	Remote procedure error. An X.25 protocol error at the remote equipment.
13	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 723.

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 719 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 33. Diagnosing IMS problems

This chapter 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 729
- "Common configuration mistakes" on page 731
- "Quick checklist for common problems" on page 731
- "Documentation references for problem diagnosis" on page 745

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 103 on page 728.

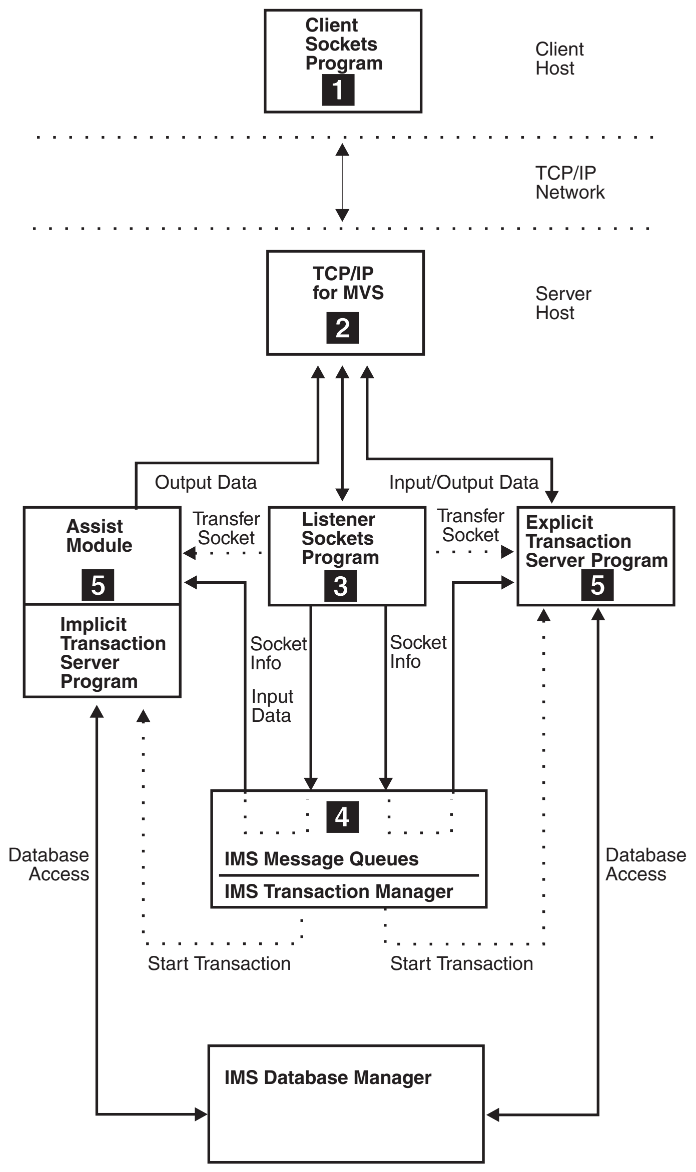


Figure 103. Components of the IMS TCP/IP services socket interface system

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.

5 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 103 on page 728.

This list of steps can be used to diagnose problems in starting components by identifying the prerequisites. The steps immediately preceding a step in which you are told to start a component are required to give definitions and configuration information that must be completed correctly before that component can be started. The reference keys in the steps refer to the components as shown in Figure 103 on page 728. All components except the client sockets program belong to the server host.

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 chapters 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. **2** 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. **5** 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. **4** 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.

   ```
   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. **3** 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. **1** 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 746 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 748 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 729 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 746 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 69 lists some of the problems related to starting or stopping one of the components in the IMS TCP/IP Services socket interface system.

Table 69. Component problems

Problem	Cause	Resolution
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 729. 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 733 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 69. Component problems (continued)

Problem	Cause	Resolution
The Listener is abending while accepting the TRM	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.	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*.

Connection problems

Table 70 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 70. Connection problems

Problem	Cause	Resolution
The client program is experiencing intermittent reject connect responses from TCP/IP	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.	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.
The TCP/IP socket connection to the client program is being broken immediately after the implicit IMS transaction is scheduled	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.	Verify that the TRANSACTION statements in the Listener configuration data set specify the TYPE parameter correctly.

Table 70. Connection problems (continued)

Problem	Cause	Resolution
Connection lockup for an implicit IMS transaction A connection lockup occurs when both the implicit IMS transaction and the client program are waiting for data from the other end of the socket connection.	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.	Use the IP packet trace facility to determine whether the client program is sending a valid EOM segment. See "Using IP packet trace" on page 745 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.
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 745 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 70. Connection problems (continued)

Problem	Cause	Resolution
Connection lockup for either an explicit or implicit IMS transaction A connection lockup occurs when both the IMS transaction and the client program are waiting for data from the other end of the socket connection.	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 745. 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 70. Connection problems (continued)

Problem	Cause	Resolution
The takesocket call issued by the IMS transaction fails **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.	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. 2. An IMS transaction not defined as multisegment to the IMS transaction manager is scheduled as soon as the TIM is added to the message queue. This gives the IMS transaction an opportunity to issue the takesocket before the givesocket is issued by the Listener. The takesocket fails with an error return code.	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 746. 2. Make certain the IMS transaction is defined as multisegment.
The client program is always receiving reject connect responses from TCP/IP	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.	Identify the client programs causing the problem using the NETSTAT command as specified in "Using NETSTAT" on page 746; 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*.

Table 70. Connection problems (continued)

Problem	Cause	Resolution
Connection lockup or loss when passing a socket connection from one explicit IMS transaction to another A connection lockup is when the socket connection reaches a state where it never completes.	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. 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.	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.

Error message and return code problems

Table 71 lists problems related to error responses.

Table 71. Error message and return code problems

Problem	Cause	Resolution
The client program is receiving a request status message (RSM)	The Listener sends this message to the client program when it detects an error condition.	Use the return and reason codes from the message to look up the explanation. See "Where to find return code documentation" on page 747.
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*	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.	Look up the meaning of the special return codes. See "Where to find return code documentation" on page 747.

Table 71. Error message and return code problems (continued)

Problem	Cause	Resolution
The Listener error messages are written to the MVS system console instead of the SYSPRINT data set	If the Listener experiences data set I/O errors, it redirects the error messages to the MVS system console.	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.

Socket data protocol problems

Table 72 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 72. Socket data protocol problems

Problem	Cause	Resolution
The Listener is not responding to the client program	1. If the TRM sent by the client program is incomplete, the Listener waits indefinitely for the rest of the message. 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.	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 745. 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 746.
All the input data sent from the client program is not being passed to the implicit IMS transaction from the Listener	Any input data written after the first EOM segment is ignored by the Listener.	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 745. 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.

Table 72. Socket data protocol problems (continued)

Problem	Cause	Resolution
Explicit IMS transaction is receiving garbled data from or sending garbled data to the client program	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.	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.

Table 72. Socket data protocol problems (continued)

Problem	Cause	Resolution
Implicit IMS transaction is receiving garbled data from or sending garbled data to the client program	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.	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.
The security exit does not validate user data from the client program	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.	Check that the security exit has been coded and built correctly as specified in the *z/OS Communications Server: IP IMS Sockets Guide*.

Table 72. Socket data protocol problems (continued)

Problem	Cause	Resolution
Data is corrupted after an implicit IMS transaction issues a GU	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.	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*.
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 73 on page 742 lists some problems related to building a component in the IMS TCP/IP Services socket interface system.

Table 73. IMS transaction build problems

Problem	Cause	Resolution
Unresolved external reference errors are causing the linker to fail when linking an IMS transaction	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. 2. The explicit IMS transaction link JCL is not including the MVS TCP/IP Services sockets library to resolve external references.	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*. 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*.

IMS database problems

Table 74 lists some problems related to unexpected IMS database actions or failures. They include changes not made or requests for changes that fail.

Table 74. IMS database problems

Problem	Cause	Resolution
The IMS transaction is terminating without performing the required function and without issuing any error messages	The IMS transaction might not be checking for interface errors.	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.
The client program is not receiving any data from the implicit IMS transaction, but is receiving a successful CSM	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).	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*.

Table 74. IMS database problems (continued)

Problem	Cause	Resolution
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 74. IMS database problems (continued)

Problem	Cause	Resolution
The IMS database manager commits the changes made by an IMS transaction, but the client program receives an error	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 sends 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. 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.	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. **Guideline:** This should be considered as an uncommon case.
The client program does not receive a valid CSM from an implicit IMS transaction	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.	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 745 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. Refer to the *z/OS Communications Server: IP IMS Sockets Guide* for the format of the EOM and CSM segments.

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 41 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 41, 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*. For details about tracing and reading the trace reports refer to the *IMS Version 8: Utilities Reference: System*.

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:
 Type   Bound to                Connected to          State    Conn
 ====   ========                ============          =====    ====
 Name: INETD1    Subtask: 006DB5B8
 Dgram  0.0.0.0..37             *..*                  UDP      00000058
 Dgram  0.0.0.0..13             *..*                  UDP      00000057
 Dgram  0.0.0.0..19             *..*                  UDP      00000056
 Dgram  0.0.0.0..9              *..*                  UDP      00000055
 Dgram  0.0.0.0..7              *..*                  UDP      00000054
 Stream 0.0.0.0..623            0.0.0.0..0            Listen   0000004B
 Stream 0.0.0.0..514            0.0.0.0..0            Listen   0000004D
 Stream 0.0.0.0..513            0.0.0.0..0            Listen   0000004C
 Stream 0.0.0.0..512            0.0.0.0..0            Listen   0000004E
 Stream 0.0.0.0..37             0.0.0.0..0            Listen   00000053
 Stream 0.0.0.0..7              0.0.0.0..0            Listen   0000004F
 Stream 0.0.0.0..13             0.0.0.0..0            Listen   00000052
 Stream 0.0.0.0..19             0.0.0.0..0            Listen   00000051
 Stream 0.0.0.0..9              0.0.0.0..0            Listen   00000050
 Name: OSNMPD    Subtask: 006DBA70
 Dgram  0.0.0.0..161            *..*                  UDP      00000013
 Stream 0.0.0.0..1027           0.0.0.0..0            Listen   00000014
 Name: TCPCS     Subtask: 00000000
 Stream 127.0.0.1..23           127.0.0.1..1033       Estblsh  00000045
 Stream 9.67.113.27..23         9.37.81.207..1096     ClosWait 00000039
 Name: TCPCS     Subtask: 006C57B0
 Stream 0.0.0.0..23             0.0.0.0..0            Listen   00000012
 Name: TCPCS     Subtask: 006D56F0
 Stream 127.0.0.1..1026          127.0.0.1..1025      Establsh 0000000F
 Name: TCPCS     Subtask: 006D5CF0
 Stream 0.0.0.0..1025           0.0.0.0..0            Listen   0000000C
 Stream 127.0.0.1..1025         127.0.0.1..1026       Establsh 00000010
 Name: USER18    Subtask: 006A3400
 Stream 127.0.0.1..1033         127.0.0.1..23         Establsh 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 34. Diagnosing restartable VMCF/TNF problems

This chapter describes how to diagnose restartable VMCF/TNF problems and contains the following sections:

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 chapter.

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.

Chapter 35. Diagnosing problems with IP CICS sockets

This chapter 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 752
- "CICS sockets application problems" on page 754
- "CICS sockets control blocks" on page 755
- "CICS trace" on page 755

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 Sockets 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 Sockets 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 transaction 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 sockets interface not initialized

If the CICS sockets 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 Sockets 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 sockets interface did not initialize. (See "Steps for diagnosing CICS sockets interface not initialized" on page 752.)

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

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.

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 sockets 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 sockets interface using the EZAO transaction.

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 sockets 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 36. 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 chapter 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 758
- "Analyzing client interface problems" on page 759

For complete information about Express Logon, refer to the following:
- The User's Guide at http://www.ibm.com/software/network/commserver/library/whitepapers/csos390.html
- *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.log`.

If you specify a debug level, but not logging, the DCAS attempts to open the default log file `/tmp/dcas.log`. If this fails, debugging is turned off.

For SYSLOGD, the DCAS uses the log facility local0.

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*.SYS1.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 HFS, verify that the KEYRING and STASHFILE keywords in the DCAS configuration file point to valid HFS 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:

  ```
  RDEFINE STARTED  DCAS.* STDATA(USER(dcasid))
  RDEFINEOPERCMDS (MVS.SERVMGR.DCAS) UACC(NONE)
  PERMIT MVS.SERVMGR.DCAS  CLASS(OPERCMDS) ACCESS(CONTROL) ID(dcasid)
  SETROPTS RACLIST(OPERCMDS)   REFRESH
  ```

- 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:

    ```
    SETOPTS 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 37. Diagnosing resolver problems

This chapter describes how to diagnose resolver problems and contains the following sections:

- "Steps for resolving the hostname"
- "TRACE RESOLVER" on page 762
- "CTRACE — RESOLVER" on page 775

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, GetHostByName 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 267).

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:

If ping for the hostname. . .	Then. . .	Solution
Succeeds, but another application fails when resolving the same hostname	The problem is with the resolver configuration for the application in the users environment.	Use the Trace Resolver to solve the problem.
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 27 to continue researching the problem.
Fails to convert the name to an IP address	The problem might be with the resolver configuration, 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:

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 27 to continue researching the problem.
Fails because DNS reports that the resource was not found	www.johndoe.com 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.
Succeeds	The problem in resolving the hostname using ping or another application might be in configuring the resolver.	The **dig** command bypasses the Resolver 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.

You know you are done when the application that previously failed to resolve the hostname can now resolve it.

TRACE RESOLVER

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, IEC141I 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, 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?
- 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.
- 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″.

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.
- An HFS 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 HFS 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 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:
  ```
  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:
  ```
  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:
  ```
  alloc dd(systcpt) da('user1.appl.restrace')
  ```
  To disable the Trace Resolver output, enter the following:
  ```
  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:

  ```
  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 HFS file as described below.
- For a new HFS 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 an HFS 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 the SYSPRINT DD if the TCPIP.DATA, statements TRACE RESOLVER or OPTIONS DEBUG, are specified. If the DD does not exist, no trace output is written.

z/OS batch environment

In the z/OS batch environment, use one of the following methods to specify the Trace Resolver output location:

- If the application resides in an HFS 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,OCREAT),
  //  PATHMODE=SIRWXU
  ```

  Note: In this example, OTRUNC is not specified on the PATHOPTS statement. This means the Trace Resolver output is appended to the HFS file. To avoid HFS 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 the STDOUT DD if the TCPIP.DATA statements, TRACE RESOLVER or OPTIONS DEBUG, are specified. If the DD does not exist, no trace output is written.

- To pass the RESOLVER_TRACE environment variable using BPXBATSL or BPXBATCH, enter the following:

  ```
   //STDENV  DD JCL statement
  ```

  The following shows an example:

  ```
  //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:

  ```
  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 an HFS file, a C03 ABEND might occur when the forked process ends.

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:**

    ```
    //RESOLVER PROC PARMS='CTRACE(CTIRESFL)'
    //*
    //EZBREINI EXEC PGM=EZBREINI,REGION=0M,TIME=1440,PARM=&PARMS
    //*
    //SETUP   DD   DSN=TPOUSER.RESOLVER.SETUP.DATA,DISP=SHR,FREE=CLOSE
    ```

  - **Setup File TPOUSER.RESOLVER.SETUP.DATA contains:**

    ```
    ;
     DEFAULTTCPIPDATA('TPOUSER.RESOLVER.DEFAULT.DATA')
    ;
    ; GLOBALTCPIPDATA(/ETC/TCPIPGLOBAL.DATA)
    #
    GLOBALTCPIPDATA('SYS1.TCPPARMS(RESGLOBL)')
    ```

  - **Global TCPIP.DATA file SYS1.TCPPARMS(RESGLOBL) contains:**

    ```
       # Note that DOMAIN is ignored because SEARCH is mutually exclusive
       # and SEARCH appears after DOMAIN.
       Domain abcxyz
       ; 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 mil
       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
   NSportAddr 53
2    ;ResolveVia UDP
   ResolverTimeout 3
   ResolverUdpRetries 1
1c   loaddbcstables unknown
   loaddbcstables big5
3    MVS026:  Hostname MVS026
```

– **Default TCPIP.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 TCPIP.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
3    HostName MVS000
   DomainOrigin edu
   ;
   NameServer   127.0.0.1  ; loopback
   #
2    ResolveVia TCP
   ResolverTimeout 22
1d   alwayswto xyz
   messagecase mixed
   loaddbcstables schinese
```

- **TSO commands issued to obtain the trace (gethostbyname):**

```
     alloc dd(systcpt) dsn(traceres) reuse
4      invoke a REXX application which issues gethostbyname for www.ibm.com
```

- **Trace Resolver ouput in USER55.TRACERES contains (gethostbyname):**

```
5   Resolver Trace Initialization Complete -> 2001/04/26 13:09:37.509773
1a  res_init Skipped option(s) on line     8: SYS1.TCPPARMS(RESGLOBL)
1b  res_init Skipped option(s) on line    11: SYS1.TCPPARMS(RESGLOBL)
1c  res_init Parse error on line    18: SYS1.TCPPARMS(RESGLOBL)
1d  res_init Parse error on line    14: USER55.TCPIP.DATA

6   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
19 Caller API             = TCP/IP Rexx Sockets
   (L) DataSetPrefix = USER55
3    (G) HostName      = MVS026
   (L) TcpIpJobName  = TCPCS
3 (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
3 (G) NameServer    = 9.67.128.82
                       9.67.128.255
   (G) NsPortAddr    = 53            3 (G) ResolverTimeout    = 3
2    (*) ResolveVia    = UDP           (G) ResolverUdpRetries = 1
   (*) Options NDots = 1
   (*) SockNoTestStor
   (*) AlwaysWto     = NO            (L) MessageCase        = MIXED
   (G) LoadDbcsTable = BIG5
11    (*) LookUp = DNS LOCAL
  res_init Succeeded
4   GetHostByName Resolving Name:  WWW.IBM.COM
  res_search(WWW.IBM.COM, C_IN, T_A)
  res_search Host Alias Search found no alias
  res_querydomain(WWW.IBM.COM., , C_IN, T_A)
  res_querydomain resolving name:  WWW.IBM.COM.
  res_query(WWW.IBM.COM., C_IN, T_A)
  res_mkquery(QUERY, WWW.IBM.COM., C_IN, T_A)
7   res_mkquery created message:
  * * * * * Beginning of Message * * * * *
   Query Id:                62981
   Flags:                   00000001 00000000
   Flags set:               recurDes
   OpCode:                  QUERY
   Response Code:           NOERROR

   Number of Question RRs:  1
   Question 1:
   WWW.IBM.COM
   Type (0X0001) T_A  Class (0X0001) C_IN

   Number of Answer RRs:  0
   Number of Authority RRs:  0
   Number of Additional RRs:  0
  * * * * * End of Message * * * * *
8   res_send Sending query to Name Server 9.67.128.82
   BPX1SOC:  RetVal = 0, RC = 0, Reason = 0x00000000
   BPX1IOC:  RetVal = 0, RC = 0, Reason = 0x00000000
   BPX1STO:  RetVal = 29, RC = 0, Reason = 0x00000000
   BPX1SEL:  RetVal = 1, RC = 0, Reason = 0x00000000
   BPX1RCV:  RetVal = 139, RC = 0, Reason = 0x00000000
   UDP Data Length: 139
9   res_send received data via UDP.  Message received:
  * * * * * Beginning of Message * * * * *
   Query Id:                62981
   Flags:                   10000001 10000000
   Flags set:               resp recurDes recurAvl
   OpCode:                  QUERY
   Response Code:           NOERROR

   Number of Question RRs:  1
   Question 1:
   WWW.IBM.COM
   Type (0X0001) T_A  Class (0X0001) C_IN

   Number of Answer RRs:  4
   Answer 1:
   WWW.IBM.COM
   Type (0X0001) T_A  Class (0X0001) C_IN
   TTL:  0093 (0 days, 0 hours, 1 minutes, 33 seconds)
   129.42.16.99
   Answer 2:
   WWW.IBM.COM
   Type (0X0001) T_A  Class (0X0001) C_IN
   TTL:  0093 (0 days, 0 hours, 1 minutes, 33 seconds)
   129.42.17.99
   Answer 3:
   WWW.IBM.COM
   Type (0X0001) T_A  Class (0X0001) C_IN
   TTL:  0093 (0 days, 0 hours, 1 minutes, 33 seconds)
   129.42.18.99
   Answer 4:
   WWW.IBM.COM
   Type (0X0001) T_A  Class (0X0001) C_IN
   TTL:  0093 (0 days, 0 hours, 1 minutes, 33 seconds)
   129.42.19.99

   Number of Authority RRs:  2
   Authority 1:
   WWW.IBM.COM
   Type (0X0002) T_NS  Class (0X0001) C_IN
   TTL:  33827 (0 days, 9 hours, 23 minutes, 47 seconds)
   ns.nyc.ibm.com
   Authority 2:
   WWW.IBM.COM
   Type (0X0002) T_NS  Class (0X0001) C_IN
   TTL:  33827 (0 days, 9 hours, 23 minutes, 47 seconds)
   ns2.nyc.ibm.com

   Number of Additional RRs:  0
  * * * * * End of Message * * * * *
   BPX1CLO:  RetVal = 0, RC = 0, Reason = 0x00000000
  res_send Succeeded
  res_query Succeeded
  res_querydomain Succeeded
  res_search Succeeded
10 GetHostByName Succeeded:  IP Address(es) found:
    IP Address(1) is 129.42.19.99
    IP Address(2) is 129.42.18.99
    IP Address(3) is 129.42.17.99
    IP Address(4) is 129.42.16.99
```

- **TSO commands issued to obtain the trace (getaddrinfo):**

```
   alloc dd(systcpt) dsn(traceres) reuse
12    ping cs390-2e

 Ping CS V1R5: Pinging host CS390-2E.tcp.raleigh.ibm.com
 at IPv6 address fec9:c2d4::9:67:115:7
 sendto(): EDC8130I Host cannot be reached.
```

- **Trace Resolver ouput in USER55.TRACERES contains (getaddrinfo):**

```
5   Resolver Trace Initialization Complete -> 2002/02/08 11:40:37.237177
1a  res_init Skipped option(s) on line     8: SYS1.TCPPARMS(RESGLOBL)
1b  res_init Skipped option(s) on line    11: SYS1.TCPPARMS(RESGLOBL)
1c  res_init Parse error on line    18: SYS1.TCPPARMS(RESGLOBL)
1d  res_init Parse error on line    14: USER55.TCPIP.DATA

6   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
19 Caller API             = TCP/IP Sockets Extended
   (L) DataSetPrefix = USER55
3    (G) HostName      = MVS026
   (L) TcpIpJobName  = TCPCS
3 (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
 3 (G) NameServer    = 9.67.128.82
                        9.67.128.255
    (G) NsPortAddr    = 53             3 (G) ResolverTimeout    = 3
 2    (*) ResolveVia    = UDP            (G) ResolverUdpRetries = 1
    (*) Options NDots = 1
    (*) SockNoTestStor
    (*) AlwaysWto     = NO             (L) MessageCase        = MIXED
    (G) LoadDbcsTable = BIG5
 11    (*) LookUp = DNS LOCAL
   res_init Succeeded
 12 GetAddrinfo Invoked with following inputs:
   Host Name:  CS390-2E
   No Service operand specified
   Hints parameter supplied with settings:
       ai_family = 0, ai_flags = 0x00000062
       ai_protocol = 0, ai_socktype = 0
 13 GetAddrInfo Opening Socket for IOCTLs
 BPX1SOC:  RetVal = 0, RC = 0, Reason = 0x00000000
GetAddrInfo Opened Socket 0x00000000
 14 GetAddrInfo Both IPv4 and IPv6 Interfaces Exist
GetAddrInfo Host Alias Search found no alias
res_querydomain(CS390-2E, tcp.raleigh.ibm.com, C_IN, T_AAAA)
res_querydomain resolving name:  CS390-2E.tcp.raleigh.ibm.com
 15 res_query(CS390-2E.tcp.raleigh.ibm.com, C_IN, T_AAAA)
res_mkquery(QUERY, CS390-2E.tcp.raleigh.ibm.com, C_IN, T_AAAA)
res_mkquery created message:
* * * * * Beginning of Message * * * * *
 Query Id:                63243
 Flags:                   00000001 00000000
 Flags set:               recurDes
 OpCode:                  QUERY
 Response Code:           NOERROR

 Number of Question RRs:  1
 Question 1:
 CS390-2E.tcp.raleigh.ibm.com
 Type (0X001C) T_AAAA  Class (0X0001) C_IN

 Number of Answer RRs:  0
 Number of Authority RRs:  0
 Number of Additional RRs:  0
* * * * * End of Message * * * * *
 8 res_send Sending query to Name Server 9.67.128.82
 BPX1STO:  RetVal = 46, RC = 0, Reason = 0x00000000
 BPX1SEL:  RetVal = 1, RC = 0, Reason = 0x00000000
 BPX1RCV:  RetVal = 109, RC = 0, Reason = 0x00000000
 UDP Data Length: 109

 9 res_send received data via UDP.  Message received:
* * * * * Beginning of Message * * * * *
 Query Id:                63243
 Flags:                   10000101 10000000
 Flags set:               resp auth recurDes recurAvl
 OpCode:                  QUERY
 Response Code:           NOERROR

 Number of Question RRs:  1
 Question 1:
 CS390-2E.tcp.raleigh.ibm.com
 Type (0X001C) T_AAAA  Class (0X0001) C_IN

 Number of Answer RRs:  1
 Answer 1:
 CS390-2E.tcp.raleigh.ibm.com
 Type (0X001C) T_AAAA  Class (0X0001) C_IN
 TTL:  86400 (1 days, 0 hours, 0 minutes, 0 seconds)
 FEC9:C2D4::9:67:115:7

 Number of Authority RRs:  1
 Authority 1:
 tcp.raleigh.ibm.com
 Type (0X0002) T_NS  Class (0X0001) C_IN
 TTL:  86400 (1 days, 0 hours, 0 minutes, 0 seconds)
 buzz.tcp.raleigh.ibm.com

 Number of Additional RRs:  1
 Additional 1:
 buzz.tcp.raleigh.ibm.com
 Type (0X0001) T_A  Class (0X0001) C_IN
 TTL:  86400 (1 days, 0 hours, 0 minutes, 0 seconds)
 9.67.128.82
* * * * * End of Message * * * * *
                     res_send Succeeded
res_query Succeeded
res_querydomain Succeeded
res_querydomain(CS390-2E, tcp.raleigh.ibm.com, C_IN, T_A)
res_querydomain resolving name:  CS390-2E.tcp.raleigh.ibm.com
16 res_query(CS390-2E.tcp.raleigh.ibm.com, C_IN, T_A)
res_mkquery(QUERY, CS390-2E.tcp.raleigh.ibm.com, C_IN, T_A)
7 res_mkquery created message:
* * * * * Beginning of Message * * * * *
 Query Id:                63244
 Flags:                   00000001 00000000
 Flags set:               recurDes
 OpCode:                  QUERY
 Response Code:           NOERROR

 Number of Question RRs:  1
 Question 1:
 CS390-2E.tcp.raleigh.ibm.com
 Type (0X0001) T_A  Class (0X0001) C_IN

 Number of Answer RRs:  0
 Number of Authority RRs:  0
 Number of Additional RRs:  0
* * * * * End of Message * * * * *
8 res_send Sending query to Name Server 9.67.128.82
 BPX1STO:  RetVal = 46, RC = 0, Reason = 0x00000000
 BPX1SEL:  RetVal = 1, RC = 0, Reason = 0x00000000
 BPX1RCV:  RetVal = 97, RC = 0, Reason = 0x00000000
 UDP Data Length: 97
9 res_send received data via UDP.  Message received:
* * * * * Beginning of Message * * * * *
 Query Id:                63244
 Flags:                   10000101 10000000
 Flags set:               resp auth recurDes recurAvl
```

```
  OpCode:                  QUERY
  Response Code:           NOERROR

  Number of Question RRs:  1
  Question 1:
  CS390-2E.tcp.raleigh.ibm.com
  Type (0X0001) T_A  Class (0X0001) C_IN

  Number of Answer RRs:  1
  Answer 1:
  CS390-2E.tcp.raleigh.ibm.com
  Type (0X0001) T_A  Class (0X0001) C_IN
  TTL:  3600 (0 days, 1 hours, 0 minutes, 0 seconds)
  9.67.115.7

  Number of Authority RRs:  1
  Authority 1:
  tcp.raleigh.ibm.com
  Type (0X0002) T_NS  Class (0X0001) C_IN
  TTL:  86400 (1 days, 0 hours, 0 minutes, 0 seconds)
  buzz.tcp.raleigh.ibm.com

  Number of Additional RRs:  1
  Additional 1:
  buzz.tcp.raleigh.ibm.com
  Type (0X0001) T_A  Class (0X0001) C_IN
  TTL:  86400 (1 days, 0 hours, 0 minutes, 0 seconds)
  9.67.128.82
* * * * * End of Message * * * * *
res_send Succeeded
res_query Succeeded
res_querydomain Succeeded
 17 GetAddrInfo Returning Zero as Port Number
GetAddrInfo Built 2 Addrinfos
 13 GetAddrInfo Closing IOCTL Socket 0x00000000
 BPX1CLO:  RetVal = 0, RC = 0, Reason = 0x00000000
 18 GetAddrInfo Succeeded:  IP Address(es) found:
  IP Address(1) is FEC9:C2D4::9:67:115:7
  IP Address(2) is 9.67.115.7
***********************************************************************
FreeAddrInfo Called to free addrinfo structures
FreeAddrInfo Succeeded, Freed 2 Addrinfos
***********************************************************************
```

The following describes highlighted numbered areas of the example setup files and example trace resolver output.

1
: Errors deliberately entered into this example to show action taken.

  a
  : Line 8 in the global file has 7 search values - the maximum is 6. The seventh one is ignored.

  b
  : Line 11 in the global file has 5 sortlist values - the maximum is 4. The fifth one is ignored.

  c
  : Line 18 in the global file has a value for LOADDBCSTABLES that is not valid. The value is ignored.

d

Line 14 in the local file has a value for ALWAYSWTO that is not valid. The value is ignored and the default is used.

2

ResolveVia is UDP even though the local file had ResolveVia TCP. UDP is used because GLOBALTCPIPDATA 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 is assigned. In this example, resolver statements are not specified as shown by ResolveVia in the global file being commented out.

3

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 GLOBALTCPIPDATA specified all resolver related statements in a local file is ignored (for example, DOMAINORIGIN, NAMESERVER and RESOLVERTIMEOUT).

4

A REXX application calls GetHostByName. The flow through the resolver API calls shows the parameters being passed.

5

Trace output reports the date and time the command was issued.

6

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
L	Local file

7

`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.

8

`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.

9

`res_send` receives a message from DNS. Note that there are 4 IP addresses for this name (4 answers).

10

GetHostByName function reports success, listing the IP addresses returned. Note the order of the addresses matches the sortlist specification.

11

LookUp specifies the order in which the DNS and the local host file are to be used for name resolution. It can be:

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)

12

Ping calls GetAddrinfo. 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.

13

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 **14**). A socket, separate from the one used to send DNS queries, is opened for communicating with the stacks.

14

The resolver detected that the system can handle both IPv4 and IPv6 addresses.

15

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*.

16

The resolver sends the IPv4 query (T_A) to DNS second. 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*.

17

Because no Service operand was passed as input to Getaddrinfo, there is no service resolution to perform, so any sockaddr returned have a port number=0.

18

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.

19

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

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 CTIRES*xx* 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 41). 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.

```
/********************************************************************/
/*                                                                  */
/*  IBM Communications Server for z/OS                              */
/*  SMP/E Distribution Name:  CTIRES00                              */
/*                                                                  */
/*  PART Name:  CTIRES00                                            */
/*                                                                  */
/*                                                                  */
/*  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       */
/*                                                                  */
/********************************************************************/
 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 128M                      */
 /*            CTRACE buffers reside in Resolver Private storage  */
 /*            which is in the regions address space.             */
 /* ------------------------------------------------------------- *
           BUFSIZE(16M)
 /*        JOBNAME(jobname1,...)                                  */
 /*        ASID(Asid1,...)                                        */
 /*        WTR(wtr_procedure)                                     */
 /* ------------------------------------------------------------- */
 /*   OPTIONS: NAMES OF FUNCTIONS TO BE TRACED, OR "ALL"          */
 /* ------------------------------------------------------------- */
 /*        OPTIONS(            */
 /*               'ALL     '   */
 /*              ,'MINIMUM '   */
 /*               )            */
```

When formatting the Resolver trace, use the CTRACE command. See Chapter 5, “TCP/IP services traces and IPCS support,” on page 41 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(*nn*) command, where *nn* is the largest number of characters that displays on one line.

Chapter 38. 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 chapter 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 780

Activating the SNTPD debug trace

To activate the SNTPD debug trace, specify the -d or -df option when starting SNTPD via z/OS UNIX shell or the MVS started procedure.

Base your how you prefer the debug message to be written.

If this option is used...	Then (phrase) . . .
-d option	Messages are written to stdout.
-df option **Restriction:** You must specify a path name and file name.	messages are written to this file. Logging messages are written to SYSLOGD

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

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 sntpd
```

Sample SNTPD debug output

Refer to *z/OS MVS Diagnosis: Procedures* or see Chapter 3, “Diagnosing abends, loops, and hangs,” on page 23, 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 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: 9.67.2.2
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 EZZ9600I SNTP server ready
Tue Apr  2 15:28:15 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
```

Part 4. Appendixes

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 784
- "FFST probe descriptions" on page 784

FFST probe index

Table 75 provides an index of FFST probes by probe name and component:

Table 75. FFST probes

Probe name	Component	Reference probes
EZBIEDST	IOCTL Enablement	IOCTL Enablement Probes
EZBPADST	Pascal API	Pascal API Probes
EZBPFDST	PFS IOCTL	PFS IOCTL Probes
EZBTRDST	TELNET Transform	TELNET Transform Probes
EZBTTDST	TELNET SRV	TELNET SRV Probes
EZBCFDST	Configuration Services	Configuration Services Probes
EZBITDST	Infrastructure	Infrastructure
EZBCABND	TCP/IP Base	TCP/IP Base
EZBTCDST	Transmission Control Protocol	Transmission Control Protocol Probes
EZBUDDST	Update Datagram Protocol Layer	Update Datagram Protocol Layer Probes
EZBSKDST	Streams	Streams Probes
EZBRWDST	Raw IP Layer	Raw IP Layer Probes
EZBIPDST	Internet Protocol	Internet Protocol Probes

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 76 lists the naming conventions for FFST probes used in TCP/IP.

Table 76. FFST naming conventions

Characters	Example	Description
1, 2, 3	EZB	These characters represent the product identifier. For TCP/IP, these characters are EZB.
4, 5	IT	These characters represent the TCP/IP component identifier, IT is the component identifier for Infrastructure Services.
6	C	For TCP/IP, this character is usually a C.
7, 8	01	These characters represent the probe number. This number is not duplicated.

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 75 on page 783 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 77 lists the FFST probes for IOCTL enablement (EZBIECxx).

Table 77. IOCTL enablement probes

Probe name	Module	Description	Dump type
EZBIEC01	EZBIEHOM	Logical interface missing	FULL
EZBIEC03	EZBIEPRT	Add Portlist Member Failure	FULL
EZBIEC04	EZBIECTL	IOCTL Command is Not 99	FULL
EZBIEC05	EZBIECTL	Null Queue Pointers	FULL
EZBIEC06	EZBIECTL	Invalid IOCTL Message	FULL
EZBIEC07	EZBIEINI	m_begin Interval Exceeded	FULL

Guideline: When the EZBIE07 FFST probe is hit, it is recommended that you recycle the TCP/IP stack because it is not stable.

Table 78 lists the FFST probes for Infrastructure Services (EZBITCxx).

Table 78. Infrastructure services probes

Probe name	Module	Description	Dump type
EZBITC01	EZBITPCI	Connect entry failure	FULL
EZBITC02	EZBITTUB	Timer cancel for BAD TQE	FULL
EZBITC05	EZBITTUB	Timer cancel for BAD TQE2	FULL
EZBITC07	EZBITDUS	Invalid ASCB	FULL
EZBITC08	EZBITPCT	Entry table destroy failure	FULL
EZBITC09	EZBPTDEF	Pat tree key zero	FULL

Table 78. Infrastructure services probes (continued)

Probe name	Module	Description	Dump type
EZBITC10	EZBPTDEF	Pat tree key too big	FULL
EZBITC11	EZBPTADD	Pat tree key exists	FULL
EZBITC13	EZBITKRA	Lock release error	FULL
EZBITC15	EZBITKRA	Lock release error - DUCB	FULL
EZBITC16	EZBITKRS	Suspend Lock Failure1	FULL
EZBITC17	EZBITKRS	DUCB mismatch	FULL
EZBITC18	EZBITKRS	Lock Suspend Failure2	FULL
EZBITC19	EZBITSCS	Storage size requested error	FULL
EZBITC21	EZBITSMT	Message triple release failure	FULL
EZBITC22	EZBITPCI	Create entry table failure	FULL
EZBITC23	EZBITPCI	TRESERVE linkage index failure	FULL

Table 79 lists the FFST probes for Pascal API (EZBPACxx).

Table 79. FFST probes for Pascal API

Probe name	Module	Description	Dump type
EZBPAC01	EZBPAISL	Streams operation software failure	FULL
EZBPAC02	EZBPAISL	Streams operation software failure	FULL
EZBPAC03	EZBPAMQY	Streams operation software failure	FULL
EZBPAC04	EZBPAMQY	Streams operation software failure	FULL
EZBPAC05	EZBPAPIN	Streams operation software failure	FULL
EZBPAC06	EZBPAPIN	Streams operation software failure	FULL
EZBPAC07	EZBPAPIN	Streams operation software failure	FULL
EZBPAC08	EZBPAPIN	Streams operation software failure	FULL
EZBPAC09	EZBPARCL	Streams operation software failure	FULL
EZBPAC10	EZBPAROP	Streams operation software failure	FULL
EZBPAC11	EZBPAROP	Streams operation software failure	FULL
EZBPAC12	EZBPAROP	Streams operation software failure	FULL
EZBPAC13	EZBPAROP	Streams operation software failure	FULL
EZBPAC14	EZBPAROP	Streams operation software failure	FULL

Table 79. FFST probes for Pascal API (continued)

Probe name	Module	Description	Dump type
EZBPAC15	EZBPAROP	Streams operation software failure	FULL
EZBPAC16	EZBPAROP	Streams operation software failure	FULL
EZBPAC17	EZBPARRV	Streams operation software failure	FULL
EZBPAC18	EZBPARRV	Streams operation software failure	FULL
EZBPAC19	EZBPARRV	Streams operation software failure	FULL
EZBPAC20	EZBPARSN	Streams operation software failure	FULL
EZBPAC21	EZBPART2	Function code error	FULL
EZBPAC22	EZBPATAB	Streams operation software failure	FULL
EZBPAC23	EZBPATAB	Streams operation software failure	FULL
EZBPAC24	EZBPATAB	Streams operation software failure	FULL
EZBPAC25	EZBPASTR	Invalid type of M_ERROR	FULL
EZBPAC26	EZBPASTR	Storage allocation failure	FULL
EZBPAC27	EZBPASTR	Unsupported option	FULL
EZBPAC28	EZBPASTR	Unsupported option	FULL
EZBPAC29	EZBPASTR	Unrecognized TPI	FULL
EZBPAC30	EZBPASTR	Streams operation software failure	FULL
EZBPAC31	EZBPASTR	Streams operation software failure	FULL
EZBPAC32	EZBPASTR	Storage allocation failure	FULL
EZBPAC33	EZBPASTR	Streams operation software failure	FULL
EZBPAC34	EZBPASTR	Streams operation software failure	FULL
EZBPAC35	EZBPASTR	Streams operation software failure	FULL
EZBPAC36	EZBPASTR	Streams operation software failure	FULL
EZBPAC37	EZBPASTR	Streams operation software failure	FULL
EZBPAC38	EZBPASTR	Streams operation software failure	FULL
EZBPAC39	EZBPASTR	Streams operation software failure	FULL
EZBPAC40	EZBPASTR	Streams operation software failure	FULL

Table 79. FFST probes for Pascal API (continued)

Probe name	Module	Description	Dump type
EZBPAC41	EZBPASTR	Streams operation software failure	FULL
EZBPAC42	EZBPASTR	Streams operation software failure	FULL
EZBPAC43	EZBPASTR	Storage allocation failure	FULL
EZBPAC44	EZBPASTR	Storage allocation failure	FULL
EZBPAC45	EZBPASTR	Storage allocation failure	FULL
EZBPAC46	EZBPAUCL	Streams operation software failure	FULL
EZBPAC47	EZBPAUNR	Streams operation software failure	FULL
EZBPAC48	EZBPAUNR	Streams operation software failure	FULL
EZBPAC49	EZBPAUNR	Streams operation software failure	FULL
EZBPAC50	EZBPAURV	Streams operation software failure	FULL
EZBPAC51	EZBPAURV	Streams operation software failure	FULL
EZBPAC52	EZBPAURV	Streams operation software failure	FULL
EZBPAC53	EZBPATOP	Streams operation software failure	FULL
EZBPAC54	EZBPATOP	Streams operation software failure	FULL
EZBPAC55	EZBPATOP	Streams operation software failure	FULL
EZBPAC56	EZBPATOP	Streams operation software failure	FULL
EZBPAC57	EZBPATOP	Streams operation software failure	FULL
EZBPAC58	EZBPATOP	Streams operation software failure	FULL
EZBPAC59	EZBPATOP	Streams operation software failure	FULL
EZBPAC60	EZBPAUOP	Streams operation software failure	FULL
EZBPAC61	EZBPAUOP	Streams operation software failure	FULL
EZBPAC62	EZBPAUOP	Streams operation software failure	FULL
EZBPAC63	EZBPAUOP	Streams operation software failure	FULL
EZBPAC64	EZBPAUOP	Streams operation software failure	FULL
EZBPAC65	EZBPAUOP	Streams operation software failure	FULL

Table 79. FFST probes for Pascal API (continued)

Probe name	Module	Description	Dump type
EZBPAC66	EZBPAUOP	TPI protocol error	FULL
EZBPAC67	EZBPATFR	Streams operation software failure	FULL
EZBPAC68	EZBPATFR	Streams operation software failure	FULL
EZBPAC69	EZBPATOA	Streams operation software failure	FULL
EZBPAC70	EZBPATOA	Streams operation software failure	FULL
EZBPAC71	EZBPATOA	Streams operation software failure	FULL
EZBPAC72	EZBPATOA	Streams operation software failure	FULL
EZBPAC73	EZBPATSN	Streams operation software failure	FULL
EZBPAC74	EZBPATST	Streams Operation Software Error	FULL
EZBPAC75	EZBPATTN	Streams operation software failure	FULL
EZBPAC76	EZBPATTN	Streams operation software failure	FULL
EZBPAC77	EZBPATTN	Streams operation software failure	FULL
EZBPAC78	EZBPAUSN	Streams operation software failure	FULL
EZBPAC79	EZBPAUST	Streams operation software failure	FULL
EZBPAC80	EZBPAUST	Streams operation software failure	FULL
EZBPAC81	EZBPATCL	Streams operation software failure	FULL
EZBPAC82	EZBPATCL	Allocate storage failure	FULL
EZBPAC83	EZBPATCL	Streams operation software failure	FULL
EZBPAC84	EZBPATCL	Allocate storage failure	FULL
EZBPAC85	EZBPATON	Streams Software Operation Error	FULL
EZBPAC86	EZBPATON	Streams operation software failure	FULL
EZBPAC87	EZBPATON	Streams operation software failure	FULL
EZBPAC88	EZBPATON	Streams operation software failure	FULL
EZBPAC89	EZBPATON	Streams operation software failure	FULL
EZBPAC90	EZBPATON	Streams operation software failure	FULL

Table 79. FFST probes for Pascal API (continued)

Probe name	Module	Description	Dump type
EZBPAC91	EZBPATON	Streams operation software failure	FULL
EZBPAC92	EZBPATON	Streams operation software failure	FULL
EZBPAC93	EZBPATON	Streams operation software failure	FULL
EZBPAC94	EZBPATON	Streams operation software failure	FULL
EZBPAC95	EZBPATON	TPI protocol error	FULL
EZBPAC96	EZBPATON	Streams operation software failure	FULL
EZBPAC97	EZBPATON	Streams operation software failure	FULL
EZBPAC98	EZBPATON	TPI protocol error	FULL
EZBPAC99	EZBPATON	Streams operation software failure	FULL
EZBPAC0A	EZBPATON	Streams operation software failure	FULL
EZBPAC0B	EZBPATON	TPI protocol error	FULL
EZBPAC0C	EZBPATON	Streams operation software failure	FULL
EZBPAC0D	EZBPATON	Streams operation software failure	FULL
EZBPAC0E	EZBPATON	TPI protocol error	FULL
EZBPACA0	EZBPATOP	Streams operation software failure	FULL
EZBPACA1	EZBPATOP	Streams operation software failure	FULL
EZBPACA2	EZBPATOP	Streams operation software failure	FULL
EZBPACB0	EZBPASTR	Storage allocate failure	FULL

Table 80 lists the FFST probes for PFS IOCTL (EZBPFCxx).

Table 80. PFS IOCTL probes

Probe name	Module	Description	Dump type
EZBPFC01	EZBPFIOC	SIOCSETTKN mismatch	FULL
EZBPFC02	EZBPFIOC	SIOCSETTKN mismatch	FULL

Table 81 lists the FFST probes for Telnet Transform (EZBTRCxx).

Table 81. Telnet transform probes

Probe name	Module	Description	Dump type
EZBTRC01	EZBTRCLT	Unexpected transform request	FULL
EZBTRC03	EZBTRGTI	Terminal ID mismatch	FULL
EZBTRC04	EZBTRMST	Unexpected transform WorkQ request	FULL
EZBTRC05	EZBTRRTI	Negative transform terminal value	FULL

Table 82 lists the FFST probes for Telnet SRV (EZBTTCxx).

Table 82. FFST probes for Telnet SRV

Probe name	Module	Description	Dump type
EZBTTC01	EZBTTCLS	Unlocatable server/vector table	FULL
EZBTTC02	EZBTTCLS	CVB lock failure	FULL
EZBTTC03	EZBTTTLT	Invalid TCVB token range	FULL
EZBTTC04	EZBTTTLT	Invalid TST entry	FULL
EZBTTC05	EZBTTTLT	Telnet token segment table not found	FULL

Table 83 lists FFST probes for Configuration Services (EZBCFCxx).

Table 83. Configuration services probes

Probe name	Module	Description	Dump type
EZBCFC01	EZACFFST	Unknown configuration error	FULL
EZBCFC02	EZACFTEL	Bad protocol Type 1	FULL
EZBCFC03	EZACFFST	Configuration bad parameters error	FULL
EZBCFC04	EZACFTEL	Socket closed	FULL
EZBCFC05	EZACFTEL	Bad protocol Type 2	FULL
EZBCFC06	EZACFTEL	Bad protocol Type 3	FULL

Table 84 lists the FFST probe for TCP/IP Base (EZBABCxx).

Table 84. TCP/IP Base probes

Probe name	Module	Description	Dump type
EZBABC01	EZBCABND	A C abend recovery failed	FULL

Table 85 on page 791 lists the FFST probes for Transmission Control Protocol (EZBTCCxx).

Table 85. Transmission Control Protocol probes

Probe name	Module	Description	Dump type
EZBTCC01	EZBTCSTR	Name on Open Does Not Match	FULL
EZBTCC02	EZBTCSTR	Could not allocate the SID	FULL
EZBTCC03	EZBTCSTR	Cannot Repeat Named Open	FULL
EZBTCC04	EZBTCSTR	Hashtable Insert Failure	FULL
EZBTCC05	EZBTCWRT	Not the Controlling Stream	FULL
EZBTCC06	EZBTCWRT	Not the Controlling Stream	FULL
EZBTCC07	EZBTCWRT	Not the Controlling Stream	FULL

Table 86 lists the FFST probes for Update Datagram Protocol Layer (EZBUDCxx).

Table 86. Update Datagram Protocol Layer probes

Probe name	Module	Description	Dump type
EZBUDC01	EZBUDEXC	DMUX Machine Index Failure	FULL
EZBUDC02	EZBUDEXC	DMUX Machine Index Failure	FULL
EZBUDC03	EZBUDEXC	SNMP Machine Index Failure	FULL
EZBUDC04	EZBUDSTR	Name on Open Does Not Match	FULL
EZBUDC05	EZBUDSTR	Allocate the MUCB SID failure	FULL
EZBUDC06	EZBUDSTR	Stack is Already Active	FULL
EZBUDC07	EZBUDSTR	Unlock for Machine Index Failure	FULL
EZBUDC08	EZBUDSTR	Unlock for Machine Index Failure	FULL
EZBUDC09	EZBUDSTR	Unlock for Machine Index Failure	FULL
EZBUDC10	EZBUDWRT	Unknown Primitive Error Exit	FULL
EZBUDC11	EZBUDWRT	Unknown Primitive Error Exit	FULL
EZBUDC12	EZBUDWRE	Matching Prefix Error	FULL
EZBUDC13	EZBUDWRE	Matching Prefix Error	FULL

Table 87 lists the FFST probes for Streams (EZBSKCxx).

Table 87. Streams probes

Probe name	Module	Description	Dump type
EZBSKC01	EZBSKVRB	Streams Are Not Functioning (TSDX_Streams_vcastint)	FULL
EZBSKC02	EZBSKVRB	Unsupported Message Type	FULL

Table 88 lists the FFST probes for Raw IP Layer (EZBRWCxx).

Table 88. Raw IP Layer probes

Probe name	Module	Description	Dump type
EZBRWC01	EZBRWWRI	WILD TPI Primitive to RAW	FULL
EZBRWC02	EZBRWWRI	Invalid Messages	FULL
EZBRWC03	EZBRWSTR	Name on Open Does Not Match	FULL
EZBRWCO4	EZBRWSTR	Could Not Allocate the MRCB SID	FULL
EZBRWCO5	EZBRWSTR	Stack is Already Active	FULL

Table 89 lists the FFST probes for Internet Protocol (EZBIPCxx).

Table 89. FFST probes for Internet Protocol

Probe name	Module	Description	Dump type
EZBIPC01	EZBIPSTR	Not a Clone Open	FULL

Table 90 lists the FFST probes for the Cross-System Coupling Facility (XCF) (EZBXFCxx).

Table 90. XCF probes

Probe name	Module	Description	Dump type
EZBXFC01	EZBXFINI	Join Failed	FULL
EZBXFC02	EZBXFINI	Second Query Failed	FULL
EZBXFC03	EZBXFINI	First Query Failed	FULL
EZBXFC04	EZBXFMSI	MsgI Failed	FULL
EZBXFC05	EZBXFMSO	MsgO Failed	FULL

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:

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 104 depicts a router and a bridge. The router connects Network 1 to Network 2 to form an intranet.

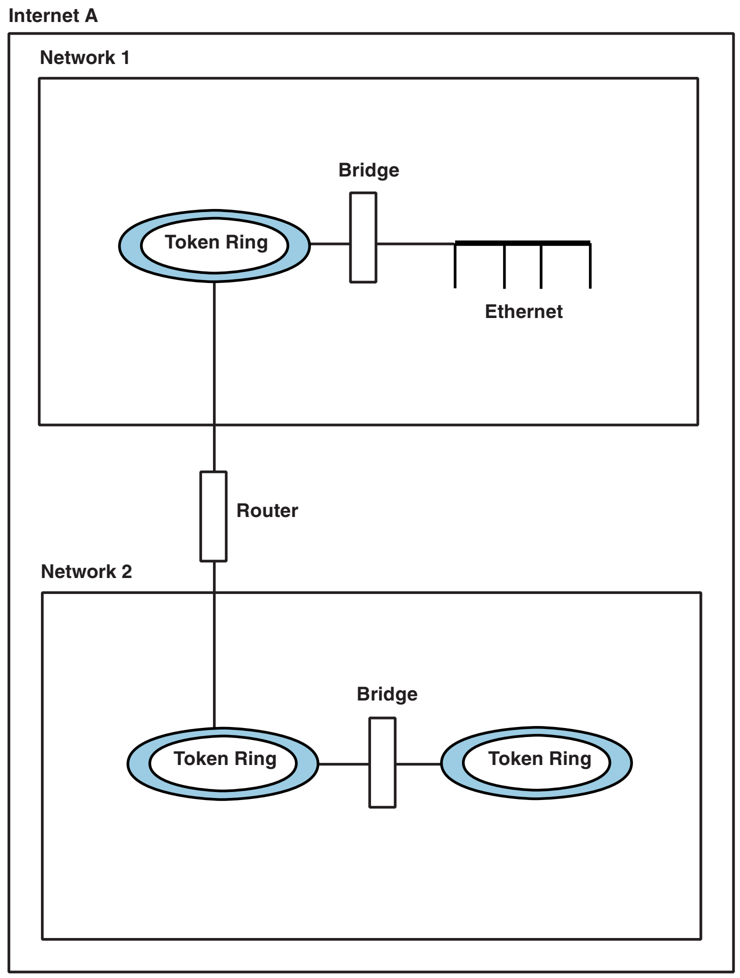


Figure 104. Routers and bridges within an internet

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 105 on page 795 shows the relationship between MTU and frame size.

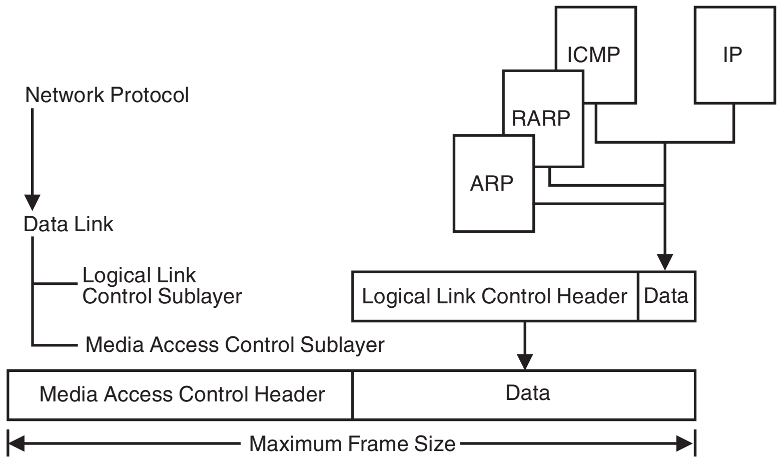

ICMP
IP
Network Protocol
RARP
ARP
Data Link
Logical Link Control Sublayer
Logical Link Control Header
Data
Media Access Control Sublayer
Media Access Control Header
Data
Maximum Frame Size

Figure 105. Relationship of MTU to 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 106 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 106.

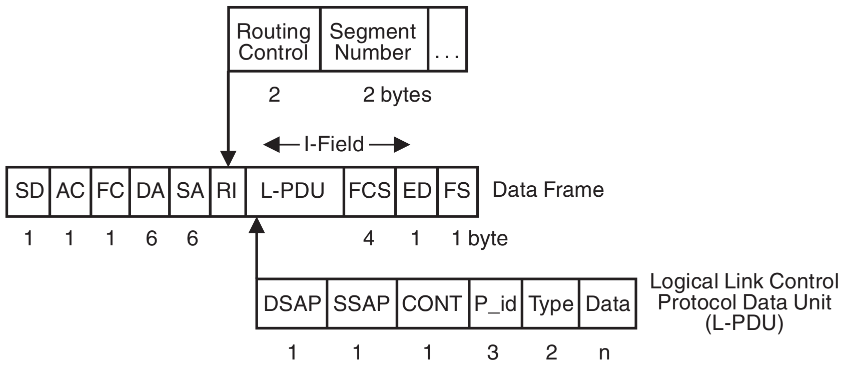


Figure 106. Format of an IEEE 802.5 token-ring frame

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 91 on page 797 shows the relationship between the RC field and the maximum I-field values.

Table 91. Relationship between RC field and maximum I-field value

Routing control field	Maximum I-field in bytes
x000 xxxx xxxx xxxx	516
x001 xxxx xxxx xxxx	1500
x010 xxxx xxxx xxxx	2052
x011 xxxx xxxx xxxx	4472
x100 xxxx xxxx xxxx	8144
x101 xxxx xxxx xxxx	11407
x110 xxxx xxxx xxxx	17800

Figure 106 on page 796 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 798. 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 107.

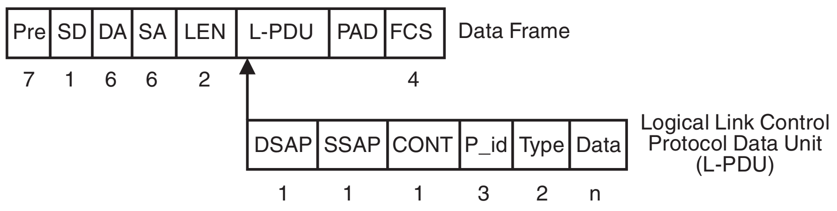


Figure 107. Format of an IEEE 802.3 frame

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 108.

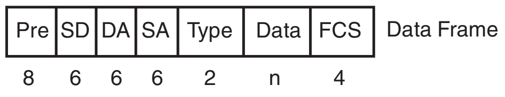


Figure 108. Format of an Ethernet V2 frame

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 109.

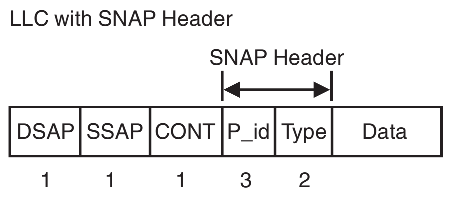


Figure 109. SNAP header

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:

Field	**Value**
DSAP	X'AA'
SSAP	X'AA'
CONT	X'03' Specifies unnumbered information (UI)
P_id	X'00 00 00'
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 110).

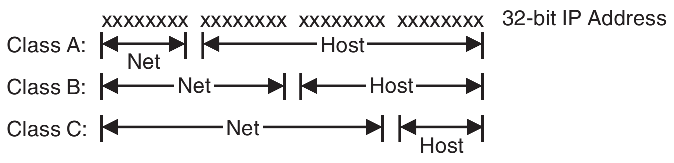


Figure 110. Classes of IPv4 addresses

The class of the address is determined by the first octet of the IPv4 address. Figure 111 on page 800 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.

```
32-bit address          xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx

Class A                 0xxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
              min       00000000
              max       01111111
              range     1 - 126    (decimal notation; 0 and 127 are reserved)

Class B                 10xxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
              min       10000000
              max       10111111
              range     128 - 191  (decimal notation)

Class C                 110xxxxx xxxxxxxx xxxxxxxx xxxxxxxx
              min       11000000
              max       11011111
              range     192 - 223  (decimal notation)

Class D                 1110xxxx xxxxxxxx xxxxxxxx xxxxxxxx
              min       11100000
              max       11101111
              range     224-239 (decimal notation)
```

Figure 111. Determining the class of an IPv4 address

As shown in Figure 111, 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: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, `0:0:0:0:0:0:13.1.68.3` can be expressed in condensed form as `::13.1.68.3`

Figure 112 on page 802 shows a simple network with a bridge and a router.

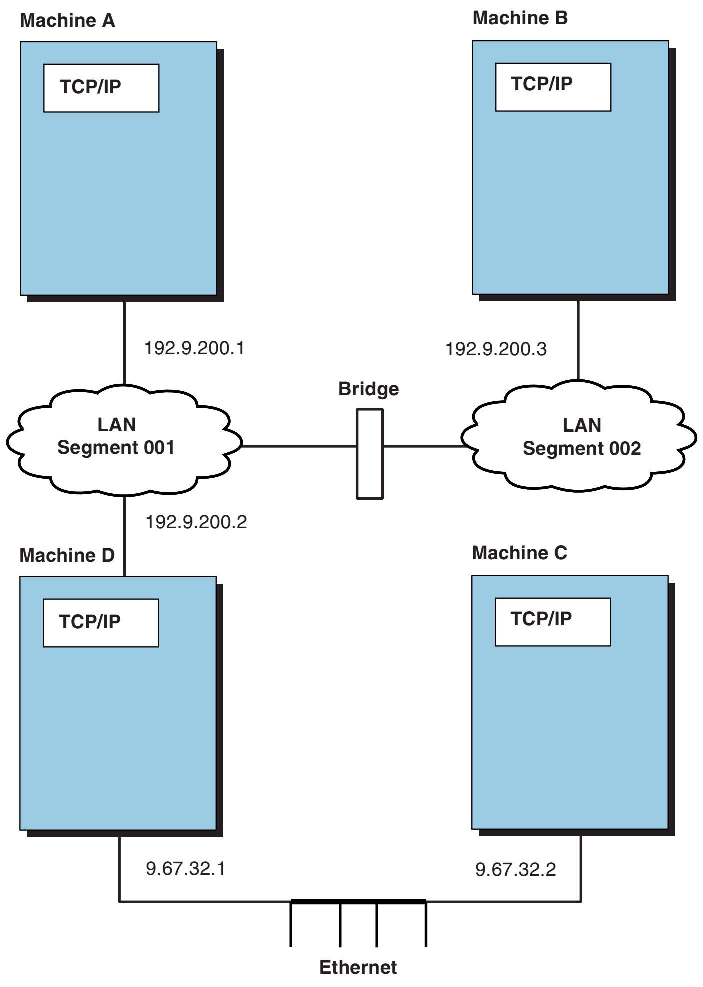


Figure 112. Routing and bridging

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 112 on page 802, 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 112 on page 802, 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 113 is used.

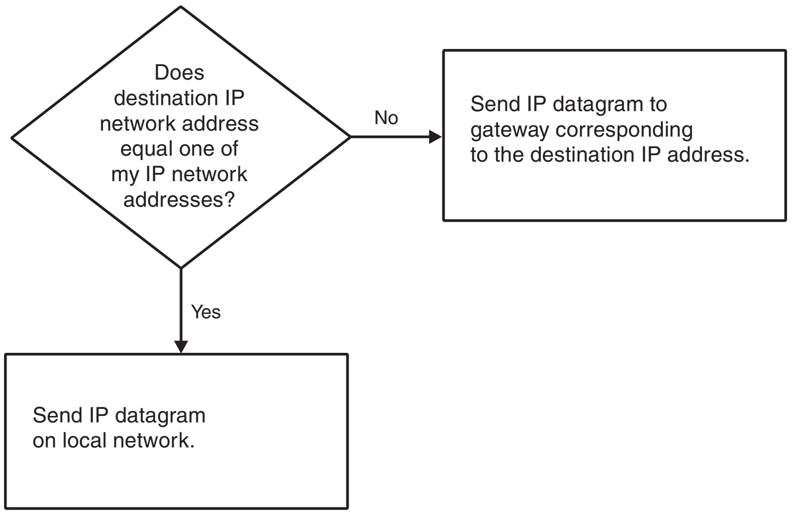

Does destination IP network address equal one of my IP network addresses?
No
Send IP datagram to gateway corresponding to the destination IP address.
Yes
Send IP datagram on local network.

Figure 113. General IP routing algorithm

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 112 on page 802.

```
Machine A   IP Address = 192.9.200.1

Routing Table

Destination     Gateway

192.9.200.1     192.9.200.1   (Machine A's network interface)

9.0.0.0         192.9.200.2   (Route to the 9.n.n.n address is
                               via Machine D, 192.9.200.2)
```

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 114:

```
Without Subnetting:
                                          Network        Host Address
                                          Address        Range
       192      9        200       host
    11000000 00001001 11001000 xxxxxxxx   192.9.200      1 - 254


With Subnetting:
                                          Subnet         Host Address  Subnet
                                          Address        Range         Value
       192      9        200   64  host
    11000000 00001001 11001000 01xxxxxx   192.9.200.64   65 - 126      01


                                          Subnet         Host Address  Subnet
                                          Address        Range         Value
       192      9        200   128 host
    11000000 00001001 11001000 10xxxxxx   192.9.200.128  129 - 190     10

The subnet mask would be

       255      255      255     192
    11111111 11111111 11111111 11000000
```

Figure 114. Subnetting scheme

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 **onetstat -g** command. The subnet mask is applied only to the host segment of the IP address, and onetstat 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 of the chapter, 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 12AB00000000CD3 (hexadecimal):

```
12AB:0000:0000:CD30:0000:0000:0000:0000/60
12AB::CD30:0:0:0:0/60
12AB:0:0: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 115 shows the routing algorithm used with subnets.

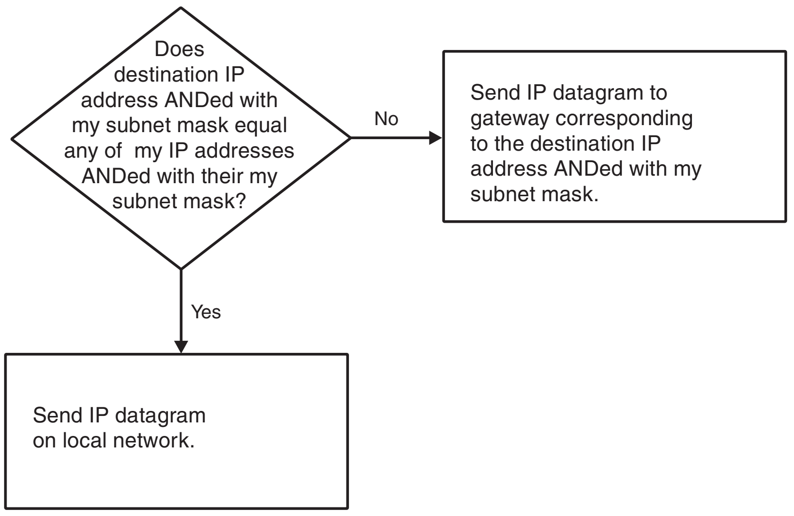

Does destination IP address ANDed with my subnet mask equal any of my IP addresses ANDed with their my subnet mask?
No
Send IP datagram to gateway corresponding to the destination IP address ANDed with my subnet mask.
Yes
Send IP datagram on local network.

Figure 115. Routing algorithm with subnets

Figure 116 on page 807 shows how a subnet route is resolved.

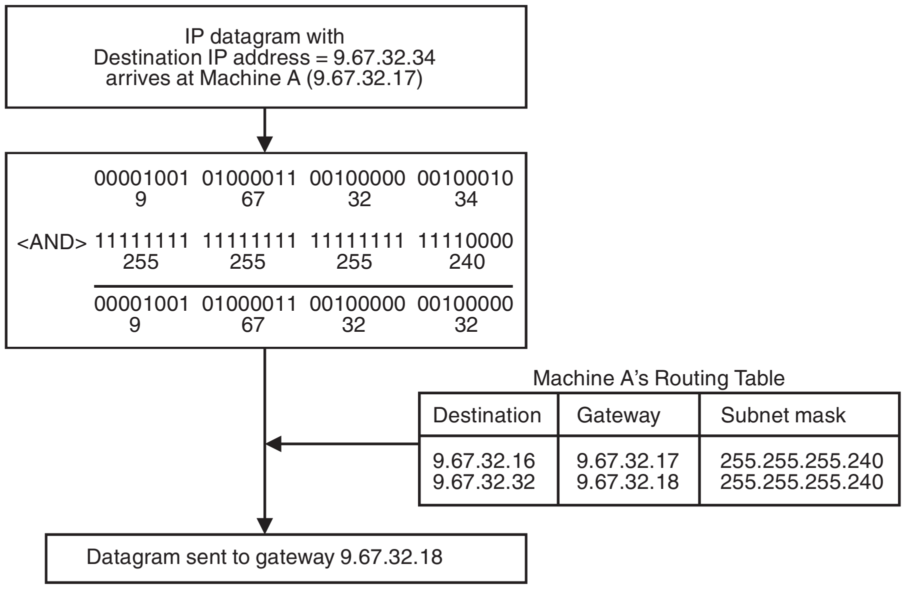


Figure 116. Example of resolving a subnet route

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 statements are defined in the *hlq*.PROFILE.TCPIP data set. The internal routing tables for z/OS TCP/IP can be modified by either:

- Changing the GATEWAY or BEGINROUTES statements and recycling the TCP/IP address space.
- Using the VARY TCPIP,,OBEYFILE command.

Refer to the *z/OS Communications Server: IP System Administrator's Commands* for details about defining the GATEWAY or BEGINROUTES statements.

Tip: When the GATEWAY or BEGINROUTES statements are updated using VARY TCPIP,,OBEYFILE, all previously defined static routes are discarded and replaced by the new GATEWAY or BEGINROUTES definitions.

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 30, “Diagnosing OMPROUTE problems,” on page 665. 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 823
- "Commit-bit support in the IKE daemon" on page 824

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 117.

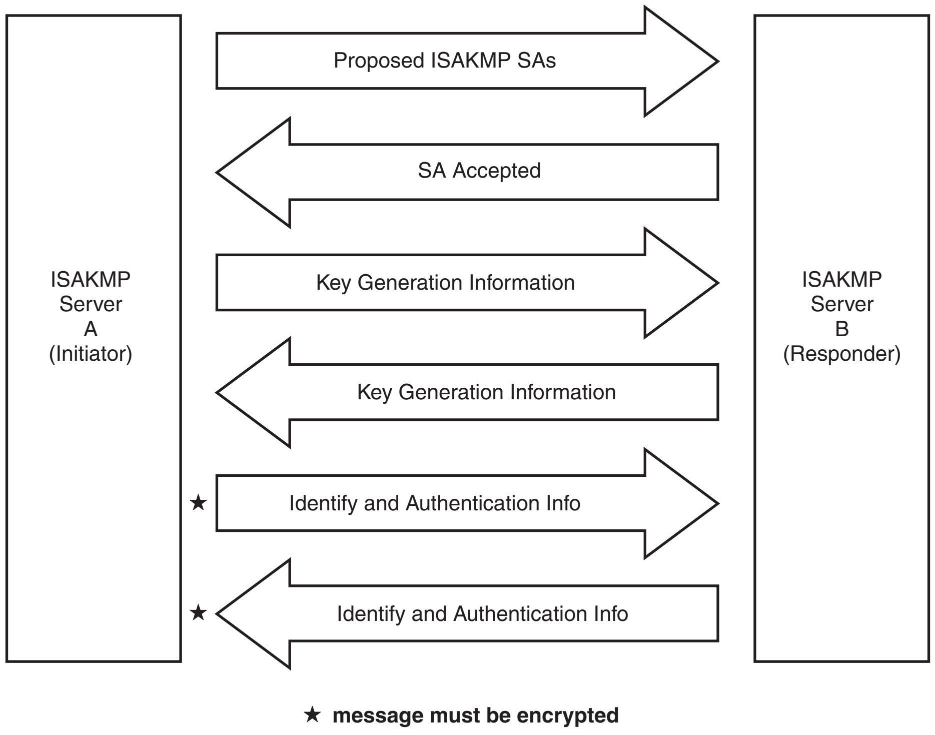

ISAKMP Server A (Initiator)
Proposed ISAKMP SAs
SA Accepted
Key Generation Information
Key Generation Information
★ Identify and Authentication Info
★ Identify and Authentication Info
ISAKMP Server B (Responder)
★ message must be encrypted

Figure 117. 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 phase1:
- Authentication method (for example, pre-shared key or RSA signature)
- Hash algorithm (for example, MD5 or SHA1)
- Encryption algorithm (for example, DES or 3DES)
- Diffie-Hellman group information (for example, group 1 or group 2)
- 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 117 on page 810, 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 118 on page 812.

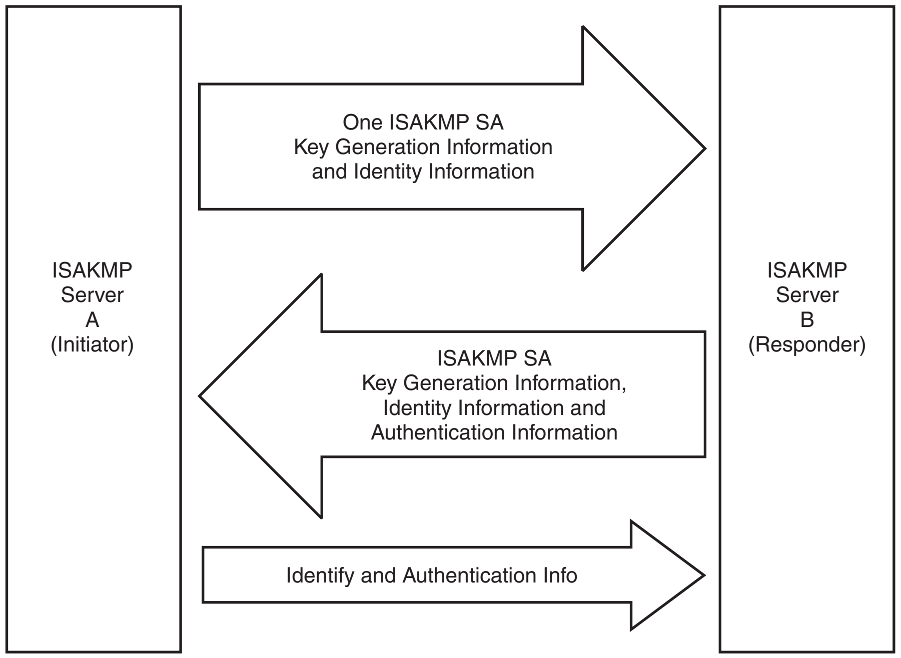


Figure 118. Aggressive mode exchange

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 118 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 119 shows how to interpret phase 1 SA states in Main mode.

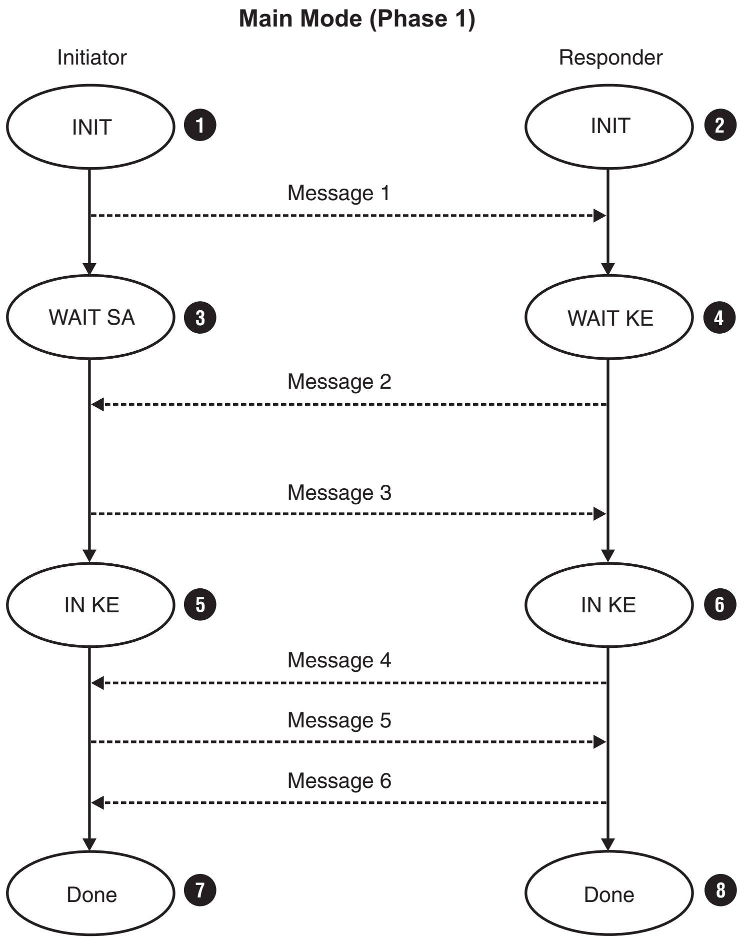


Figure 119. Interpreting phase 1 SA states in Main mode

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 (Figure 119). These states are shown in the state field of the **ipsec -k display** command output. See "Main mode" on page 810 for a description of the contents of the messages. The numbers in the following list correspond to the numbered items in Figure 119.

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 120 shows how to interpret phase 1 SA states in aggressive mode.

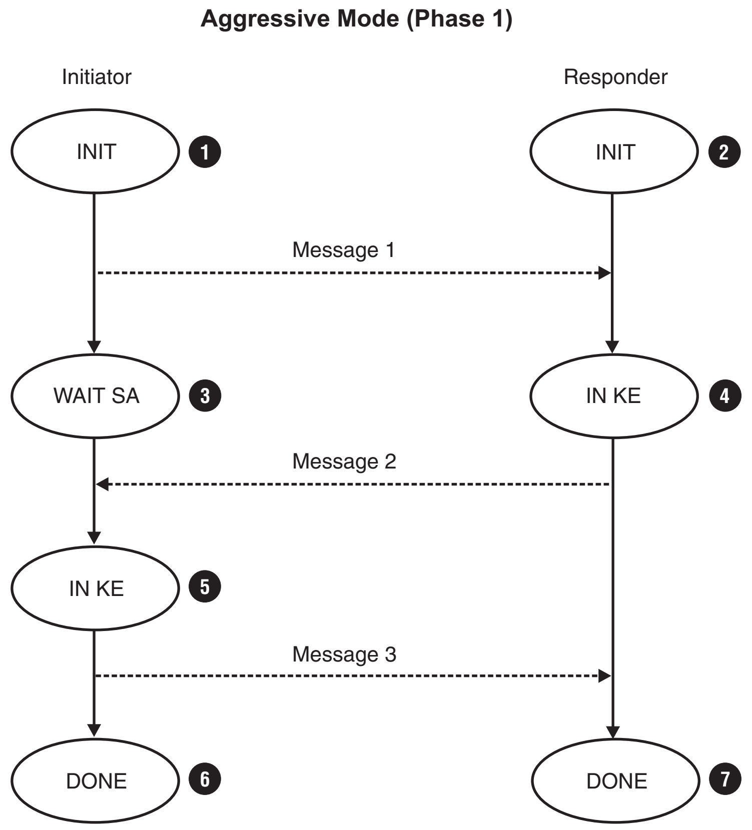


Figure 120. Interpreting phase 1 SA states in Aggressive mode

The following state descriptions apply to the Communications Server IKE daemon when acting as the initiator or responder of an Aggressive mode phase 1 SA negotiation (Figure 120). These states are shown in the state field of the **ipsec -k display** command output. See “Aggressive mode” on page 811 for a description of the contents of the messages. The numbers in the following list correspond to the numbered items inn Figure 120.

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 received from the initiator.
3. The WAIT SA state on the initiator side indicates that the initiator has sent message 1.
4. The IN KE state on the initiator side indicates that the initiator has processed message 1.
5. The IN KE state on the responder side indicates that the responder has received message 2.
6. The DONE state on the initiator side indicates that the initiator has sent message 3.

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 phase1 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 121.

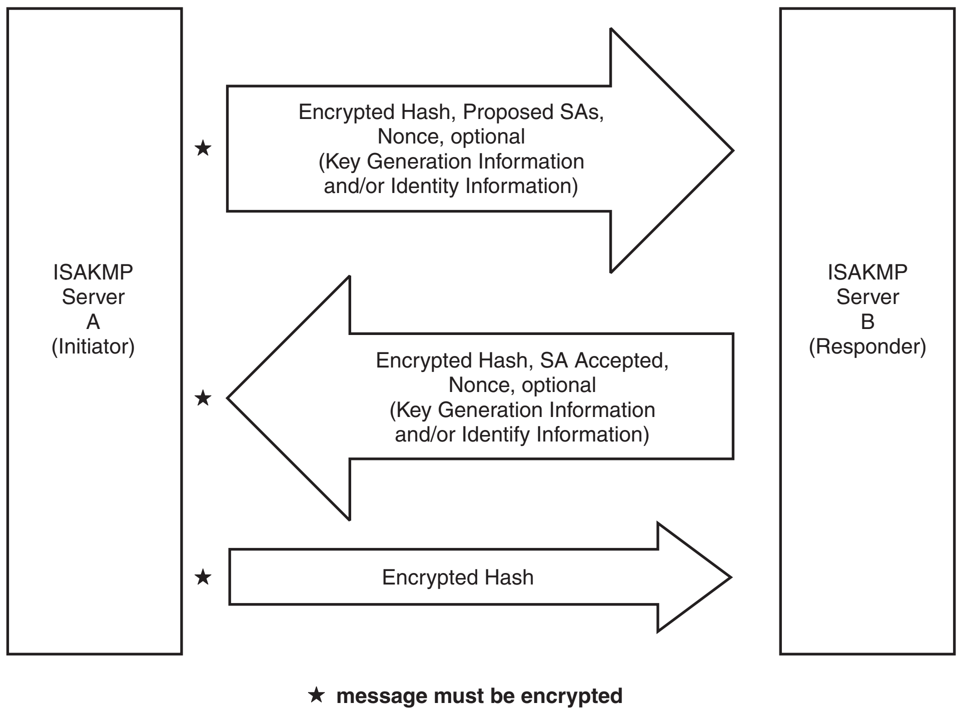

ISAKMP Server A (Initiator)
ISAKMP Server B (Responder)
★ Encrypted Hash, Proposed SAs, Nonce, optional (Key Generation Information and/or Identity Information)
★ Encrypted Hash, SA Accepted, Nonce, optional (Key Generation Information and/or Identify Information)
★ Encrypted Hash
★ message must be encrypted

Figure 121. Quick mode exchange messages

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 or 3DES)
- Diffie-Hellman group information (for example, none, group 1 or group 2)
- 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 122 shows this new message, which includes an encrypted hash along with a notify payload indicating that message 3 was received.

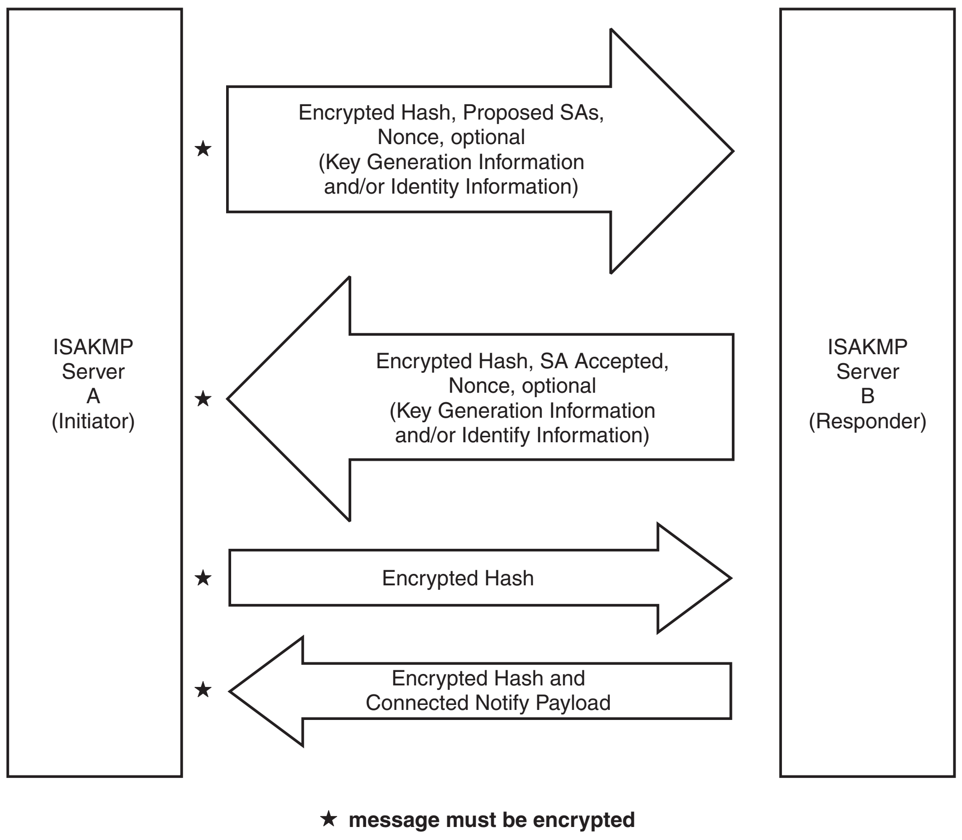

ISAKMP Server A (Initiator)
Encrypted Hash, Proposed SAs, Nonce, optional (Key Generation Information and/or Identity Information)
Encrypted Hash, SA Accepted, Nonce, optional (Key Generation Information and/or Identify Information)
Encrypted Hash
Encrypted Hash and Connected Notify Payload
ISAKMP Server B (Responder)
★ message must be encrypted

Figure 122. 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 123 on page 818 shows interpreting Quick mode (phase 2) SA states without commit-bit support

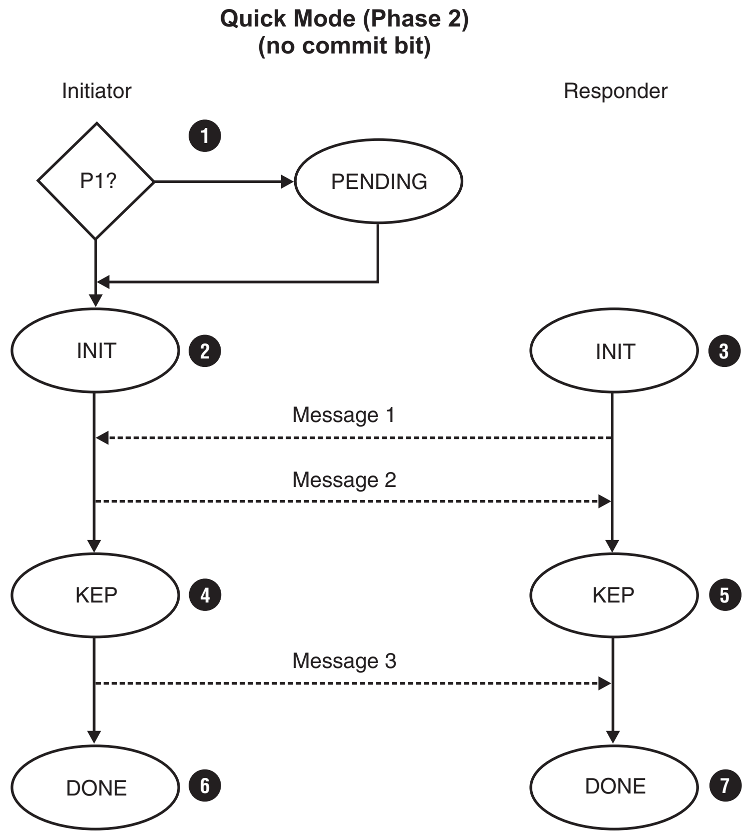


Figure 123. Quick (phase 2) SA states without commit-bit support

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 123). These states are shown in the state field of the **ipsec -y display -b** command output. See “Quick mode” on page 815 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 812. 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. The DONE state on the initiator side indicates that message 3 has been sent.
6. The DONE state on the responder side indicates that message 3 has been received.

Quick mode (phase 2) SA states with commit-bit support: Figure 124 on page 819 shows interpreting Quick mode (phase 2) SA states with commit-bit support.

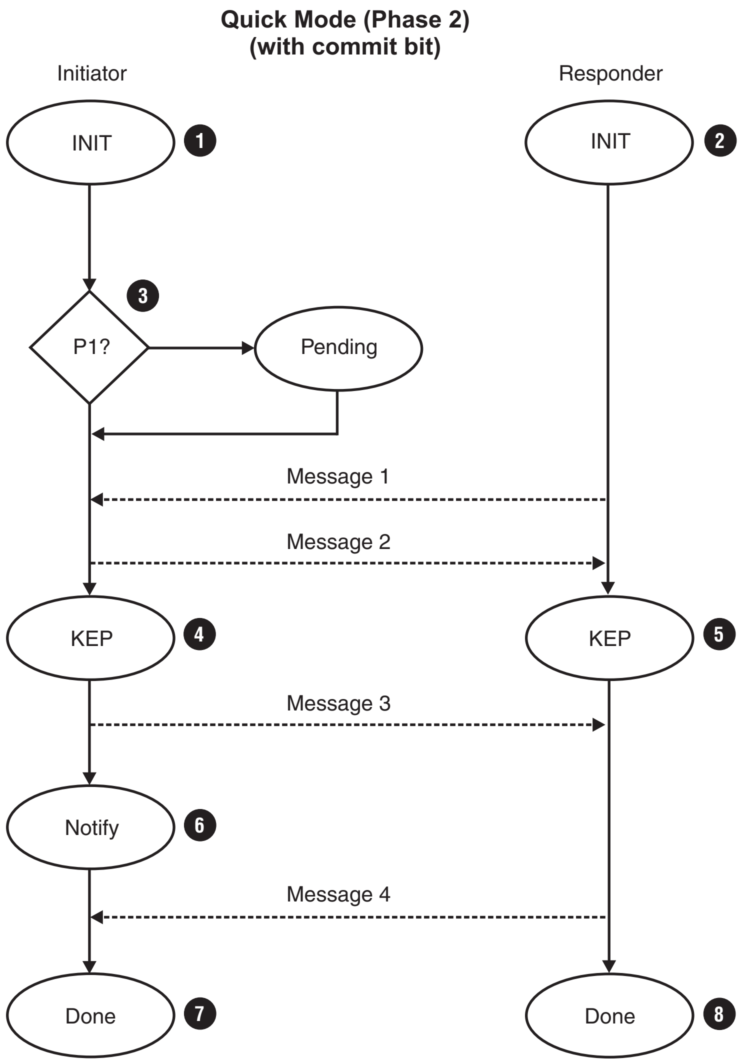


Figure 124. Quick (phase 2) SA states with commit-bit support

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 124). These states are shown in the state field of the **ipsec -y display -b** command output. See “Quick mode with commit bit” on page 816 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 812. 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.

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. 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 92 on page 821 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 92 lists vendor ID strings.

Table 92. Vendor ID

Vendor ID string	Vendor ID
draft-ietf-ipsec-nat-t-ike-02\n	90cb8091 3ebb696e 086381b5 ec427b1f
draft-ietf-ipsec-nat-t-ike-02	cd604643 35df21f8 7cfdb2fc 68b6a448
draft-ietf-ipsec-nat-t-ike-03	7d9419a6 5310ca6f 2c179d92 15529d56
RFC 3947	4a131c81070358455c5728f20e95452f
z/OS CS-IKE NAT Traversal Level 1	95305bb5 64b82a30b 66968bbc 5326a8d

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, all subsequent messages that are exchanged between the initiator and responder must use a source port of UDP 4500 and a destination port of UDP 4500. 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 that is 4 bytes of 0.

Figure 125 shows an IKE packet with and without the non-ESP marker.

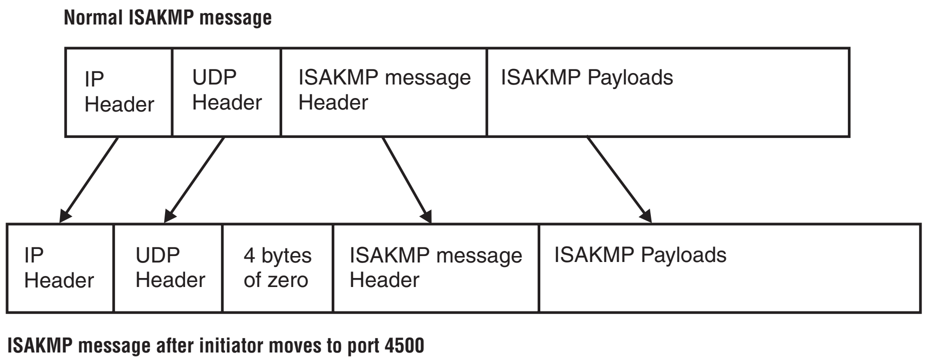


Figure 125. IKE packet with and without the non-ESP marker

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 823.
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 816.

Appendix D. Related protocol specifications (RFCs)

This appendix lists the related protocol specifications 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 838 for draft RFCs implemented in this and previous Communications Server releases.

Many features of TCP/IP Services are based on the following RFCs:

RFC	Title and Author
652	*Telnet output carriage-return disposition option* D. Crocker
653	*Telnet output horizontal tabstops option* D. Crocker
654	*Telnet output horizontal tab disposition option* D. Crocker
655	*Telnet output formfeed disposition option* D. Crocker
657	*Telnet output vertical tab disposition option* D. Crocker
658	*Telnet output linefeed disposition* D. Crocker
698	*Telnet extended ASCII option* T. Mock
726	*Remote Controlled Transmission and Echoing Telnet option* J. Postel, D. Crocker
727	*Telnet logout option* M.R. Crispin
732	*Telnet Data Entry Terminal option* J.D. Day
733	*Standard for the format of ARPA network text messages* D. Crocker, J. Vittal, K.T. Pogran, D.A. Henderson

734	*SUPDUP Protocol* M.R. Crispin
735	*Revised Telnet byte macro option* D. Crocker, R.H. Gumpertz
736	*Telnet SUPDUP option* M.R. Crispin
749	*Telnet SUPDUP—Output option* B. Greenberg
765	*File Transfer Protocol specification* J. Postel
768	*User Datagram Protocol* J. Postel
779	*Telnet send-location option* E. Killian
783	*TFTP Protocol (revision 2)* K.R. Sollins
791	*Internet Protocol* J. Postel
792	*Internet Control Message Protocol* J. Postel
793	*Transmission Control Protocol* J. Postel
820	*Assigned numbers* J. Postel
821	*Simple Mail Transfer Protocol* J. Postel
822	*Standard for the format of ARPA Internet text messages* D. Crocker
823	*DARPA Internet gateway* R. Hinden, A. Sheltzer
826	*Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware* D. Plummer
854	*Telnet Protocol Specification* J. Postel, J. Reynolds
855	*Telnet Option Specification* J. Postel, J. Reynolds
856	*Telnet Binary Transmission* J. Postel, J. Reynolds
857	*Telnet Echo Option* J. Postel, J. Reynolds
858	*Telnet Suppress Go Ahead Option* J. Postel, J. Reynolds
859	*Telnet Status Option* J. Postel, J. Reynolds
860	*Telnet Timing Mark Option* J. Postel, J. Reynolds
861	*Telnet Extended Options: List Option* J. Postel, J. Reynolds
862	*Echo Protocol* J. Postel
863	*Discard Protocol* J. Postel
864	*Character Generator Protocol* J. Postel
865	*Quote of the Day Protocol* J. Postel
868	*Time Protocol* J. Postel, K. Harrenstien
877	*Standard for the transmission of IP datagrams over public data networks* J.T. Korb
883	*Domain names: Implementation specification* P.V. Mockapetris
884	*Telnet terminal type option* M. Solomon, E. Wimmers
885	*Telnet end of record option* J. Postel
894	*Standard for the transmission of IP datagrams over Ethernet networks* C. Hornig
896	*Congestion control in IP/TCP internetworks* J. Nagle

903 *Reverse Address Resolution Protocol* R. Finlayson, T. Mann, J. Mogul, M. Theimer

904 *Exterior Gateway Protocol formal specification* D. Mills

919 *Broadcasting Internet Datagrams* J. Mogul

922 *Broadcasting Internet datagrams in the presence of subnets* J. Mogul

927 *TACACS user identification Telnet option* B.A. Anderson

933 *Output marking Telnet option* S. Silverman

946 *Telnet terminal location number option* R. Nedved

950 *Internet Standard Subnetting Procedure* J. Mogul, J. Postel

951 *Bootstrap Protocol* W.J. Croft, J. Gilmore

952 *DoD Internet host table specification* K. Harrenstien, M. Stahl, E. Feinler

959 *File Transfer Protocol* J. Postel, J.K. Reynolds

961 *Official ARPA-Internet protocols* J.K. Reynolds, J. Postel

974 *Mail routing and the domain system* C. Partridge

1001 *Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods* NetBios Working Group in the Defense Advanced Research Projects Agency, Internet Activities Board, End-to-End Services Task Force

1002 *Protocol Standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications* NetBios Working Group in the Defense Advanced Research Projects Agency, Internet Activities Board, End-to-End Services Task Force

1006 *ISO transport services on top of the TCP: Version 3* M.T. Rose, D.E. Cass

1009 *Requirements for Internet gateways* R. Braden, J. Postel

1011 *Official Internet protocols* J. Reynolds, J. Postel

1013 *X Window System Protocol, version 11: Alpha update April 1987* R. Scheifler

1014 *XDR: External Data Representation standard* Sun Microsystems

1027 *Using ARP to implement transparent subnet gateways* S. Carl-Mitchell, J. Quarterman

1032 *Domain administrators guide* M. Stahl

1033 *Domain administrators operations guide* M. Lottor

1034 *Domain names—concepts and facilities* P.V. Mockapetris

1035 *Domain names—implementation and specification* P.V. Mockapetris

1038 *Draft revised IP security option* M. St. Johns

1041 *Telnet 3270 regime option* Y. Rekhter

1042 *Standard for the transmission of IP datagrams over IEEE 802 networks* J. Postel, J. Reynolds

1043 *Telnet Data Entry Terminal option: DODIIS implementation* A. Yasuda, T. Thompson

1044 *Internet Protocol on Network System's HYPERchannel: Protocol specification* K. Hardwick, J. Lekashman

1053 *Telnet X.3 PAD option* S. Levy, T. Jacobson

1055 *Nonstandard for transmission of IP datagrams over serial lines: SLIP* J. Romkey

1057 *RPC: Remote Procedure Call Protocol Specification: Version 2* Sun Microsystems

1058 *Routing Information Protocol* C. Hedrick

1060 *Assigned numbers* J. Reynolds, J. Postel

1067 *Simple Network Management Protocol* J.D. Case, M. Fedor, M.L. Schoffstall, J. Davin

1071 *Computing the Internet checksum* R.T. Braden, D.A. Borman, C. Partridge

1072 *TCP extensions for long-delay paths* V. Jacobson, R.T. Braden

1073 *Telnet window size option* D. Waitzman

1079 *Telnet terminal speed option* C. Hedrick

1085 *ISO presentation services on top of TCP/IP based internets* M.T. Rose

1091 *Telnet terminal-type option* J. VanBokkelen

1094 *NFS: Network File System Protocol specification* Sun Microsystems

1096 *Telnet X display location option* G. Marcy

1101 *DNS encoding of network names and other types* P. Mockapetris

1112 *Host extensions for IP multicasting* S.E. Deering

1113 *Privacy enhancement for Internet electronic mail: Part I — message encipherment and authentication procedures* J. Linn

1118 *Hitchhikers Guide to the Internet* E. Krol

1122 *Requirements for Internet Hosts—Communication Layers* R. Braden, Ed.

1123 *Requirements for Internet Hosts—Application and Support* R. Braden, Ed.

1146 *TCP alternate checksum options* J. Zweig, C. Partridge

1155 *Structure and identification of management information for TCP/IP-based internets* M. Rose, K. McCloghrie

1156 *Management Information Base for network management of TCP/IP-based internets* K. McCloghrie, M. Rose

1157 *Simple Network Management Protocol (SNMP)* J. Case, M. Fedor, M. Schoffstall, J. Davin

1158 *Management Information Base for network management of TCP/IP-based internets: MIB-II* M. Rose

1166 *Internet numbers* S. Kirkpatrick, M.K. Stahl, M. Recker

1179 *Line printer daemon protocol* L. McLaughlin

1180 *TCP/IP tutorial* T. Socolofsky, C. Kale

1183 *New DNS RR Definitions* C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. Mockapetris

1184 *Telnet Linemode Option* D. Borman

1186 *MD4 Message Digest Algorithm* R.L. Rivest

1187 *Bulk Table Retrieval with the SNMP* M. Rose, K. McCloghrie, J. Davin

1188 *Proposed Standard for the Transmission of IP Datagrams over FDDI Networks* D. Katz

1190 *Experimental Internet Stream Protocol: Version 2 (ST-II)* C. Topolcic

1191 *Path MTU discovery* J. Mogul, S. Deering

1198 *FYI on the X window system* R. Scheifler

1207 *FYI on Questions and Answers: Answers to commonly asked "experienced Internet user" questions* G. Malkin, A. Marine, J. Reynolds

1208 *Glossary of networking terms* O. Jacobsen, D. Lynch

1213 *Management Information Base for Network Management of TCP/IP-based internets: MIB-II* K. McCloghrie, M.T. Rose

1215 *Convention for defining traps for use with the SNMP* M. Rose

1227 *SNMP MUX protocol and MIB* M.T. Rose

1228 *SNMP-DPI: Simple Network Management Protocol Distributed Program Interface* G. Carpenter, B. Wijnen

1229 *Extensions to the generic-interface MIB* K. McCloghrie

1230 *IEEE 802.4 Token Bus MIB* K. McCloghrie, R. Fox

1231 *IEEE 802.5 Token Ring MIB* K. McCloghrie, R. Fox, E. Decker

1236 *IP to X.121 address mapping for DDN* L. Morales, P. Hasse

1256 *ICMP Router Discovery Messages* S. Deering, Ed.

1267 *Border Gateway Protocol 3 (BGP-3)* K. Lougheed, Y. Rekhter

1268 *Application of the Border Gateway Protocol in the Internet* Y. Rekhter, P. Gross

1269 *Definitions of Managed Objects for the Border Gateway Protocol: Version 3* S. Willis, J. Burruss

1270 *SNMP Communications Services* F. Kastenholz, ed.

1285 *FDDI Management Information Base* J. Case

1315 *Management Information Base for Frame Relay DTEs* C. Brown, F. Baker, C. Carvalho

1321 *The MD5 Message-Digest Algorithm* R. Rivest

1323 *TCP Extensions for High Performance* V. Jacobson, R. Braden, D. Borman

1325 *FYI on Questions and Answers: Answers to Commonly Asked "New Internet User" Questions* G. Malkin, A. Marine

1327 *Mapping between X.400 (1988)/ISO 10021 and RFC 822* S. Hardcastle-Kille

1340 *Assigned Numbers* J. Reynolds, J. Postel

1344 *Implications of MIME for Internet Mail Gateways* N. Bornstein

1349 *Type of Service in the Internet Protocol Suite* P. Almquist

1350 *The TFTP Protocol (Revision 2)* K.R. Sollins

1351 *SNMP Administrative Model* J. Davin, J. Galvin, K. McCloghrie

1352 *SNMP Security Protocols* J. Galvin, K. McCloghrie, J. Davin

1353 *Definitions of Managed Objects for Administration of SNMP Parties* K. McCloghrie, J. Davin, J. Galvin

1354 *IP Forwarding Table MIB* F. Baker

1356	*Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode* A. Malis, D. Robinson, R. Ullmann
1358	*Charter of the Internet Architecture Board (IAB)* L. Chapin
1363	*A Proposed Flow Specification* C. Partridge
1368	*Definition of Managed Objects for IEEE 802.3 Repeater Devices* D. McMaster, K. McCloghrie
1372	*Telnet Remote Flow Control Option* C. L. Hedrick, D. Borman
1374	*IP and ARP on HIPPI* J. Renwick, A. Nicholson
1381	*SNMP MIB Extension for X.25 LAPB* D. Throop, F. Baker
1382	*SNMP MIB Extension for the X.25 Packet Layer* D. Throop
1387	*RIP Version 2 Protocol Analysis* G. Malkin
1388	*RIP Version 2 Carrying Additional Information* G. Malkin
1389	*RIP Version 2 MIB Extensions* G. Malkin, F. Baker
1390	*Transmission of IP and ARP over FDDI Networks* D. Katz
1393	*Traceroute Using an IP Option* G. Malkin
1398	*Definitions of Managed Objects for the Ethernet-Like Interface Types* F. Kastenholz
1408	*Telnet Environment Option* D. Borman, Ed.
1413	*Identification Protocol* M. St. Johns
1416	*Telnet Authentication Option* D. Borman, ed.
1420	*SNMP over IPX* S. Bostock
1428	*Transition of Internet Mail from Just-Send-8 to 8bit-SMTP/MIME* G. Vaudreuil
1442	*Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
1443	*Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
1445	*Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Galvin, K. McCloghrie
1447	*Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)* K. McCloghrie, J. Galvin
1448	*Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
1464	*Using the Domain Name System to Store Arbitrary String Attributes* R. Rosenbaum
1469	*IP Multicast over Token-Ring Local Area Networks* T. Pusateri
1483	*Multiprotocol Encapsulation over ATM Adaptation Layer 5* Juha Heinanen
1497	*BOOTP Vendor Information Extensions* J. Reynolds
1514	*Host Resources MIB* P. Grillo, S. Waldbusser
1516	*Definitions of Managed Objects for IEEE 802.3 Repeater Devices* D. McMaster, K. McCloghrie

1521	*MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies* N. Borenstein, N. Freed
1533	*DHCP Options and BOOTP Vendor Extensions* S. Alexander, R. Droms
1534	*Interoperation Between DHCP and BOOTP* R. Droms
1535	*A Security Problem and Proposed Correction With Widely Deployed DNS Software* E. Gavron
1536	*Common DNS Implementation Errors and Suggested Fixes* A. Kumar, J. Postel, C. Neuman, P. Danzig, S. Miller
1537	*Common DNS Data File Configuration Errors* P. Beertema
1540	*Internet Official Protocol Standards* J. Postel
1541	*Dynamic Host Configuration Protocol* R. Droms
1542	*Clarifications and Extensions for the Bootstrap Protocol* W. Wimer
1571	*Telnet Environment Option Interoperability Issues* D. Borman
1572	*Telnet Environment Option* S. Alexander
1573	*Evolution of the Interfaces Group of MIB-II* K. McCloghrie, F. Kastenholz
1577	*Classical IP and ARP over ATM* M. Laubach
1583	*OSPF Version 2* J. Moy
1591	*Domain Name System Structure and Delegation* J. Postel
1592	*Simple Network Management Protocol Distributed Protocol Interface Version 2.0* B. Wijnen, G. Carpenter, K. Curran, A. Sehgal, G. Waters
1594	*FYI on Questions and Answers— Answers to Commonly Asked "New Internet User" Questions* A. Marine, J. Reynolds, G. Malkin
1644	*T/TCP — TCP Extensions for Transactions Functional Specification* R. Braden
1646	*TN3270 Extensions for LUname and Printer Selection* C. Graves, T. Butts, M. Angel
1647	*TN3270 Enhancements* B. Kelly
1652	*SMTP Service Extension for 8bit-MIMEtransport* J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker
1664	*Using the Internet DNS to Distribute RFC1327 Mail Address Mapping Tables* C. Allochio, A. Bonito, B. Cole, S. Giordano, R. Hagens
1693	*An Extension to TCP: Partial Order Service* T. Connolly, P. Amer, P. Conrad
1695	*Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2* M. Ahmed, K. Tesink
1701	*Generic Routing Encapsulation (GRE)* S. Hanks, T. Li, D. Farinacci, P. Traina
1702	*Generic Routing Encapsulation over IPv4 networks* S. Hanks, T. Li, D. Farinacci, P. Traina
1706	*DNS NSAP Resource Records* B. Manning, R. Colella
1712	*DNS Encoding of Geographical Location* C. Farrell, M. Schulze, S. Pleitner D. Baldoni
1713	*Tools for DNS debugging* A. Romao

1723	*RIP Version 2—Carrying Additional Information* G. Malkin
1752	*The Recommendation for the IP Next Generation Protocol* S. Bradner, A. Mankin
1766	*Tags for the Identification of Languages* H. Alvestrand
1771	*A Border Gateway Protocol 4 (BGP-4)* Y. Rekhter, T. Li
1794	*DNS Support for Load Balancing* T. Brisco
1819	*Internet Stream Protocol Version 2 (ST2) Protocol Specification—Version ST2+* L. Delgrossi, L. Berger Eds.
1826	*IP Authentication Header* R. Atkinson
1828	*IP Authentication using Keyed MD5* P. Metzger, W. Simpson
1829	*The ESP DES-CBC Transform* P. Karn, P. Metzger, W. Simpson
1830	*SMTP Service Extensions for Transmission of Large and Binary MIME Messages* G. Vaudreuil
1832	*XDR: External Data Representation Standard* R. Srinivasan
1850	*OSPF Version 2 Management Information Base* F. Baker, R. Coltun
1854	*SMTP Service Extension for Command Pipelining* N. Freed
1869	*SMTP Service Extensions* J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker
1870	*SMTP Service Extension for Message Size Declaration* J. Klensin, N. Freed, K. Moore
1876	*A Means for Expressing Location Information in the Domain Name System* C. Davis, P. Vixie, T. Goodwin, I. Dickinson
1883	*Internet Protocol, Version 6 (IPv6) Specification* S. Deering, R. Hinden
1884	*IP Version 6 Addressing Architecture* R. Hinden, S. Deering, Eds.
1886	*DNS Extensions to support IP version 6* S. Thomson, C. Huitema
1888	*OSI NSAPs and IPv6* J. Bound, B. Carpenter, D. Harrington, J. Houldsworth, A. Lloyd
1891	*SMTP Service Extension for Delivery Status Notifications* K. Moore
1892	*The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages* G. Vaudreuil
1894	*An Extensible Message Format for Delivery Status Notifications*K. Moore, G. Vaudreuil
1901	*Introduction to Community-based SNMPv2* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
1902	*Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
1903	*Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
1904	*Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser
1905	*Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1906 *Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1907 *Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1908 *Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework* J. Case, K. McCloghrie, M. Rose, S. Waldbusser

1912 *Common DNS Operational and Configuration Errors* D. Barr

1918 *Address Allocation for Private Internets* Y. Rekhter, B. Moskowitz, D. Karrenberg, G.J. de Groot, E. Lear

1928 *SOCKS Protocol Version 5* M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones

1930 *Guidelines for creation, selection, and registration of an Autonomous System (AS)* J. Hawkinson, T. Bates

1939 *Post Office Protocol-Version 3* J. Myers, M. Rose

1981 *Path MTU Discovery for IP version 6* J. McCann, S. Deering, J. Mogul

1982 *Serial Number Arithmetic* R. Elz, R. Bush

1985 *SMTP Service Extension for Remote Message Queue Starting* J. De Winter

1995 *Incremental Zone Transfer in DNS* M. Ohta

1996 *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)* P. Vixie

2010 *Operational Criteria for Root Name Servers* B. Manning, P. Vixie

2011 *SNMPv2 Management Information Base for the Internet Protocol using SMIv2* K. McCloghrie, Ed.

2012 *SNMPv2 Management Information Base for the Transmission Control Protocol using SMIv2* K. McCloghrie, Ed.

2013 *SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2* K. McCloghrie, Ed.

2018 *TCP Selective Acknowledgement Options* M. Mathis, J. Mahdavi, S. Floyd, A. Romanow

2026 *The Internet Standards Process — Revision 3* S. Bradner

2030 *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI* D. Mills

2033 *Local Mail Transfer Protocol* J. Myers

2034 *SMTP Service Extension for Returning Enhanced Error Codes*N. Freed

2040 *The RC5, RC5–CBC, RC-5–CBC-Pad, and RC5–CTS Algorithms*R. Baldwin, R. Rivest

2045 *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies* N. Freed, N. Borenstein

2052 *A DNS RR for specifying the location of services (DNS SRV)* A. Gulbrandsen, P. Vixie

2065 *Domain Name System Security Extensions* D. Eastlake 3rd, C. Kaufman

2066 *TELNET CHARSET Option* R. Gellens

2080 *RIPng for IPv6* G. Malkin, R. Minnear

2096 *IP Forwarding Table MIB* F. Baker

2104 *HMAC: Keyed-Hashing for Message Authentication* H. Krawczyk, M. Bellare, R. Canetti

2119 *Keywords for use in RFCs to Indicate Requirement Levels* S. Bradner

2132 *DHCP Options and BOOTP Vendor Extensions* S. Alexander, R. Droms

2133 *Basic Socket Interface Extensions for IPv6* R. Gilligan, S. Thomson, J. Bound, W. Stevens

2136 *Dynamic Updates in the Domain Name System (DNS UPDATE)* P. Vixie, Ed., S. Thomson, Y. Rekhter, J. Bound

2137 *Secure Domain Name System Dynamic Update* D. Eastlake 3rd

2163 *Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM)* C. Allocchio

2168 *Resolution of Uniform Resource Identifiers using the Domain Name System* R. Daniel, M. Mealling

2178 *OSPF Version 2* J. Moy

2181 *Clarifications to the DNS Specification* R. Elz, R. Bush

2205 *Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification* R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin

2210 *The Use of RSVP with IETF Integrated Services* J. Wroclawski

2211 *Specification of the Controlled-Load Network Element Service* J. Wroclawski

2212 *Specification of Guaranteed Quality of Service* S. Shenker, C. Partridge, R. Guerin

2215 *General Characterization Parameters for Integrated Service Network Elements* S. Shenker, J. Wroclawski

2217 *Telnet Com Port Control Option* G. Clarke

2219 *Use of DNS Aliases for Network Services* M. Hamilton, R. Wright

2228 *FTP Security Extensions* M. Horowitz, S. Lunt

2230 *Key Exchange Delegation Record for the DNS* R. Atkinson

2233 *The Interfaces Group MIB using SMIv2* K. McCloghrie, F. Kastenholz

2240 *A Legal Basis for Domain Name Allocation* O. Vaughn

2246 *The TLS Protocol Version 1.0* T. Dierks, C. Allen

2251 *Lightweight Directory Access Protocol (v3)* M. Wahl, T. Howes, S. Kille

2253 *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names* M. Wahl, S. Kille, T. Howes

2254 *The String Representation of LDAP Search Filters* T. Howes

2261 *An Architecture for Describing SNMP Management Frameworks* D. Harrington, R. Presuhn, B. Wijnen

2262 *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)* J. Case, D. Harrington, R. Presuhn, B. Wijnen

2271 *An Architecture for Describing SNMP Management Frameworks* D. Harrington, R. Presuhn, B. Wijnen

2273 *SNMPv3 Applications* D. Levi, P. Meyer, B. Stewartz

2274 *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)* U. Blumenthal, B. Wijnen

2275 *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)* B. Wijnen, R. Presuhn, K. McCloghrie

2292 *Advanced Sockets API for IPv6* W. Stevens, M. Thomas

2308 *Negative Caching of DNS Queries (DNS NCACHE)* M. Andrews

2317 *Classless IN-ADDR.ARPA delegation* H. Eidnes, G. de Groot, P. Vixie

2320 *Definitions of Managed Objects for Classical IP and ARP Over ATM Using SMIv2 (IPOA-MIB)* M. Greene, J. Luciani, K. White, T. Kuo

2328 *OSPF Version 2* J. Moy

2345 *Domain Names and Company Name Retrieval* J. Klensin, T. Wolf, G. Oglesby

2352 *A Convention for Using Legal Names as Domain Names* O. Vaughn

2355 *TN3270 Enhancements* B. Kelly

2358 *Definitions of Managed Objects for the Ethernet-like Interface Types* J. Flick, J. Johnson

2373 *IP Version 6 Addressing Architecture* R. Hinden, S. Deering

2374 *An IPv6 Aggregatable Global Unicast Address Format* R. Hinden, M. O'Dell, S. Deering

2375 *IPv6 Multicast Address Assignments* R. Hinden, S. Deering

2385 *Protection of BGP Sessions via the TCP MD5 Signature Option*A. Hefferman

2389 *Feature negotiation mechanism for the File Transfer Protocol* P. Hethmon, R. Elz

2401 *Security Architecture for Internet Protocol* S. Kent, R. Atkinson

2402 *IP Authentication Header* S. Kent, R. Atkinson

2403 *The Use of HMAC-MD5–96 within ESP and AH* C. Madson, R. Glenn

2404 *The Use of HMAC-SHA–1–96 within ESP and AH* C. Madson, R. Glenn

2405 *The ESP DES-CBC Cipher Algorithm With Explicit IV* C. Madson, N. Doraswamy

2406 *IP Encapsulating Security Payload (ESP)* S. Kent, R. Atkinson

2407 *The Internet IP Security Domain of Interpretation for ISAKMP*D. Piper

2408 *Internet Security Association and Key Management Protocol (ISAKMP)* D. Maughan, M. Schertler, M. Schneider, J. Turner

2409 *The Internet Key Exchange (IKE)* D. Harkins, D. Carrel

2410 *The NULL Encryption Algorithm and Its Use With IPsec* R. Glenn, S. Kent,

2428 *FTP Extensions for IPv6 and NATs* M. Allman, S. Ostermann, C. Metz

2445 *Internet Calendaring and Scheduling Core Object Specification (iCalendar)* F. Dawson, D. Stenerson

2459 *Internet X.509 Public Key Infrastructure Certificate and CRL Profile* R. Housley, W. Ford, W. Polk, D. Solo

2460 *Internet Protocol, Version 6 (IPv6) Specification* S. Deering, R. Hinden

2461 *Neighbor Discovery for IP Version 6 (IPv6)* T. Narten, E. Nordmark, W. Simpson

2462 *IPv6 Stateless Address Autoconfiguration* S. Thomson, T. Narten

| **2463** *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification* A. Conta, S. Deering

2464 *Transmission of IPv6 Packets over Ethernet Networks* M. Crawford

| **2466** *Management Information Base for IP Version 6: ICMPv6 Group* D. Haskin, S. Onishi

| **2476** *Message Submission* R. Gellens, J. Klensin

2487 *SMTP Service Extension for Secure SMTP over TLS* P. Hoffman

2505 *Anti-Spam Recommendations for SMTP MTAs* G. Lindberg

| **2523** *Photuris: Extended Schemes and Attributes* P. Karn, W. Simpson

2535 *Domain Name System Security Extensions* D. Eastlake 3rd

| **2538** *Storing Certificates in the Domain Name System (DNS)* D. Eastlake 3rd, O. Gudmundsson

2539 *Storage of Diffie-Hellman Keys in the Domain Name System (DNS)* D. Eastlake 3rd

| **2540** *Detached Domain Name System (DNS) Information* D. Eastlake 3rd

| **2554** *SMTP Service Extension for Authentication* J. Myers

2570 *Introduction to Version 3 of the Internet-standard Network Management Framework* J. Case, R. Mundy, D. Partain, B. Stewart

2571 *An Architecture for Describing SNMP Management Frameworks* B. Wijnen, D. Harrington, R. Presuhn

2572 *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)* J. Case, D. Harrington, R. Presuhn, B. Wijnen

2573 *SNMP Applications* D. Levi, P. Meyer, B. Stewart

2574 *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)* U. Blumenthal, B. Wijnen

2575 *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)* B. Wijnen, R. Presuhn, K. McCloghrie

2576 *Co-Existence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework* R. Frye, D. Levi, S. Routhier, B. Wijnen

2578 *Structure of Management Information Version 2 (SMIv2)* K. McCloghrie, D. Perkins, J. Schoenwaelder

| **2579** *Textual Conventions for SMIv2* K. McCloghrie, D. Perkins, J. Schoenwaelder

| **2580** *Conformance Statements for SMIv2* K. McCloghrie, D. Perkins, J. Schoenwaelder

| **2581** *TCP Congestion Control* M. Allman, V. Paxson, W. Stevens

| **2583** *Guidelines for Next Hop Client (NHC) Developers* R. Carlson, L. Winkler

| **2591** *Definitions of Managed Objects for Scheduling Management Operations* D. Levi, J. Schoenwaelder

| **2625** *IP and ARP over Fibre Channel* M. Rajagopal, R. Bhagwat, W. Rickard

2635	*Don't SPEW A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam*)* S. Hambridge, A. Lunde
2637	*Point-to-Point Tunneling Protocol* K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, G. Zorn
2640	*Internationalization of the File Transfer Protocol* B. Curtin
2665	*Definitions of Managed Objects for the Ethernet-like Interface Types* J. Flick, J. Johnson
2671	*Extension Mechanisms for DNS (EDNS0)* P. Vixie
2672	*Non-Terminal DNS Name Redirection* M. Crawford
2675	*IPv6 Jumbograms* D. Borman, S. Deering, R. Hinden
2710	*Multicast Listener Discovery (MLD) for IPv6* S. Deering, W. Fenner, B. Haberman
2711	*IPv6 Router Alert Option* C. Partridge, A. Jackson
2740	*OSPF for IPv6* R. Coltun, D. Ferguson, J. Moy
2753	*A Framework for Policy-based Admission Control* R. Yavatkar, D. Pendarakis, R. Guerin
2758	*Definitions of Managed Objects for Service Level Agreements Performance Monitoring* K. White
2782	*A DNS RR for specifying the location of services (DNS SRV)* A. Gubrandsen, P. Vixix, L. Esibov
2821	*Simple Mail Transfer Protocol* J. Klensin, Ed.
2822	*Internet Message Format* P. Resnick, Ed.
2840	*TELNET KERMIT OPTION* J. Altman, F. da Cruz
2845	*Secret Key Transaction Authentication for DNS (TSIG)* P. Vixie, O. Gudmundsson, D. Eastlake 3rd, B. Wellington
2851	*Textual Conventions for Internet Network Addresses* M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder
2852	*Deliver By SMTP Service Extension* D. Newman
2874	*DNS Extensions to Support IPv6 Address Aggregation and Renumbering* M. Crawford, C. Huitema
2915	*The Naming Authority Pointer (NAPTR) DNS Resource Record* M. Mealling, R. Daniel
2920	*SMTP Service Extension for Command Pipelining* N. Freed
2930	*Secret Key Establishment for DNS (TKEY RR)* D. Eastlake, 3rd
2941	*Telnet Authentication Option* T. Ts'o, ed., J. Altman
2942	*Telnet Authentication: Kerberos Version 5* T. Ts'o
2946	*Telnet Data Encryption Option* T. Ts'o
2952	*Telnet Encryption: DES 64 bit Cipher Feedback* T. Ts'o
2953	*Telnet Encryption: DES 64 bit Output Feedback* T. Ts'o
2992	*Analysis of an Equal-Cost Multi-Path Algorithm* C. Hopps

3019	*IP Version 6 Management Information Base for The Multicast Listener Discovery Protocol* B. Haberman, R. Worzella
3060	*Policy Core Information Model—Version 1 Specification* B. Moore, E. Ellesson, J. Strassner, A. Westerinen
3152	*Delegation of IP6.ARPA* R. Bush
3291	*Textual Conventions for Internet Network Addresses* M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder
3363	*Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System* R. Bush, A. Durand, B. Fink, O. Gudmundsson, T. Hain
3390	*Increasing TCP's Initial Window* M. Allman, S. Floyd, C. Partridge
3411	*An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks* D. Harrington, R. Presuhn, B. Wijnen
3412	*Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)* J. Case, D. Harrington, R. Presuhn, B. Wijnen
3413	*Simple Network Management Protocol (SNMP) Applications* D. Levi, P. Meyer, B. Stewart
3414	*User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)* U. Blumenthal, B. Wijnen
3415	*View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)* B. Wijnen, R. Presuhn, K. McCloghrie
3419	*Textual Conventions for Transport Addresses* M. Daniele, J. Schoenwaelder
3484	*Default Address Selection for Internet Protocol version 6 (IPv6)* R. Draves
3493	*Basic Socket Interface Extensions for IPv6* R. Gilligan, S. Thomson, J. Bound, J. McCann, W. Stevens
3513	*Internet Protocol Version 6 (IPv6) Addressing Architecture* R. Hinden, S. Deering
3542	*Advanced Sockets Application Programming Interface (API) for IPv6* W. Richard Stevens, M. Thomas, E. Nordmark, T. Jinmei
3658	*Delegation Signer (DS) Resource Record (RR)* O. Gudmundsson
3715	*IPsec-Network Address Translation (NAT) Compatibility Requirements* B. Aboba, W. Dixon
3947	*Negotiation of NAT-Traversal in the IKE* T. Kivinen, B. Swander, A. Huttunen, V. Volpe
3948	*UDP Encapsulation of IPsec ESP Packets* A. Huttunen, B. Swander, V. Volpe, L. DiBurro, M. Stenberg

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.

Several areas of IPv6 implementation include elements of the following Internet drafts and are subject to change during the RFC review process.

Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification

A. Conta, S. Deering

Appendix E. Information APARs

This appendix lists information APARs for IP and SNA documents.

Notes:

1. Information APARs contain updates to previous editions of the manuals listed below. Documents updated for V1R7 are complete except for the updates contained in the information APARs that might be issued after V1R7 documents went to press.
2. Information APARs are predefined for z/OS V1R7 Communications Server and might not contain updates.
3. Information APARs for z/OS documents are in the document called *z/OS and z/OS.e DOC APAR and PTF ++HOLD Documentation*, which can be found at http://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS.

Information APARs for IP documents

Table 93 lists information APARs for IP documents. For information APARs for V1R7, see http://www.ibm.com/support/docview.wss?uid=swg21178966.

Table 93. IP information APARs for z/OS Communications Server

Title	V1R6	V1R5	V1R4
New Function Summary (both IP and SNA)	II13824		
Quick Reference (both IP and SNA)	II13831		II13246
IP and SNA Codes	II13842		II13254
IP API Guide	II13844	II13577	II13255 II13790
IP CICS Sockets Guide		II13578	II13257
IP Configuration Guide	II13826	II13568	II13244 II13541 II13652 II13646
IP Configuration Reference	II13827	II13569 II13789	II13245 II13521 II13647 II13739
IP Diagnosis	II13836	II13571	II13249 II13493
IP Messages Volume 1	II13838	II13572	II13624 II13250
IP Messages Volume 2	II13839	II13573	II13251
IP Messages Volume 3	II13840	II13574	II13252
IP Messages Volume 4	II13841	II13575	II13253 II13628
IP Migration		II13566	II13242 II13738
IP Network and Application Design Guide	II13825	II13567	II13243

Table 93. IP information APARs for z/OS Communications Server (continued)

Title	V1R6	V1R5	V1R4
IP Network Print Facility			
IP Programmer's Reference	II13843	II13581	II13256
IP User's Guide and Commands	II13832	II13570	II13247
IP System Admin Commands	II13833	II13580	II13248 II13792

Information APARs for SNA documents

Table 94 lists information APARs for SNA documents. For information APARs for V1R7, see http://www.ibm.com/support/docview.wss?uid=swg21178966.

Table 94. SNA information APARs for z/OS Communications Server

Title	V1R6	V1R5	V1R4
New Function Summary (both IP and SNA)	II13824		
Quick Reference (both IP and SNA)	II13831		II13246
IP and SNA Codes	II13842		II13254
SNA Customization	II13857	II13560	II13240
SNA Diagnosis		II13558	II13236 II13735
SNA Diagnosis, Vol. 1: Techniques and Procedures	II13852		
SNA Diagnosis, Vol. 2: FFST Dumps and the VIT	II13853		
SNA Messages	II13854	II13559	II13238 II13736
SNA Network Implementation Guide	II13849	II13555	II13234 II13733
SNA Operation	II13851	II13557	II13237
SNA Migration		II13554	II13233 II13732
SNA Programming	II13858		II13241
SNA Resource Definition Reference	II13850	II13556	II13235 II13734
SNA Data Areas, Vol. 1 and 2			II13239
SNA Data Areas, 1	II13855		
SNA Data Areas, 2	II13856		

Other information APARs

Table 95 lists information APARs not related to documents.

Table 95. Non-document information APARs

Content	Number
Index to APARs that list recommended VTAM maintenance	II11220

Table 95. Non-document information APARs (continued)

Content	Number
Index to APARs that list trace and dump requests for VTAM problems	II13202
Index of Communication Server IP information APARs	II12028
MPC and CTC	II01501
Collecting TCPIP CTRACEs	II12014
CSM for VTAM	II13442
CSM for TCP/IP	II13951
DLUR/DLUS for z/OS V1R2, V1R4, and V1R5	II12986, II13456, and II13783
DOCUMENTATION REQUIRED FOR OSA/2, OSA EXPRESS AND OSA QDIO	II13016
DYNAMIC VIPA (BIND)	II13215
DNS — common problems and solutions	II13453
Enterprise Extender	II12223
FTPing doc to z/OS Support	II12030
FTP problems	II12079
Generic resources	II10986
HPR	II10953
iQDIO	II13142
LPR problems	II12022
MNPS	II10370
NCPROUTE problems	II12025
OMPROUTE	II12026
PASCAL API	II11814
Performance	II11710 II11711 II11712
Resolver	II13398 II13399 II13452
Socket API	II11996 II12020
SMTP problems	II12023
SNMP	II13477 II13478
SYSLOGD howto	II12021
TCPIP connection states	II12449
Telnet	II11574 II13135
TN3270 TELNET SSL common problems	II13369

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/servers/eserver/zseries/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.

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 of unlicensed books and the z/OS Licensed Product Library (LK3T-4307), which contains BookManager and PDF formats of licensed books.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

Advanced Peer-to-Peer Networking
AFP
AD/Cycle
AIX
AIX/ESA
AnyNet
APL2
AS/400
AT
BookManager
BookMaster
C/370
CICS
CICS/ESA
C/MVS
Common User Access
C Set ++
CT
CUA
DB2
DFSMSdfp
DFSMShsm
DFSMS/MVS
DPI
Domino
DRDA
Enterprise Systems Architecture/370
ESCON
eServer
ES/3090
ES/9000
ES/9370
EtherStreamer
Extended Services
FFST
FFST/2
First Failure Support Technology
GDDM
IBM
IBMLink
IMS
IMS/ESA
HiperSockets
Language Environment
LANStreamer
Library Reader
LPDA
Micro Channel
Multiprise
MVS
MVS/DFP
MVS/ESA
MVS/SP
MVS/XA
NetView
Network Station
Nways
Notes
OfficeVision/MVS
OfficeVision/VM
Open Class
OS/2
OS/390
OS/400
Parallel Sysplex
PR/SM
PROFS
PS/2
RACF
Redbooks
Resource Link
RETAIN
RISC System/6000
RMF
RS/6000
S/370
S/390
S/390 Parallel Enterprise Server
SAA
SecureWay
SP
SP2
SQL/DS
System/360
System/370
System/390
SystemView
Tivoli
TURBOWAYS
VM/ESA
VSE/ESA
VTAM
WebSphere
XT
z/Architecture
z/OS
zSeries
z/VM
400
3090
3890

DB2 and NetView are registered trademarks of International Business Machines Corporation or Tivoli Systems Inc. in the U.S., other countries, or both.

Bibliography

z/OS Communications Server information

This section contains descriptions of the documents in the z/OS Communications Server library.

z/OS Communications Server documentation is available:

- Online at the z/OS Internet Library web page at http://www.ibm.com/servers/eserver/zseries/zos/bkserv
- In softcopy on CD-ROM collections. See "Softcopy information" on page xxvi.

z/OS Communications Server library

z/OS Communications Server documents are available on the CD-ROM accompanying z/OS (SK3T-4269 or SK3T-4307). Unlicensed documents can be viewed at the z/OS Internet library site.

Updates to documents are available on RETAIN and in information APARs (info APARs). See Appendix E, "Information APARs," on page 841 for a list of the documents and the info APARs associated with them.

Info APARs for z/OS documents are in the document called *z/OS and z/OS.e DOC APAR and PTF ++HOLD Documentation* which can be found at http://publibz.boulder.ibm.com:80/cgi-bin/bookmgr_OS390/ BOOKS/ZIDOCMST/CCONTENTS.

Planning

Title	Number	Description
z/OS Communications Server: New Function Summary	GC31-8771	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.
z/OS Communications Server: IPv6 Network and Application Design Guide	SC31-8885	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.

Resource definition, configuration, and tuning

Title	Number	Description
z/OS Communications Server: IP Configuration Guide	SC31-8775	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*.

Title	Number	Description
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: AnyNet SNA over TCP/IP	SC31-8832	This guide provides information to help you install, configure, use, and diagnose SNA over TCP/IP.
z/OS Communications Server: AnyNet Sockets over SNA	SC31-8831	This guide provides information to help you install, configure, use, and diagnose sockets over SNA. It also provides information to help you prepare application programs to use sockets over SNA.
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

Title	Number	Description
z/OS Communications Server: IP User's Guide and Commands	SC31-8780	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.
z/OS Communications Server: IP System Administrator's Commands	SC31-8781	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.
z/OS Communications Server: SNA Operation	SC31-8779	This document serves as a reference for programmers and operators requiring detailed information about specific operator commands.
z/OS Communications Server: Quick Reference	SX75-0124	This document contains essential information about SNA and IP commands.

Customization

Title	Number	Description
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

Title	Number	Description
z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference	SC31-8788	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.
z/OS Communications Server: IP CICS Sockets Guide	SC31-8807	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.
z/OS Communications Server: IP IMS Sockets Guide	SC31-8830	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.
z/OS Communications Server: IP Programmer's Guide and Reference	SC31-8787	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.
z/OS Communications Server: SNA Programming	SC31-8829	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.
z/OS Communications Server: SNA Programmer's LU 6.2 Guide	SC31-8811	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.)
z/OS Communications Server: SNA Programmer's LU 6.2 Reference	SC31-8810	This document provides reference material for the SNA LU 6.2 programming interface for host application programs.
z/OS Communications Server: CSM Guide	SC31-8808	This document describes how applications use the communications storage manager.

Title	Number	Description
z/OS Communications Server: CMIP Services and Topology Agent Guide	SC31-8828	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.

Diagnosis

Title	Number	Description
z/OS Communications Server: IP Diagnosis Guide	GC31-8782	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.
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*	GC31-6850 GC31-6851	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.
z/OS Communications Server: SNA Data Areas Volume 1 and *z/OS Communications Server: SNA Data Areas Volume 2*	GC31-6852 GC31-6853	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.

Messages and codes

Title	Number	Description
z/OS Communications Server: SNA Messages	SC31-8790	This document describes the ELM, IKT, IST, ISU, 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.

APPC Application Suite

Title	Number	Description
z/OS Communications Server: APPC Application Suite User's Guide	SC31-8809	This documents the end-user interface (concepts, commands, and messages) for the AFTP, ANAME, and APING facilities of the APPC application suite. Although its primary audience is the end user, administrators and application programmers may also find it useful.

Title	Number	Description
z/OS Communications Server: APPC Application Suite Administration	SC31-8835	This document contains the information that administrators need to configure the APPC application suite and to manage the APING, ANAME, AFTP, and A3270 servers.
z/OS Communications Server: APPC Application Suite Programming	SC31-8834	This document provides the information application programmers need to add the functions of the AFTP and ANAME APIs to their application programs.

Index

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+919-254-4028
- 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.

Program Number: 5694–A01 and 5655–G52

Printed in USA

GC31-8782-06

Spine information:

IBM z/OS Communications Server z/OS V1R7.0 Comm Svr: IP Diagnosis Guide Version 1 Release 7