OS/390 IBM Communications Server

IP Diagnosis Guide

Version 2 Release 10

SC31-8521-04

OS/390 IBM Communications Server

IP Diagnosis Guide

Version 2 Release 10

SC31-8521-04

Note:
Before using this information and the product it supports, be sure to read the general information under “Appendix G. Notices” on page 553.

Fifth Edition (September 2000)

This edition applies to OS/390 V2R10 (Program Number 5647-A01).

Publications are not stocked at the address given below. If you want more IBM publications, ask your IBM representative or write to the IBM branch office serving your locality.

A form for your comments is provided at the back of this document. If the form has been removed, you may address comments to:

IBM Corporation
Software Reengineering
Department G7IA/ Bldg 503
Research Triangle Park, NC 27709-9990

If you prefer to send comments electronically, use one of the following methods:

Fax (USA and Canada):
1-800-227-5088

Internet e-mail:
usib2hpd@vnet.ibm.com

World Wide Web:
http://www.ibm.com/s390/os390/

IBMLink:
CIBMORCF at RALVM17

IBM Mail Exchange:
tkinlaw@us.ibm.com

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 Book

This book tells you how to diagnose and report problems occurring in the IBM®® OS/390 Transmission Control Protocol/Internet Protocol (TCP/IP). Additional information is provided for diagnosing problems with selected applications that are part of IBM Communications Server for OS/390 V2R10 (CS for OS/390).

CS for OS/390 is an integral part of the OS/390® family of products. For an overview and map of the documentation available for OS/390 V2R10, refer to *OS/390 Planning for Installation*.

Who Should Use This Book

Use this book if you are a system programmer to diagnose problems with TCP/IP or to diagnose problems with CS for OS/390 components.

To use this book, you should be familiar with OS/390 TCP/IP Services and the TCP/IP suite of protocols.

How to Use This Book

Use this book to perform the following tasks:

- Diagnose and solve problems in an CS for OS/390 installation
- Describe problems to the IBM Software Support Center and document the problems appropriately

How This Book Is Organized

Part 1 of this book contains general information about diagnostic procedures and tools.

- "Chapter 1. Overview of the Diagnosis Procedure" on page 3 describes the procedure when diagnosing problems and reporting them to the IBM Software Support Center.
- "Chapter 2. Selecting Tools and Service Aids" on page 7 provides an overview of specific tools and service aids for TCP/IP diagnosis. These include dumps, traces, and aids such as common storage tracking. This chapter also identifies the types of documentation, including machine-readable documentation, you need to submit when reporting a problem to the IBM Software Support Center.
- "Chapter 3. Diagnosing Abends, Loops, and Hangs" on page 21 describes how to analyze abends, loops, and hangs.
- "Chapter 4. Diagnosing Network Connectivity Problems" on page 25 describes how to diagnose tips regarding network-connectivity problems, including the commands you can use to diagnose these problems.

Part 2 of this book describes the types of traces, control blocks, and IPCS commands available in CS for OS/390:

- "Chapter 5. TCP/IP Services Traces and IPCS Support" on page 47 describes how to use the TCP/IP Services traces and IPCS support. Component Trace, Packet Trace, Socket Data Trace, and Configuration Profile Trace are described.
- "Chapter 6. IPCS Subcommands for TCP/IP" on page 87 provides information about IPCS commands including, for example, TCPIPCS.

Part 3 of this book provides diagnostic information about the types of problems you may encounter with selected CS for OS/390 components:

The appendixes provide additional diagnostic-related information:

- "Appendix F. Information Apars" on page 551 provides information about how to read the syntax diagrams used in this book.
- "Appendix G. Notices" on page 553 provides information about legal notices and trademarks used in this book.

What Is Not in This Book

This book **does not** cover the LESSTRACE, MORETRACE, TRACE and NOTRACE commands, which have been replaced by CTRACE.

Where to Find More Information

"Bibliography" on page 559 describes the books in the IBM Communications Server for OS/390 library, arranged according to task. The bibliography also lists the titles and order numbers of books related to this book, or cited by name in this book.

Note: Most licensed books were declassified in OS/390 V2R4 and are now included on the OS/390 Online Library Collection, SK2T-6700. The remaining licensed books appear in unencrypted BookManager® softcopy and PDF form on the OS/390 Licensed product Library, LK2T-2499.

Where to Find Related Information on the Internet

You might find the following information helpful.

You can read more about VTAM, TCP/IP, OS/390, and IBM on these Web pages. For up-to-date information about Web addresses, please refer to informational APAR II11334.

Home Page	Web address
IBM Communications Server product	http://www.software.ibm.com/network/commserver/
IBM Communications Server support	http://www.software.ibm.com/network/commserver/support/
OS/390	http://www.ibm.com/s390/os390/
OS/390 Internet Library	http://www.ibm.com/s390/os390/bkserv/
IBM Systems Center publications	http://www.redbooks.ibm.com/
IBM Systems Center flashes	http://www-1.ibm.com/support/techdocs/atsmastr.nsf
VTAM and TCP/IP	http://www.software.ibm.com/network/commserver/about/csos390.html
IBM	http://www.ibm.com

For definitions of the terms and abbreviations used in this book, you can view or download the latest *IBM Networking Softcopy Glossary* at the following Web address:

http://www.networking.ibm.com/nsg/nsgmain.htm

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.

How to Contact IBM Service

For telephone assistance in problem diagnosis and resolution (in the United States or Puerto Rico), call the IBM Software Support Center anytime (1-800-237-5511). You will receive a return call within eight 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.

Summary of Changes

Summary of Changes
for SC31-8521-04
IBM Communications Server for OS/390 V2R10

This edition contains new and changed information, indicated by vertical lines in the left margin.

New Information

- "Diagnosing Sysplex Distributor Problems" on page 39
- FFST probes added
  - EZBIEC07
  - XCF probes
- Additional popper diagnostic aids
- Load module information
- DDNAME trace support
- TRMD
- SNMP TRAPFWD trace
- Additional ARP tracing information
- CTRACE fields, options, and option groups, which include the following:
  - ACCESS
  - CSOCKET
  - DLC
  - IN
  - LATCH
  - OETCP
  - OEUDP
  - PING
  - ROUTE
  - RW
  - SMTP
  - SOCKAPI
  - SYSTEM
  - TC
  - TN
  - UD
- TCPIPCS commands, which include the following:
  - API
  - CONNECTION
  - FIREWALL
  - FRCA
  - HASH
  - HELP
  - POLICY
  - VMCF

  - XCF
- Socket API Trace section

Changed Information

- The CTRACE sample files, CTIEZB00 and CTIORA00, and TCP/IP IPCS exit definition file EZAIPCSP are now shipped in SYS1.PARMLIB instead of hlq.SEZAINST.
- IPCS commands
- Default destination of trace and debug output for omproute traces
- ARP displayed values in trace outputs
- TCP/IP ROUTE sample
- TCP/IP PROFILE sample
- ENGINE, INIT, OPCMDS, OPMSGS, and QUEUE CTRACE trace options are aliases
- The SYS1.PARMLIB member CTIEZB00
- Maximum buffer size has increased from 16M to 256M
- FFST console example
- The term SecureWay® has been removed from our product name. The new product name is IBM Communications Server for OS/390.

Deleted Information

- EZBIEC02 IOCTL enablement probe
- REXX run-time library
- TCPIPCS PROFILE command ddname and dataset_name parameters
- TCPIPCS INETSTAT and SKSH commands
- SYSCALL and CTC options, from the SYS1.PARMLIB member CTIEZB00

Summary of Changes
for SC31-8521-03
SecureWay Communications Server for OS/390 V2R8

This edition contains new and changed information, indicated by vertical lines in the left margin.

New Information

- Diagnosing Policy Agent Problems
- Diagnosing RSVP Agent Problems
- Sample component trace full format output
- EZBIEC07 IOCTL probe

Changed Information

- The term eNetwork is replaced by SecureWay as part of our product name. The new name is SecureWay Communications Server.
- The Bibliography has been revised to show book number dash levels and delivery format.
- The SYS1.PARMLIB member CTIEZB00
- The operating system Trace command OPTIONS response, updated to reflect additional trace options

Summary of Changes
for SC31-8521-02
eNetwork Communications Server for OS/390 V2R7

This book contained information about changes for CS for OS/390 V2R7.

The following enhancements were part of CS for OS/390 Version 2 Release 7:

- A new CTRACE option, AFP, for Fast Response Cache Accelerator.
- A new CTRACE option, SPACKET, for OMPROUTE. SPACKET traces inbound and outbound packets sent between the SNMP agent and the OMPROUTE subagent.
- A new option for the MODIFY command that lets you start and stop message logging for the OMPROUTE subagent. It also lets you stop DPI tracing.
- The following changes to SNMP:
  - A new standards-based network-management framework, SNMPv3, replaces the experimental SNMPv2u. Diagnostic procedures are now available for debugging problems with SNMPv3.
  - The search order for the /etc/snmpv2.conf file has been changed, allowing it to reside in a location other than the /etc directory. The file is now referred to generically as the OSNMP.CONF file.
- FFST probes for the following components:
  - TCP/IP Base (EZBABCxx)
  - Transmission Control Protocol (EZBTCCxx)
  - Update Datagram Protocol Layer (EZBUDCxx)
  - Sockets Module (EZBSMCxx)
  - Streams (EZBSKCxx)
  - Raw IP Layer (EZBRWCxx)
  - Internet Protocol (EZBIPCxx)
- Information about the OMPROUTE DISPLAY commands was moved to the *OS/390 IBM Communications Server: IP Configuration Guide*.
- The static routes associated with deleted interfaces in the routing table no longer appear in the reports generated with the NETSTAT ROUTE, NETSTAT GATE, onetstat -r, and onetstat -g commands.
- The run time for the REXX execs for the TCPIPCS subcommand is provided by TCP/IP and is located in the *tcpip*.SEZAMIG data set. Use the TSOLIB command or the TASKLIB parameter (when invoking IPCS) to add *tcpip*.SEZAMIG as a STEPLIB data set in your TSO logon procedure.

  Note: If you use the run time located in *tcpip*.SEZAMIG, you may not be able to run other compiled REXX procedures. If you want to use other compiled REXX procedures, do not place *tcpip*.SEZAMIG in your LINKLIST.
- Procedures for diagnosing problems with OS/390 UNIX System Services (OS/390 UNIX) OMPROUTE, a routing application that runs outside the TCP/IP Services stack. OMPROUTE obtains the initial static contents of the stack routing table, informs the stack of changes that it has made to a route, and installs those changes in the stack routing table.
- Procedures for diagnosing problems with OS/390 UNIX sendmail.
- MVS® Component Trace support for the OMPROUTE application.
- First failure support technology, which provides immediate notification and first failure data capture for software anomalies.

- A new TCPIPCS subcommand, TCPIPCS TELNET. Invocation of this command displays either the address or address and contents of Telnet control blocks.

Note: As part of the name change of OpenEdition to OS/390 UNIX System Services, occurrences of OS/390 OpenEdition were changed to OS/390 UNIX System Services or its abbreviated name, OS/390 UNIX. OpenEdition may continue to appear in messages, panel text, and other code with OS/390 UNIX System Services.

Part 1. General Diagnosis Information

Chapter 1. Overview of the Diagnosis Procedure

To diagnose a problem suspected to be caused by CS for OS/390, first identify the problem, then determine if it is a problem with TCP/IP, and finally, if it is a problem with TCP/IP, 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 book 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.

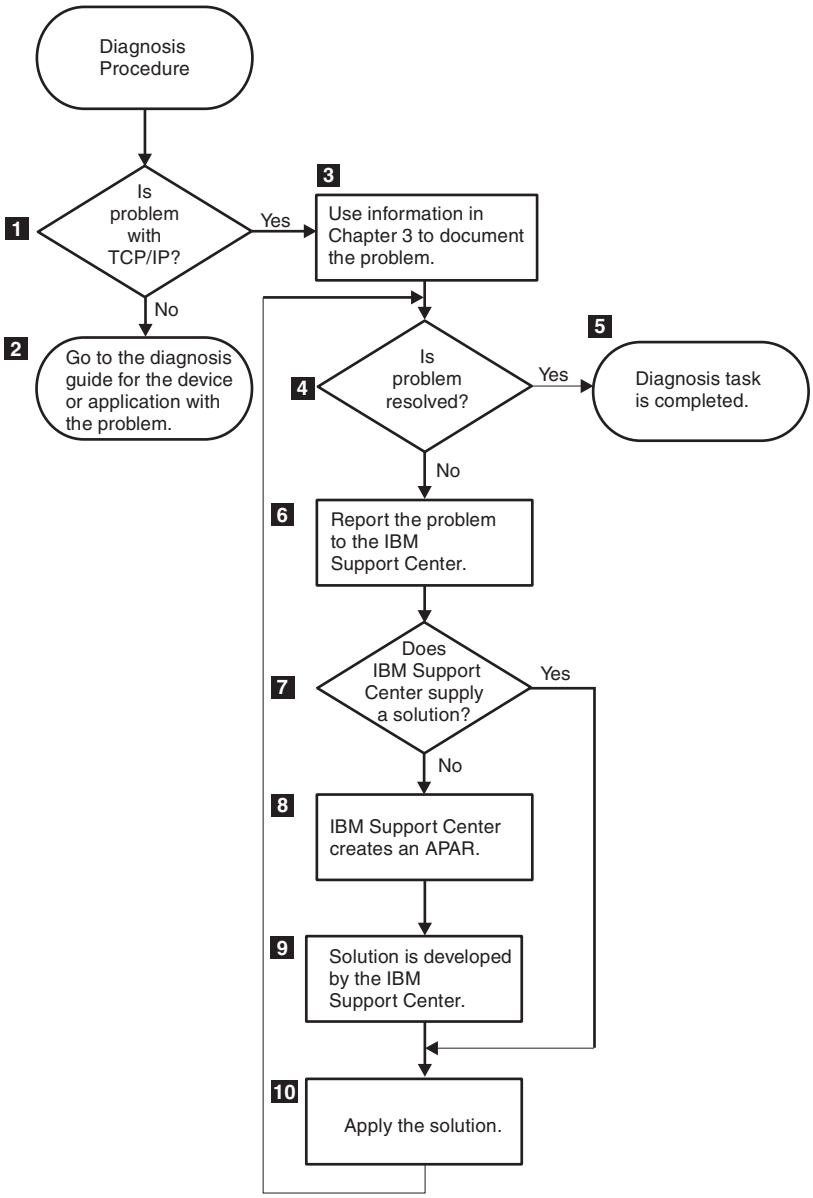


Figure 1. Overview of the Diagnosis Procedure

1 **Determine if the source of the problem is TCP/IP.**

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.**

Refer to “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.

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 will continue to work with you to solve the problem. Go to Step **8** .

8 **Create an APAR.**

If the IBM Software Support Center does not find a solution, they will create an authorized program analysis report (APAR) on 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.

Chapter 2. Selecting Tools and Service Aids

This chapter introduces the tools and service aids that CS for OS/390 provides for diagnosis. As used in this book, 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.
- "What Tools and Service Aids Are Available?" on page 11 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.
- "Guidelines for Machine-Readable Documentation" on page 17 describes the guidelines for submitting machine-readable documentation.
- "Submitting Documentation Electronically" on page 18 describes how to send documentation electronically to IBM using FTP or e-mail.
- "Necessary Documentation" on page 18 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, depending on the problem or need:

- Selecting a Dump (see Table 1)
- Selecting a TCP/IP Services Component Trace (see Table 2 on page 8)
- Selecting a Service Aid (see Table 3 on page 11)

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

Refer to "Guidelines for Machine-Readable Documentation" on page 17 for information about submitting dumps and traces to the IBM Software Support Center.

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

Table 1. Selecting a Dump

What Is the Problem?	Type of Dump to Use
Abnormal end of an authorized program or a problem program.	ABEND dump See "Analyzing Abends" on page 21 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 22 for detailed information.

Table 2. Selecting a Trace

What Is the Problem?	Type of Trace or Command to Use	Trace Output Location
Network Connectivity See “Chapter 4. Diagnosing Network Connectivity Problems” on page 25 for detailed information.	oping, ARP (onetstat -R) For information on oping, see “Using PING and oping” on page 29. For information on onetstat -R, see “onetstat -R” on page 36.	n/a
	Packet trace See Chapter 5. TCP/IP Services Traces and IPCS Support for detailed information about packet trace.	CTRACE managed data set.
TCP/IP Socket application See “Socket API Traces” on page 69 for detailed information.	Component Trace (SYSTCPIP) SOCKAPI option	TCP/IP address space or external writer
LPR Client See “LPR Client Traces” on page 170 for detailed information.	LPR command with the TRACE option	sysout
LPD Server See “LPD Server Traces” on page 176 for detailed information.	See “LPD Server Traces” on page 176 for ways to activate traces.	SYSPRINT
OS/390 UNIX FTP Server See “Chapter 8. Diagnosing File Transfer Protocol (FTP) Problems” on page 191 for detailed information.	OS/390 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 *OS/390 IBM Communications Server: IP Configuration Guide* for more detailed information about syslogd.
OS/390 UNIX Telnet See “Chapter 9. Diagnosing OS/390 UNIX Telnet Problems” on page 219, for detailed information.	OS/390 UNIX Telnet traces	syslogd
Telnet See “Chapter 10. Diagnosing Telnet Problems” on page 225 for detailed information.	Telnet traces	TCP/IP address space or external writer
SMTP See “RESOLVER Trace” on page 250 for detailed information.	Resolver Trace (see also “Debugging with a Resolver Directive” on page 297	Job log output
Popper See “Chapter 12. Diagnosing OS/390 UNIX sendmail and Popper Problems” on page 253 for detailed information.	Popper Messages	syslogd
SNALINK LU0 See “Chapter 13. Diagnosing SNALINK LU0 Problems” on page 261 for detailed information.	IP Packet Trace	CTRACE managed data set
	Debug Trace	SNALINK LU0 address space

Table 2. Selecting a Trace (continued)

What Is the Problem?	Type of Trace or Command to Use	Trace Output Location
SNALINK LU6.2 See "Chapter 14. Diagnosing SNALINK LU6.2 Problems" on page 273 for detailed information.	TRACE DETAIL ALL	SYSPRINT
	IP Packet Trace	CTRACE managed data set
	TCP/IP Internal Trace	SYSDEBUG
	VTAM Buffer Trace	GTF managed data set, refer to *OS/390 IBM Communications Server: SNA Diagnosis V1 Techniques and Procedures*
Dynamic Domain Name System (DDNS) See "Chapter 15. Diagnosing Dynamic Domain Name Server (DDNS) Problems" on page 295 for detailed information.	Error messages	syslogd
	Resolver Trace	Job log output
	TCP/IP component trace	CTRACE managed data set
OS/390 UNIX REXEC See "Chapter 17. Diagnosing OS/390 UNIX REXEC, REXECD, and RSHD Problems" on page 309.	OS/390 UNIX REXEC debug trace	syslogd
OS/390 UNIX REXECD See "Chapter 17. Diagnosing OS/390 UNIX REXEC, REXECD, and RSHD Problems" on page 309.	OS/390 UNIX REXECD debug trace	syslogd
OS/390 UNIX RSHD See "Chapter 17. Diagnosing OS/390 UNIX REXEC, REXECD, and RSHD Problems" on page 309.	OS/390 UNIX RSHD debug trace	syslogd
Network Database System (NDB) See "Chapter 18. Diagnosing Network Database System (NDB) Problems" on page 313 for detailed information.	NDB Trace	Job log output
X Windows and OSF/Motif See "Chapter 19. Diagnosing X Window System and OSF/Motif Problems" on page 329 for detailed information.	XWTRACE and XWTRACEC (environment variables)	stderr
SNMP See "Chapter 20. Diagnosing Simple Network Management Protocol (SNMP) Problems" on page 333 for detailed information.	Manager Traces	Console (osnmp) or SYSPRINT (NetView SNMP)
	SNMP Agent Traces	syslogd
	TCP/IP Subagent Traces	
	OMPROUTE Subagent Traces	
	SLA Subagent Traces	
	TRAPFWD Traces	
Policy Agent See "Chapter 21. Diagnosing Policy Agent Problems" on page 377 for detailed information.	Log file	See *OS/390 IBM Communications Server: IP Configuration Guide*

Table 2. Selecting a Trace (continued)

What Is the Problem?	Type of Trace or Command to Use	Trace Output Location
RSVP Agent See "Chapter 22. Diagnosing RSVP Agent Problems" on page 395 for detailed information.	Log file	See *OS/390 IBM Communications Server: IP Configuration Guide*
Traffic Regulator Management Daemon (TRMD) See "Chapter 23. Diagnosing Traffic Regulator Management Daemon (TRMD) Problems" on page 409 for detailed information.	Log file	syslogd
OS/390 UNIX OROUTED See "Chapter 24. Diagnosing OROUTED Problems" on page 417 for detailed information.	OS/390 UNIX OROUTED trace	Defaults to syslogd. User can use other parameters to send output to the STDOUT DD statement in the OROUTED cataloged procedure. Refer to "Where to Send OROUTED Trace Output" on page 425.
OMPROUTE See "Chapter 25. Diagnosing OMPROUTE Problems" on page 435.	Component Trace For detailed information about OMPROUTE Component Trace, see "TCP/IP Services Component Trace for OMPROUTE" on page 447.	CTRACE managed data set
	OMPROUTE Trace For detailed information, see "OMPROUTE Traces and Debug Information" on page 438.	stdout
NCPROUTE See "Chapter 26. Diagnosing NCPROUTE Problems" on page 451 for detailed information.	NCPROUTE Traces	SYSPRINT
X.25 NPSI See "Chapter 27. Diagnosing X.25 NPSI Problems" on page 475 for detailed information.	Server activity log	SYSPRINT
IMS See "Chapter 28. Diagnosing IMS Problems" on page 485 for detailed information.	IP Packet Trace	CTRACE managed data set
	TCP/IP Internal Trace	CTRACE managed data set
	IMS Trace	See *IMS/ESA Utilities Reference: System*
CICS See "Chapter 30. Diagnosing Problems with CICS" on page 505 for detailed information.	CICS external trace data set (auxtrace)	See *CICS/ESA 5.2 Problem Determination Guide*
	TCP/IP Internal trace	CTRACE managed data set

Table 3. Selecting a Service Aid

What Is the Problem?	Type of Service Aid to Use
System or hardware problem: need a starting point for diagnosis or when diagnosis requires an overview of system and hardware events in chronological order.	LOGREC data set or EREP Refer to *OS/390 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 *OS/390 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 may be used to help set a SLIP trap.	Service Level Indication Processing (SLIP) Refer to *OS/390 MVS System Commands* for detailed information.
Diagnosis requires formatted output of problem data, such as a dump or trace.	IPCS Refer to *OS/390 MVS IPCS User's Guide* for detailed information.

What Tools and Service Aids Are Available?

This section provides an overview of the tools and service aids in a little more detail. The sections that follow contain a brief description of each tool or service aid, some reasons why you would use it, and a reference to the chapter or book that covers the tool or service aid in detail. (Most of the detailed information on tools and service aids is in this book.) A description of tools and service aids are covered in the following sections:

- Dumps (see Table 4 on page 12)
- Traces (see Table 5 on page 13)
- First Failure Support Technology (see "First Failure Support Technology (FFST)" on page 14)
- Display Commands (see "Display Commands" on page 16)
- System Service Aids (see Table 6 on page 16)

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

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

Dumps

Table 4 on page 12 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 *OS/390 MVS Diagnosis: Tools and Service Aids* for detailed information.
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 to solve a problem is needed. 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 *OS/390 MVS Diagnosis: Tools and Service Aids* for detailed information.
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.
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 22 for detailed information.

Traces

Table 5 describes the types of available traces.

Table 5. Description of Traces

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 47 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 “Socket Data Trace” on page 64 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’re 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 *OS/390 MVS Diagnosis: Tools and Service Aids* for detailed information.
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 *OS/390 MVS Diagnosis: Tools and Service Aids* for detailed information.
Packet Trace	Use a packet trace to obtain traces of IP packets flowing from and into TCP/IP on a CS for OS/390 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 47 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 *OS/390 MVS Diagnosis: Tools and Service Aids* for detailed information.
VTAM Trace	CS for OS/390 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 *OS/390 IBM Communications Server: SNA Diagnosis V2 FFST Dumps and the VIT* for detailed information.

Table 5. Description of Traces (continued)

Trace	Description
OS/390 UNIX Applications	OS/390 UNIX applications send debug and trace output to syslogd. For more information on individual components, such as OS/390 UNIX FTP, OS/390 UNIX SNMP, or OS/390 UNIX OROUTED, refer to those chapters in this manual. **Reference:** Refer to the *OS/390 IBM 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 C. First Failure Support Technology (FFST)” on page 523 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 15 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 will be 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 has been 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:

- The date and time that the probe was triggered
- The system name from the CVTSNAME field
- The product name (TCP)
- The component identifier and the release number of the product triggering the probe
- The hardware identification information:
  - Machine type
  - Serial number
  - Model number
  - Plant code
- The dump data set and volume if a dump was taken
- The probe statement
- The statement description
- The 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 *OS/390 IBM Communications Server: IP Configuration Reference*.

DISPLAY TCPIP,,STOR

Use the DISPLAY TCPIP,,STOR command to display the location and level of a TCP/IP stack module. You can use this command to verify that the load module has the appropriate service level.

System Service Aids

Table 6 lists the service aids supported by CS for OS/390.

Table 6. Description of Service Aids

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 lots of 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 *OS/390 MVS Diagnosis: Tools and Service Aids* for detailed information.
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 *RMF User's Guide* for more information about RMF. • Refer to the *OS/390 MVS Initialization and Tuning Guide* for detailed information about requesting common storage tracking. • Refer to the VSM chapter of the *OS/390 MVS IPCS User's Guide* for information about the IPCS VERBEXIT VSMDATA subcommand.
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 and SYSABEND and SYSUDUMP ABEND dumps, are preformatted and are not formatted using IPCS. **Reference:** Refer to the *OS/390 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 *OS/390 MVS Diagnosis: Tools and Service Aids* for detailed information.

Table 6. Description of Service Aids (continued)

Service Aid	Description
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 *OS/390 MVS System Commands* for detailed information.

Guidelines for Machine-Readable Documentation

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, documentation may be submitted in machine-readable form (that is, on tape). Machine-readable documentation can be handled most efficiently by the IBM Support Center if it conforms to the following guidelines when creating the tape (or tapes).

When preparing machine-readable documentation for submission in an MVS environment, the following guidelines should be followed:

1. Submit dumps and traces on tape.
   - For dumps:
     Dump data should not be formatted in any way prior to or during the transfer of the dump to tape.
     The DCB parameters of the dump data set should not be changed. The DCB parameters should be:
     LRECL=4160, BLKSIZE=4160, RECFM=F (for CS for OS/390.)
   - For external CTRACE, IP packet trace, and data trace:
     CTRACE data should not be formatted in any way prior to or during the transfer to tape. 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 *IPCS Command Reference*.
   - For GTF traces:
     GTF trace data should be moved from the trace data set (which is usually SYS1.TRACE) to tape using IEBGENER only.
     The DCB parameters for a GTF trace should be one of the following:
     LRECL=4092, BLKSIZE=4096, RECFM=VB
     LRECL=4092, BLKSIZE=32760, RECFM=VB

   For both traces and dumps, do not reblock the data (that is, do not use a different BLKSIZE value) when moving the information to tape. Only the DCB parameters shown above should be used.

   Note: 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.

2. Submit other types of information (such as TCP/IP traces, configuration files, console logs, and so forth) on paper or tape. **Tape is preferable**. 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).
3. Tapes that are submitted to the TCP/IP support team may be standard label (SL) or nonlabel (NL). Cartridge (3480) or reel tapes may be used. 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.
4. 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.

Submitting Documentation Electronically

You can send documentation to IBM using File Transfer Protocol (FTP) or e-mail. If you use FTP, compress all dumps and traces with the TRSMAIN (MVS terse) program, and send the data in BINARY mode. To obtain TRSMAIN and detailed instructions on its use, follow these steps:

1. FTP to the web site at *service.software.ibm.com*.
2. Login 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 will 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.

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 MVS/ESA). Include the MVS or OS/390 release level.

LE Runtime 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
An number that uniquely identifies each TCP/IP release.

Table 7 lists the TCP/IP-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
CS for OS/390 Version 2 Release 10	SMP/E	The following identifiers are associated with this stack: • HTCP50A (base) • JTCP56A (NPF) • JTCP59A (HFS) • JTCP5PA (HSAS) • HTCP38X (X Window) • HTCP52A (Kerberos 56-bit DES) • HTCP53A (Kerberos non-DES • JTCP5KA (IP Security Triple DES) **Note:** High Speed Access Services are no longer supported, but the JTCP5PA (HSAS) FMID is required by OS/390 Unix System Services.

The following are component ID numbers for CS for OS/390.

Licensed IBM Program
CS for OS/390

Component ID Number
5655HAL00

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.

Note: 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 book.

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 *OS/390 C/C++ IBM Open Class Library Reference*.
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 *OS/390 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 *OS/390 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	LE produces certain types of abends detected for OS/390 UNIX applications such as OS/390 UNIX Telnet. CEEDUMPs are usually written to the current working directory in the hierarchical file structure.	Refer to the *OS/390 Language Environment for OS/390 and VM 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 will have to recreate the abend or wait for it to occur again.

Note: For more information about debugging the abends and the system abends (for example, abends 0C4, 0C1, and 878), refer to the *OS/390 MVS Diagnosis: Procedures*.

Analyzing Loops

If processing stops or if TCP/IP doesn't respond to commands, TCP/IP could be in a loop. Some indicators of a loop are:

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

If the problem is a loop, use the following procedure 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 *OS/390 MVS Diagnosis: Tools and Service Aids* for more information about the DUMP command.

     Note: Please make sure that the following storage areas are dumped: CSM, TCP/IP, VTAM address space and any dataspace used by the DLCS, RGN, CSA, LSQA, NUC, PSA, LPA, and TCPIPDS1. TCPIPDS1 is the data space containing the TCP/IP component trace records. For information on dumping this data space, see "Appendix A. Collecting Component Trace Data" on page 513.
   - **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 the *VTAM Diagnosis* publication.

     – Take a stand-alone dump. Refer to the *OS/390 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 the *OS/390 MVS Diagnosis: Tools and Service Aids* or the *OS/390 MVS IPCS Commands* for more information about the LOGDATA option.

   Note: The SYSERROR data set might contain additional information to help you diagnose the problem.
3. **Is a message involved?**

   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 *OS/390 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

Analyzing Hangs

If the problem is a hang, use the following procedure 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 at the time the hang was encountered by the operator or user. Some information about the activity that immediately preceded the hang might be available on the system log or in application program transaction logs.

2. **Does TCP/IP respond to any commands?**

   Determine if TCP/IP responds to commands, such as `oping` or `onetstat`. 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. **Is a particular application (such as OS/390 UNIX FTP or a user-written application) 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

Interconnectivity between network hosts encompasses the physical layer or hardware layer, the protocols such as, TCP and IP, and the applications that use the services of TCP and IP. Isolating network problems is an essential step in successful implementation of a network application.

Diagnostic commands are available for either the OS/390 UNIX environment or the TSO environment. The examples in this chapter are for the OS/390 UNIX environment. You can substitute the equivalent TSO command, as shown in Table 9. For complete descriptions of these commands, refer to the *OS/390 IBM Communications Server: IP User's Guide* .

Table 9. Diagnostic Commands

UNIX command	TSO command	Refer to
oping	PING	"Using PING and oping" on page 29
onetstat	NETSTAT	"Using NETSTAT and onetstat" on page 31
otracert	TRACERTE	"Using TSO TRACERTE and otracert" on page 37

Notes:

1. 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.
2. 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).

For detailed information on internetworking, see "Appendix D. Overview of Internetworking" on page 533.

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.TCPIP.DATA file. An application can also select a stack using the SETIBMOPT socket API.

You can use the **onetstat -p <tcpname>** command to determine the characteristics of the stack through which the application is currently communicating.

Using the information provided by `onetstat -p tcpname`, you can change, if necessary, the *hlq*.PROFILE.TCPIP data set or the application configuration file. Alternatively, the application may need to communicate through another stack.

It is also helpful to understand the search order used by CS for OS/390. See "Appendix B. Search Paths" on page 521.

For more information on running multiple stacks, refer to *OS/390 IBM Communications Server: IP Configuration Reference* .

Problems Connecting to the Server

Figure 3 on page 27 shows the steps to take in diagnosing network connectivity problems.

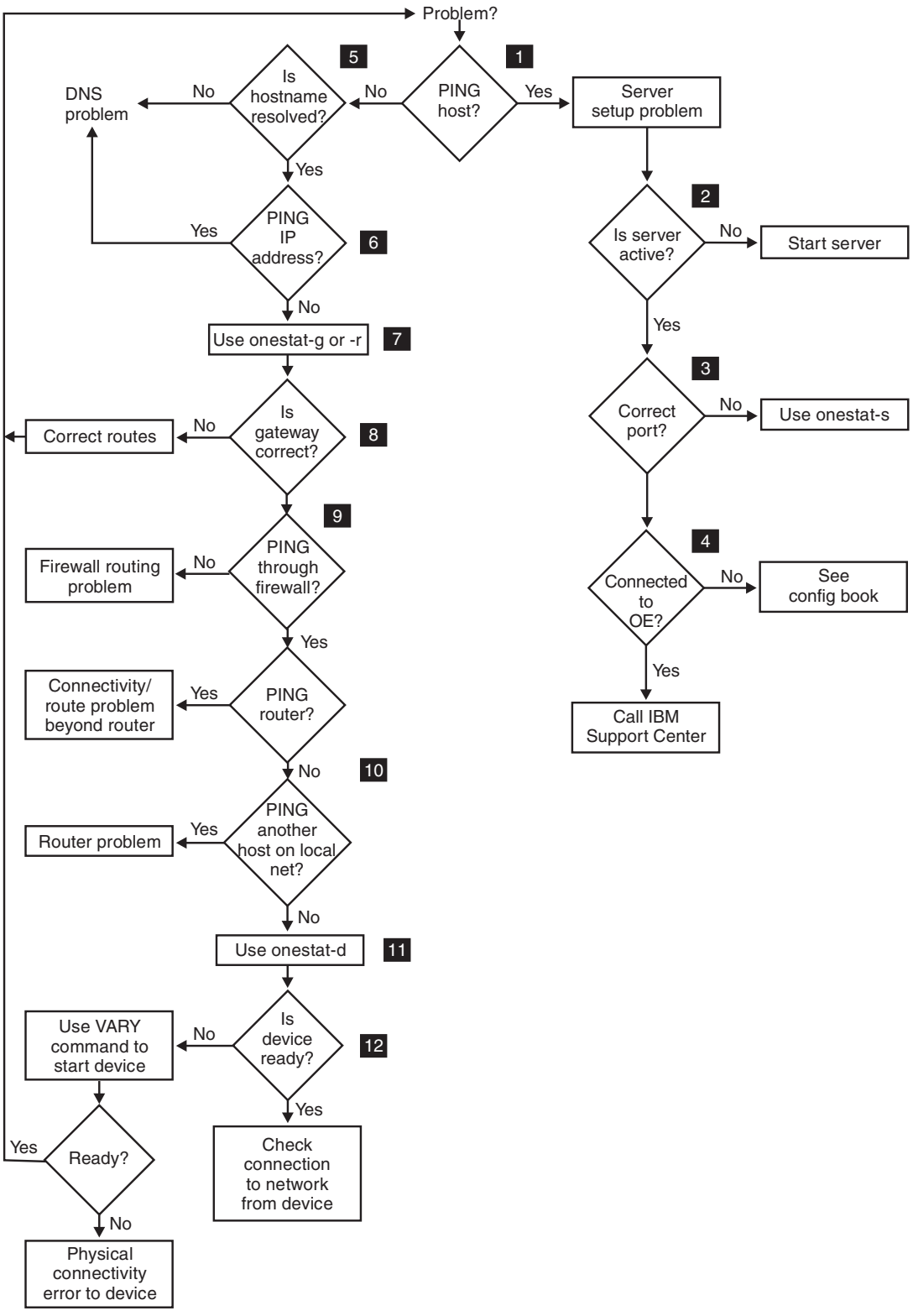


Figure 3. Procedure for Diagnosing Server Connection Problems

Following are explanations of the steps in Figure 3.

1 Use **oping** loopback to assure that TCP/IP is running correctly on your host. Then, **oping** one of your home addresses.

2 Use the **oping** *hostname* command to check connectivity to a remote host. If the host is accessible, you have a server setup problem.

For basic information about the oping command, see Using PING and oping or refer to the *OS/390 IBM Communications Server: IP User's Guide* for details.

3 Is the server started? If not, start the server.

4 Is the server started on the correct port? Use the onetstat -s command to determine which port the server is on. If the server is not on the correct port, configure it correctly.

For basic information about the onetstat -s command, see "Using NETSTAT and onetstat" on page 31 or refer to the *OS/390 IBM Communications Server: IP User's Guide* for details. For details on server configuration, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.

5 Is the server properly connected to OS/390 UNIX? If it is not, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*. If it is, call the IBM Support Center. For the types of documentation you need to provide, see "Documentation for the IBM Support Center" on page 43.

6 Was the hostname resolved? If it was not, check the resolver trace. The first thing the resolvers do is look for the value of the RESOLVER_CONFIG environment variable. You can set the value in the parameter list of the JCL of the started procedure, using the keyword ENVAR. For an example, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*. For more information on using the resolver trace, refer to *Accessing OS/390 Open/Edition MVS from the Internet*.

If, after checking the trace, you see that the hostname was not resolved, you have a name resolution problem. Check the following files to make sure the name is correct:

- /etc/resolv.conf or *hlq*.TCPIP.DATA
- /etc/hosts
- *hlq*.HOSTS.LOCAL

In addition, you may want to check other configuration files. For details, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.

7 If the hostname resolves, use the oping command with the IP address of the host to check connectivity. If you get a response, you have a name resolution problem.

8 Use the onetstat -r or the onetstat -g command to display routes to the network. Verify whether or not TCP/IP has a route to the destination.

9 Use the **oping** command to check connectivity to the gateway (router). If you do not get a response, the router may be down or the host may not be connected to the network. If you get a response using the oping command, you have a route problem. To resolve it, do the following:

1. Verify routes to your host.
2. Check for connectivity or route problems beyond the router.

10 Use the **oping** command to check connectivity from the secure to the nonsecure side of the firewall. If you do not get a response, make sure the

static routing is set properly on the firewall. To verify static routes, use the onetstat -g and onetstat -h commands. The onetstat -h command shows the addresses for the links (adapters) defined to TCP/IP and onetstat -g shows information about the static routing. You should have an interface route for each interface and your default route should point to the router on the nonsecure side of the firewall.

If you are able to **oping** the secure interface on the firewall and the nonsecure interface on the firewall, but are unable to **oping** the external router, you are probably missing a return route in the external router. The return route in the external router is a static route to your secure network listing the firewall nonsecure interface at the gateway. If you are using unregistered addresses in your secure network, you will not be able to setup a return route.

For additional information about firewall, refer to *OS/390 Firewall Technologies Guide and Reference*.

11 Use the **oping** command to check connectivity to another host in the local network.

12 Use the onetstat -d command to verify device status. If the device is ready, check the physical connectivity to the network (for example, 3172 or RS/6000® LAN connection).

13 Try to start the device using the `VARY TCPIP...START, devname` command. (For information about this command, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.) If that is unsuccessful, do the following:

1. Verify connectivity to the device (channel connector).
2. Check whether you have a problem at the device itself.

Using PING and oping

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 **oping** command is the OS/390 UNIX version of the TSO PING command. It runs from a UNIX shell.

For the most part, PING and **oping** produce the same result. (Table 10 lists some commonly used options.) However, **oping** is case sensitive and must be entered in lowercase. For a complete description of the PING and **oping** commands, refer to the *OS/390 IBM Communications Server: IP User's Guide*.

Table 10. OS/390 UNIX oping Options Compared with TSO PING Options

OS/390 UNIX oping	TSO PING
-l	Length
-c	Count
-t	Timeout
-p	n/a

- **Use PING or oping loopback to verify the installation of TCP/IP in the CS for OS/390 environment**.

  The oping loopback is essentially an internal software test. This command uses the IP address 127.0.0.1, which is the standard loopback address. An IP packet is not sent to a physical device.

```
oping loopback
Ping CS V2R10: Pinging host  (127.0.0.1).
PING: Ping #1 response took 0.001 seconds.
***
```

- **PING or oping a home address to verify the information from the onetstat -h command**.

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

```
oping 193.9.200.1
Ping CS V2R10: Pinging host 193.9.200.1.
PING: Ping #1 response took 0.001 seconds.
***
```

- **PING or oping a host on a directly-attached network to verify the following:**
  - 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 (193.9.200.2) is able to receive and send packets.

```
ping 193.9.200.2
Ping CS V2R10: Pinging host 193.9.200.2.
PING: Ping #1 response took 0.021 seconds.
***
```

- **PING or oping a host on a remote network to verify the following:**
  - 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 (9.67.43.126) has a route back to the local host.

```
oping 9.67.43.126
Ping CS V2R10: Pinging host 9.67.43.126.
PING: Ping #1 response took 0.221 seconds.
***
```

Correcting Timeout Problems

A PING or oping timeout message can occur for many reasons, and various techniques can be used to identify whether the problem is the local OS/390 server or a remote host or router. Possible reasons for a timeout are shown in Table 11:

Table 11. Diagnosis of a Timeout

Problem	Diagnostic Techniques
The device is not transmitting packets to the local network.	Use NETSTAT DEvlinks or onetstat -d to collect information to help you diagnose the problem . (See “onetstat -d” on page 31.)
The remote host is not receiving or transmitting packets on the network.	Use NETSTAT ARp or onetstat -R to display the entry for the remote host. (See “onetstat -R” on page 36.)
The remote host does not have a route back to the local OS/390 server.	Use NETSTAT Gate or onetstat -g on the remote host to make sure it has a route back. (See “onetstat -g” on page 35.)
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 47.)
The ipconfig reassemblytimeout value may be set too low.	Refer to the chapter on configuring the TCP/IP address space in *OS/390 IBM Communications Server: IP Configuration Reference*.

PING and oping Command Return Codes

Following is a list of the return codes generated by the PING and oping commands:

Code	Description
0	Response
4	No response
8	TCP/IP address space failure (PING only)
12	Socket API failure (oping only)
100	Incorrect parameter

Using NETSTAT and onetstat

Use the NETSTAT or onetstat command to verify TCP/IP configuration. The following onetstat command options are covered in this section. For the equivalent NETSTAT commands, see *OS/390 IBM Communications Server: IP User's Guide*.

- "onetstat -h"
- "onetstat -d"
- "onetstat -g" on page 35
- "onetstat -r" on page 36
- "onetstat -R" on page 36

Notes:

1. You can run onetstat or NETSTAT against a TCP/IP stack of the same version. For example, you can run CS for OS/390 onetstat against a CS for OS/390 stack.
2. The information provided in the output from the onetstat command should be checked against the values in the default configuration data set *hlq*.PROFILE.TCPIP. Refer to PROFILE DD statement in the TCP/IP started task procedure for the name of the configuration data set.

For details about the onetstat command, refer to the *OS/390 IBM Communications Server: IP User's Guide*.

onetstat -h

Use the onetstat -h command to verify ADDRESS and LINK values returned by onetstat. Be sure that correct values are coded on the HOME statement in the *hlq*.PROFILE.TCPIP data set.

```
onetstat -h

 MVS TCP/IP onetstat CS V2R10     TCPIP Name: TCPCS       12:34:56
 Home address list:
 Address           Link          Flg
 -------           ----          ---
 9.67.113.27       TR1           P
 9.67.116.91       CTC1
 9.67.116.99       ETH2
 9.67.1.8          OSA90LINK1
 9.67.116.1        X25LINK
 9.67.116.2        CDLCLINK
 9.67.116.3        HCHLINK
 9.67.116.4        PTPLINK
 9.67.116.5        SNA1
 9.67.116.6        SNA62
 127.0.0.1         LOOPBACK
```

onetstat -d

Use the onetstat -d command to display the status and associated configuration values for a device and defined links as coded in the *hlq*.PROFILE.TCPIP data set.

```
onetstat -d

MVS TCP/IP onetstat CS V2R10       TCPIP NAME: TCPCS           08:08:14
DevName: LOOPBACK          DevType: LOOPBACK  DevNum: 0000
  DevStatus: Ready
  LnkName: LOOPBACK          LnkType: LOOPBACK    LnkStatus: Ready
    NetNum: 0   QueSize: 0
    BytesIn: 0000002036      BytesOut: 0000002036
  BSD Routing Parameters:
    MTU Size: 00000             Metric: 00
    DestAddr: 0.0.0.0           SubnetMask: 0.0.0.0
  Multicast Specific:
    Multicast Capability: No
DevName: LCS1              DevType: LCS       DevNum: 0D00
  DevStatus: Ready
  LnkName: TR1               LnkType: TR          LnkStatus: Ready
    NetNum: 0   QueSize: 0
    BytesIn: 0000230896      BytesOut: 0000000056
    ArpMacAddress: Non-Canonical   SrBridgingCapability: Yes
    BroadcastCapability: Yes       BroadcastType: All Rings
  BSD Routing Parameters:
    MTU Size: 00000             Metric: 00
    DestAddr: 0.0.0.0           SubnetMask: 255.255.255.128
  Multicast Specific:
    Multicast Capability: Yes
    Group             RefCnt
    -----             ------
    224.0.0.1         0000000001
DevName: MPCIPAD           DevType: MPCIPA    DevNum: 0000
  DevStatus: Not Active    CfgRouter: Non  ActRouter: Unknown
  LnkName: MPCIPAL           LnkType: IPAQGNET    LnkStatus: Not Active
    NetNum: 0   QueSize: 0
    BytesIn: 0000000000      BytesOut: 0000000000
  BSD Routing Parameters:
    MTU Size: 00000             Metric: 00
    DestAddr: 0.0.0.0           SubnetMask: 255.255.255.128
  Multicast Specific:
    Multicast Capability: Unknown
DevName: HYDRAPFD          DevType: MPCIPA    DevNum: 0000
  DevStatus: Ready         CfgRouter: Pri  ActRouter: Pri
  LnkName: LHYDRAF           LnkType: IPAQENET    LnkStatus: Ready
    NetNum: 0   QueSize: 0   HighSpeed: 0000001000
    BytesIn: 0000000000      BytesOut:  00000000000
 ArpOffload: Yes          ArpOffloadInfo: Yes
  BSD Routing Parameters:
    MTU Size: 00000             Metric: 00
    DestAddr: 0.0.0.0           SubnetMask: 255.255.255.128
  Multicast Specific:
    Multicast Capability: Unknown
DevName: OSATRL90          DevType: ATM       DevNum: 0000
  DevStatus: Not Active
  LnkName: OSA90LINK1        LnkType: ATM         LnkStatus: Not Active
    NetNum: 0   QueSize: 0
    BytesIn: 0000000000      BytesOut: 0000000000
  BSD Routing Parameters:
    MTU Size: 00000             Metric: 00
    DestAddr: 0.0.0.0           SubnetMask: 255.0.0.0
  ATM Specific:
    ATM portName:  OSA90
    ATM PVC Name:  STEPH             PVC Status: Not Active
    ATM LIS Name:  LIS1
    SubnetValue:   9.67.1.0          SubnetMask:    255.255.255.0
    DefaultMTU:    0000009180        InactvTimeOut: 0000000300
    MinHoldTime:   0000000060        MaxCalls:      0000001000
    CachEntryAge:  0000000900        ATMArpReTry:   0000000002
    ATMArpTimeOut: 0000000003        PeakCellRate:  0000000000
    NumOfSVCs:     0000000000
    ATMARPSV Name: ARPSV1
    VcType:        PVC               ATMaddrType: NSAP
    ATMaddr:
    IpAddr:        0.0.0.0
  Multicast Specific:
    Multicast Capability: No
```

Following are brief descriptions of the fields in this example:

- Device name

  The device name.
- Device type

  The device type.
- Device number

  The device number.
- Status of the device

  The following list describes the possible device statuses:

Device Status	Description
Starting	A START of the device has been issued by the operator, and TCP/IP has sent an Activation request to the Data Link Control (DLC) layer.
Sent SETUP Request	DLC has acknowledged the TCP/IP Activation request, and TCP/IP has requested DLC to perform the initial I/O sequence with the device.
Enabling	DLC has acknowledged the TCP/IP Activation request, and TCP/IP has requested DLC to allow data connections to be established for the device.
Connecting	DLC has accepted the ″Initial I/O Sequence″ request.
Connecting2	The control connection for a CLAW device has been established, and the second connection (on which IP traffic is carried) is being established.
Negotiating	The initial I/O sequence with the device is complete, and TCP/IP is performing additional link-layer initialization.
Ready	The initialization sequence with the device is complete. The device is now ready.
Deactivating	DLC has performed the first stage of an orderly device deactivation.
Not active	The device is not active. (The device has never been started, or has been stopped after having been started.)

- Configured router status

  This field is significant only for MPCIPA devices.
- Actual router status

  This field is significant only for a MPCIPA device and is determined when the device starts.
- Link name

  The link name.
- Link type

  The link type.
- Status of the link

  The following list describes the possible link statuses:

Link Status	Description
Ready	A START of the device has been issued by the operator, and TCP/IP has been sent an Activation request to the Data Link Control (DLC) layer.
Not Active	The link is not active. (There is no command to start a link; link activation is normally performed during START device processing. A link will be marked Not Active when: – the device has not yet been started – a failure has been encountered during the link activation phase. (Such a failure will have produced an error message to the operator console, indicating the cause.)

- Net number

  Displays the link adapter number, if applicable.
- Queue size

  This field is significant only for links on LCS and ATM devices.
- Speed

  This field is significant only if the link is active and the TCP/IP stack set the speed.
- Number of bytes received
- Number of bytes transmitted
- ARP MAC addresses

  This field is significant only for token-ring links.
- SR bridging capability

  This field is significant only for token-ring links.
- Broadcast capability

  This field is significant only for links on LCS devices.
- Broadcast type

  This field is significant only for token-ring links.
- ARP offload

  This field is significant only for active links that support ARP offload.
- ARP offload information

  This field is significant only for active links that support ARP offload.
- BSD parameters
- Packet trace settings

  The effective values of each of the packet trace options for the link. This information is displayed only when packet trace settings are defined and on.
- ATM specific information

  This field is significant only for ATM devices and links.

ATM PVC Status	Description
Ready	The initialization sequence for the PVC is complete. The PVC is now ready for use.
Not Active	The PVC is not active. (There is no command to start a PVC; PVC activation is normally attempted during START device processing. A PVC will be marked Not Active when: – the device has not yet been started – the remote side of the PVC is not active – a failure has been encountered during the PVC activation phase. Such a failure will have produced an error message to the operator console, indicating the cause.)

- Multicast specific

  This field is significant only for multicast capable devices.

  If a link is being used to receive multicast data, then all the multicast groups, and the counts of receivers for each multicast group, are displayed. There is no limit to the number of multicast groups for which a link can receive data.

  – Multicast capability

    This field is always Yes for the following devices: CDLC, CLAW, CTC, and MPCPTP.

    For LCS and MPCIPA devices, the multicast capability is only known after the device is in the Ready state. If the device is not yet Ready, the multicast capability will be Unknown.

Notes:

1. No link-related information, packet trace settings, or BSD parameters are displayed for a device that has no link defined.
2. The packet trace setting is displayed only when it is defined and set to ON.
3. ATM specific information is displayed only for ATM devices that have links defined.
4. The LOOPBACK device and link are displayed.

onetstat -g

The onetstat -g 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 OS/390 server. Following is an example of onetstat -g output.

```
onetstat -g

 MVS TCP/IP onetstat CS V2R10     TCPIP Name: TCPCS            12:34:56
 Known gateways:
 NetAddress      FirstHop      Link     Pkt Sz Subnet Mask     Subnet Value
 ----------      --------      ----     ------ -----------     ------------
 Defaultnet      9.67.113.1    TR1      576    <none>
 9.0.0.0         <direct>      TR1      2000   0.255.255.128   0.67.113.0
 9.67.116.90     <direct>      CTC1     4000   HOST
 224.0.0.5       <direct>      TR1      2000   HOST
```

The onetstat -g command provides the following information about each gateway:

- Address of the network
- First hop address
- Link name used by the first hop

  Note: Only the first 8 characters of the link name are displayed by this command. Issue the onetstat -r command to see more than 8 characters of the link name.
- Packet size used by the first hop
- Subnet mask and subnet value

DETAIL

Displays the preceding information plus the metric associated with the cost of the use for the link, and displays the following MVS specific configured parameters for each gateway:

- Maximum retransmit time
- Minimum retransmit time
- Round trip gain
- Variance gain
- Variance multiplier

Note: The static routes associated with deleted interfaces in the routing table no longer appear in the reports generated with the onetstat -g command.

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 commands. (For information on the VARY TCP/IP command, refer to *OS/390 IBM Communications Server: IP Configuration Reference*.)
- Make sure the HOME statement has been coded correctly.
- If static routing is provided by way of the GATEWAY statement, make sure the entries in the statement correlate to a valid link name.

- If static routing is provided by way of the GATEWAY statement, make sure there is a GATEWAY or DEFAULTNET entry that correlates to the NETWORK or HOST addresses available on the network.

onetstat -r

The **onetstat -r** displays routing information. Following is an example of **onetstat -r** output:

```
onetstat -r

MVS TCP/IP onetstat CS V2R10       TCPIP Name: TCPCS            09:51:02
Destination      Gateway          Flags  Refcnt  Interface
-----------      -------          -----  ------  ---------
Defaultnet       9.67.113.1       UG     000000  TR1
9.67.1.9         0.0.0.0          H      000000  OSA00LINK1
9.67.113.0       0.0.0.0          U      000000  TR1
9.67.113.43      0.0.0.0          UH     000000  TR1
127.0.0.1        0.0.0.0          UH     000002  LOOPBACK
198.11.24.106    198.11.10.210    UGHD   000006  LCISCH
198.11.25.104    198.11.22.109    UGH    000000  LMCH2IT22
201.2.10.31      0.0.0.0          UH     000000  VIPLC9020A1F
```

The onetstat -r command provides the following information:

Destination
The address of a destination host or network

Gateway
The gateway used in forwarding packets

Flags The state of the route:
- **U** The route is up.
- **H** The route is to a host rather than to a network.
- **G** The route is a gateway.
- **D** The route was created dynamically by a redirect.

Reference count
The current number of active users for the route

Interface
The link name for the route

Note: The static routes associated with deleted interfaces in the routing table no longer appear in the reports generated with the onetstat -r command. New PMTU routes are displayed. Loopback route is displayed. Implicit (HOME list) routes are displayed..

onetstat -R

Use the command `onetstat -R` *<net address>* to query the ARP cache for a given address. Use `onetstat -R ALL` to query an entire ARP cache table. Make sure onetstat -R displays an ARP entry for the remote hosts.

Following is an example of onetstat -R output:

```
onetstat -R 9.67.112.25

  MVS TCP/IP onetstat CS V2R10      TCPIP Name: TCPCS       12:34:56
  Querying ARP cache for address 9.67.112.25
  Link: TR1              IBMTR: 10005A0019F5
  Route info: 0000
```

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

To ensure the host has a route back to the OS/390 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.

Notes:

1. For more information about **onetstat -R**, refer to the *OS/390 IBM Communications Server: IP User's Guide*.
2. The information provided differs depending on the type of device. See *OS/390 IBM Communications Server: IP Configuration Guide* for information on what is supported for each device.

Using TSO TRACERTE and otracert

The traceroute function displays the route that a packet takes to reach the requested target. Trace starts at the first router and uses a series of UDP probe packets with increasing IP time-to-live (TTL) values to determine the sequence of routers that must be traversed to reach the target host.

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

For the complete syntax of the TSO TRACERTE and OS/390 UNIX otracert command, refer to *OS/390 IBM Communications Server: IP User's Guide*, GC31-8514.

The following examples use the OS/390 UNIX otracert command, however, the TSO TRACERTE command can also be used.

- The otracert command issued to a valid destination:

```
 otracert 9.130.40.72
------------------------------------------------------------
Traceroute to 9.130.40.72 (9.130.40.72).
Use escape C sequence to interrupt
1 buzz-113.tcp.raleigh.ibm.com (9.67.113.1)  7 ms  56 ms  6 ms
2 b500-503-1.raleigh.ibm.com (9.37.32.1)  25 ms  66 ms  19 ms
3 rtp2wf-atm.raleigh.ibm.com (9.37.1.132)  8 ms  19 ms  23 ms
4 9.32.204.30 (9.32.204.30)  60 ms  16 ms  19 ms
5 9.32.1.69 (9.32.1.69)  49 ms  65 ms  48 ms
6 9.32.44.2 (9.32.44.2)  58 ms * *
7 pok008-1.pok.ibm.com (9.117.1.1)  41 ms  44 ms  58 ms
8 * sqv014-1.endicott.ibm.com (9.130.104.12)  148 ms  93 ms
9 gdlvm7.endicott.ibm.com (9.130.40.72)  91 ms  89 ms  87 ms
```

  Following are brief descriptions of some of the items in the preceding example:

  – Each line in this example shows a "hop" to a different router.
  – An asterisk (*) represents a lost packet.
  – The time displayed (7 ms 56 ms, and so on) is in milliseconds. It shows the round-trip time calculated from when the probe was sent from otracert and when otracert received the ICMP reply. Each time displayed is independent; it is the time it takes for otracert to send a probe and receive a reply.

- The otracert command issued to an address of an unreachable host (!H):

```
otracert 1.1.1.1
------------------------------------------------------------
TRACEROUTE TO 1.1.1.1 (1.1.1.1).
USE ESCAPE C SEQUENCE TO INTERRUPT
1 buzz-113.tcp.raleigh.ibm.com (9.67.113.1)  5 ms  5 ms  5 ms
2 b500-503-1.raleigh.ibm.com (9.37.32.1)  8 ms  12 ms  8 ms
3 rtp2wf-atm.raleigh.ibm.com (9.37.1.132)  8 ms  12 ms  11 ms
```

```
4 9.32.204.30 (9.32.204.30)  30 ms  9 ms  8 ms
5 9.32.204.30 (9.32.204.30)  14 ms !H * *
6 * * *
7 * * 9.32.204.30 (9.32.204.30)  10 ms !H
8 * * *
9 * * *
```

Following are brief descriptions of some of the items in the preceding example:

” Unreachable port

!H Unreachable host

!P Unreachable protocol

- The otracert -d hugo command to turn on debug information:

```
$ otracert -d hugo
 <EZACDTRT C> Line <1606>: SETIADDR: dest set to:
 <EZACDTRT C> Line <1606>: SETIADDR: sin_len:  16 sin_family:   2 sin_port
 sin_addr: 0x07432B3E
 Traceroute to hugo (11.7.45.62).
 Use escape C sequence to interrupt
 TRT1623 parsOE Traceroute: host = hugo
 TRT1624 parsOE Traceroute: retryCount = 3
 TRT1625 parsOE Traceroute: port = 4096
 TRT1626 parsOE Traceroute: waitTime = 5
 TRT1627 parsOE Traceroute: maxTTL = 30
 TRT1628 parsOE Traceroute: maxPacket = 40
 TRT1629 Parsed otracert parameters
 TRT0700 openSock: recv socket failed 111(12FC00B0): EDC5111I Permission Denied
 EZZ6354I OE Traceroute: Recv socket() error detected (111/12FC00B0)
 TRT1633 openSock() failed with rc = 12
```

Following are brief descriptions of the some of the items in the preceding example:

host Name of the host you are trying to reach

retryCount
Number of times the probe is sent with the same TTL value

port The number of the unused port used at the destination (defaults to 4096)

waitTime
The wait time in seconds (defaults to 5)

maxTTL
The number of “hops” the trace will take before it dies

maxPacket
The size of the probe packet.

Note: The otracert command uses the site tables, rather than the Domain Name Server, for inverse name resolution. Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for more information about using the site tables. If a host name is found in the site table, it is printed along with its IP address.

Using SNMP Remote PING

Use this 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 *OS/390 IBM Communications Server: IP User's Guide* for more information.

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. Using a combination of onetstat, TSO netstat, and display sysplex commands provide a clear picture of the sysplex. See the *OS/390 IBM Communications Server: IP Configuration Guide* chapter on virtual addressing for an introduction to sysplex distribution with virtual addressing.

Netstat information can be obtained in any of the following ways.

- Using onetstat from the omvs shell
- Using netstat from TSO
- Using the `display netstat` command from the system console

The display netstat command is used in the examples below but any of the three commands can be used.

- For the types of problems where the actual DVIPAs defined on a stack are not what the user expected, confirm the current definitions on a stack (Step 1).
- For Sysplex Distributor workload monitoring, Steps 5 and 6 can be used. If the output from these commands is not what you expected Step 4 will give you the overall picture of all DVIPA activity in your sysplex.
- If the output from Step 4 reveals an expected target stack not listed for a distributed DVIPA use Step 2 on the target stack in question. This will help identify configuration problems on that stack. Note what is required of target stacks. Also verify, with Step 7, that a server application has indeed been activated and bound to the correct port.
- A CTRACE with options XCF, INTERNET, TCP and VTAMDATA on participating stacks is useful in following the flow of packets into and throughout the sysplex. These can be used to:
  1. Identify the connection being received by the distributing stack.
  2. Determine the stack to which the connection will be forwarded.
  3. Verify the connection being forwarded..
  4. Determine the expected target stack receiving and processing the connection.

  Once the connection has been established subsequent packets can be followed in the same manner. When the connection is terminated CTRACE records will record target stacks cleaning up the connection and notifying the distributing stack.

The following are the steps to take in diagnosing sysplex problems.

1. Run the display command `d tcpip,tcpcs,net,vipadcfg` on the distributing stack to confirm it is configured to distribute the DVIPA and how it is to be distributed.

   The following report shows that the TCP/IP identified by TCPCS has been configured to distribute two DVIPAs. Workload for the first DVIPA, 201.2.10.11 ports 20 and 21, will be distributed to all stacks in the sysplex including TCPCS itself. Workload for 201.2.10.12, ports 20 and 21, will only be distributed to the TCP/IP with Dynamic XCF address 193.9.200.2.

   ```
   d tcpip,tcpcs,net,vipadcfg

   EZZ2500I NETSTAT CS V2R10 TCPCS 477
   DYNAMIC VIPA INFORMATION:
   ```

```
VIPA DEFINE:
  IP ADDRESS       ADDRESSMASK      MOVEABLE
  ----------       -----------      --------
  201.2.10.11      255.255.255.240  IMMEDIATE
  201.2.10.12      255.255.255.240  IMMEDIATE

VIPA DISTRIBUTE:
  IP ADDRESS       PORT      XCF ADDRESS
  ----------       ----      -----------
  201.2.10.11      00020     ALL
  201.2.10.11      00021     ALL
  201.2.10.12      00020     193.9.200.2
  201.2.10.12      00021     193.9.200.2
```

2. Run the display command `d tcpip,tcpcs,net,config` on the distributing stack and all target stacks to confirm that the correct IPCONFIG options have been specified.
   a. DATAGRAMFWD must be specified on the distributing stack in order for datagrams to be forwarded to target stacks.
   b. SYSPLEXROUTING must be specified on the distributor and all target stacks to get WLM based distribution.
   c. Verify that DYNAMICXCF was specified on the distributor and all target stacks.

   Following is the output of this command for the distributing TCP/IP.

```
d tcpip,tcpcs,net,config

EZZ2500I NETSTAT CS V2R10 TCPCS 568
TCP CONFIGURATION TABLE:
 .
 .
IP CONFIGURATION TABLE:
FORWARDING: YES   TIMETOLIVE: 00064  RSMTIMEOUT:  00060
FIREWALL:   00000  ARPTIMEOUT: 01200  MAXRSMSIZE:  65535
IGREDIRECT: 00001  SYSPLXROUT: 00001 DOUBLENOP:   00000
STOPCLAWER: 00000  SOURCEVIPA: 00000  VARSUBNET:   00001
MULTIPATH:  NO     PATHMTUDSC: 00000
DYNAMICXCF: 00001
   IPADDR: 193.9.200.1      SUBNET: 255.255.255.0  METRIC: 14
```

   Run the display command `D WLM,SYSTEM=MVS004` on the distributing and all targets stacks to confirm that WLM is active and in GOAL mode.

```
D WLM,SYSTEM=MVS004

IWM025I  08.06.58  WLM DISPLAY 610
  ACTIVE WORKLOAD MANAGEMENT SERVICE POLICY NAME: CS390SVT
  ACTIVATED: 2000/01/11  AT: 19:09:58  BY: xxxxxxx   FROM: yyyyyyyy
  DESCRIPTION: WLM POLICY FOR CS/390 SVT Group.
  RELATED SERVICE DEFINITION NAME: CS390WLM
  INSTALLED: 2000/01/11  AT: 19:09:51  BY: xxxxxxx   FROM: yyyyyyyy
  WLM VERSION LEVEL:       LEVEL008
  WLM FUNCTIONALITY LEVEL: LEVEL006
  WLM CDS FORMAT LEVEL:    FORMAT 3
  STRUCTURE SYSZWLM_WORKUNIT STATUS: DISCONNECTED
  *SYSNAME*  *MODE*  *POLICY*  *WORKLOAD MANAGEMENT STATUS*
  MVS004     GOAL    CS390SVT  ACTIVE
```

3. Run the display command `d tcpip,tcpcs,net,vipadyn` on the distributing stack to verify the DVIPA status is ACTIVE and the distribution status is DIST or DIST/DEST.

```
d tcpip,tcpcs,net,vipadyn

EZZ2500I NETSTAT CS V2R10 TCPCS 570
```

```
IP ADDRESS      ADDRESSMASK     STATUS    ORIGINATION     DISTSTAT
201.2.10.11     255.255.255.240 ACTIVE    VIPADEFINE      DIST/DEST
201.2.10.12     255.255.255.240 ACTIVE    VIPADEFINE      DIST
```

Then run display command `d tcpip,tcpcs2,net,vipadyn` on the target stacks to verify it has activated the distributing DVIPA and has it marked as a DEST. In this case TCPCS2 has marked the distributing DVIPAs as DEST and that TCPCS2 is a backup stack for these two DVIPAs (status and origination show backup).

```
d tcpip,tcpcs2,net,vipadyn

EZZ2500I NETSTAT CS V2R10 TCPCS2 891
IP ADDRESS      ADDRESSMASK     STATUS    ORIGINATION     DISTSTAT
201.2.10.11     255.255.255.240 BACKUP    VIPABACKUP      DEST
201.2.10.12     255.255.255.240 BACKUP    VIPABACKUP      DEST
201.2.10.13     255.255.255.192 ACTIVE    VIPADEFINE
201.2.10.21     255.255.255.192 BACKUP    VIPABACKUP
201.2.10.22     255.255.255.192 BACKUP    VIPABACKUP
```

4. Run the display command `d tcpip,tcpcs,sysplex,vipad` 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. The following display shows:
   - Two distributed DVIPAs in the sysplex, 201.2.10.11 and 201.2.10.12
   - Which TCP/IPs have been made targets, DIST field = DEST
   - The status of all other DVIPAs in this sysplex

```
d tcpip,tcpcs,sysplex,vipad

EZZ8260I SYSPLEX CS V2R10 513
VIPA DYNAMIC DISPLAY FROM TCPCS   AT MVS004
IPADDR: 201.2.10.11  LINKNAME: VIPLC9020A0B
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      BOTH
  TCPCS2   MVS004   BACKUP 100                                  DEST
  TCPCS3   MVS004   BACKUP 010                                  DEST
IPADDR: 201.2.10.12  LINKNAME: VIPLC9020A0C
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS    MVS004   ACTIVE      255.255.255.240 201.2.10.0      DIST
  TCPCS2   MVS004   BACKUP 075                                  DEST
  TCPCS3   MVS004   BACKUP 010
IPADDR: 201.2.10.13
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS2   MVS004   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS    MVS004   BACKUP 100
  TCPCS3   MVS004   BACKUP 010
IPADDR: 201.2.10.21
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
  TCPCS3   MVS004   ACTIVE      255.255.255.192 201.2.10.0
  TCPCS2   MVS004   BACKUP 100
  TCPCS    MVS004   BACKUP 080
IPADDR: 201.2.10.22
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK ADDRESS MASK    NETWORK PREFIX  DIST
  -------- -------- ------ ---- --------------- --------------- ----
```

```
   TCPCS3   MVS004   ACTIVE      255.255.255.192 201.2.10.0
   TCPCS    MVS004   BACKUP 080
   TCPCS2   MVS004   BACKUP 075
 15 OF 15 RECORDS DISPLAYED
```

5. Run the display command `d tcpip,tcpcs,net,vdpt` 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 current WLM/QOS distributing values for each target stack (each DESTXCF ADDR). See the*OS/390 IBM Communications Server: IP User's Guide* for more information.

   This display will only show target stacks that are currently up and have joined the sysplex. If there are fewer entries than what showed up in the display command `d tcpip,,net,vipadcfg`, then the missing entries may be for target stacks that are not yet up, or are up but did not specify the expected DynamicXCF address.

```
d tcpip,tcpcs,net,vdpt detail

EZZ2500I NETSTAT CS V2R10 TCPCS 439
DYNAMIC VIPA DISTRIBUTION PORT TABLE:
DEST IPADDR     DPORT DESTXCF ADDR    RDY  TOTALCONN
-----------     ----- ------------    --- ---------
201.2.10.11     00020 193.9.200.1     000 0000003561
  WLM: 32  W/Q: 04
201.2.10.11     00020 193.9.200.2     000 0000003500
  WLM: 16  W/Q: 08
201.2.10.11     00020 193.9.200.3     000 0000003700
  WLM: 00  W/Q: 00
201.2.10.11     00021 193.9.200.1     001 0000000499
  WLM: 32  W/Q: 16
201.2.10.11     00021 193.9.200.2     001 0000000450
  WLM: 16  W/Q: 08
201.2.10.11     00021 193.9.200.3     001 0000000415
  WLM: 00  W/Q: 00
201.2.10.12     00020 193.9.200.2     000 0000000239
  WLM: 16  W/Q: 02
201.2.10.12     00021 193.9.200.2     001 0000000059
  WLM: 16  W/Q: 02

8 OF 8 RECORDS DISPLAYED
```

   Interrogate the READY (RDY) count fields. If the count is 0, then no server application has been activated or no server is LISTENing on DPORT on the target stack identified by the DESTXCF ADDR. For servers that use more than one port, the RDY value reflects the port the LISTEN is performed on. For example, for FTP, the control connection port (port 21) is where you would expect to see a RDY count greater than 0. If the ready count is not what you expect, proceed to Step 7 to verify whether any server is LISTENing on the DPORT on the target stack.

   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, or the WLM 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 (distributor and all targets) have SYSPLEXROUTING specified (see Step 2 above for instructions for verifying this). Also, verify that WLM is configured and active in GOAL mode on all participating stacks (see Step 2 above).

- If the WLM configuration appears correct, consider whether the unexpected distribution results may be due to the current workload on the target stacks.
- If the unexpected distribution results have not yet been explained and Sysplex Distributor Performance Policies have been defined using PAGENT, consider whether the distribution may be due to network performance (TCP retransmissions and time-outs).
- If Sysplex Distributor Routing Policies have been defined using PAGENT, consider whether the definition of that policy is affecting the connection distribution. See “Diagnosing Policy Agent Problems” on page 379 for more information.

6. Run the display command `d tcpip,tcpcs,net,vcrt` on the distributing stack to see if there are any active connections being routed by the distributor.

   If this table is empty, then connection requests may not be getting to 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 7.

```
d tcpip,tcpcs,net,vcrt

 EZZ2500I NETSTAT CS V2R10 TCPCS 363
 DYNAMIC VIPA CONNECTION ROUTING TABLE:
 DEST IPADDR      DPORT  SRC IPADDR       SPORT  DESTXCF ADDR
 -----------      -----  ----------       -----  ------------
 201.2.10.11      00021  193.9.200.5      01029  193.9.200.1
 201.2.10.11      00021  193.9.200.8      01050  193.9.200.2
 201.2.10.11      00021  193.9.200.11     01079  193.9.200.3
 201.2.10.12      00021  193.9.200.9      01030  193.9.200.2
```

7. Go to the target stack(s) represented by the DESTXCF ADDR field in the VCRT or VDPT display and run the display command `d tcpip,tcpcs3,net,allconn,ipa=201.2.10.11` to see the connection(s) on the target stack.

```
d tcpip,tcpcs3,net,allconn,ipa=201.2.10.11

01.45.56  EZZ2500I NETSTAT CS V2R10 TCPCS3 737
USER ID  CONN       LOCAL SOCKET      FOREIGN SOCKET      STATE
FTPD1    00000034   201.2.10.11..21   193.9.200.11..1079  ESTBLSH
```

   Note: For a variety of reasons, the VCRT and ALLCONN displays may not match exactly. For example with short lived connections such as Web connections, an entry may 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 may reject these connection requests and therefore cause a temporary mismatch in the two displays.

Documentation for the IBM Support Center

Persistent error conditions, in most cases, 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 prior to contacting the IBM Support Center:

- *hlq*.PROFILE.TCPIP
- *hlq*.TCPIP.DATA
- Output from onetstat commands
- Output from onetstat -r or oping traces
- Network diagram or layout

- Error messages received (refer to *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)* for information about messages)
- OMPROUTE component trace in the CTRACE managed data set and OMPROUTE trace in stdout

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 ITRACE. For short descriptions of other key tracing procedures, such as displaying trace status, see “Appendix A. Collecting Component Trace Data” on page 513.

Component Trace

You typically use component trace when recreating 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.

CS for OS/390 provides component trace support for TCP/IP stacks, OMPROUTE, and for end-user socket programs written with APIs that use either the TCP/IP Macro API or the Call Instruction API. See “Component Trace for TCP/IP Stacks” and “TCP/IP Services Component Trace for OMPROUTE” on page 447. For detailed information, refer to the following books:

- *OS/390 MVS Diagnosis: Tools and Service Aids* for information about Component Trace Procedures
- *OS/390 MVS Initialization and Tuning Reference* for information about the Component Trace SYS1.PARMLIB member
- *OS/390 MVS System Commands* for information about commands
- *OS/390 MVS Authorized Assembler Services Guide* for procedures and return codes for component trace macros

Component Trace for TCP/IP Stacks

Component trace for TCP/IP stacks traces individual TCP/IP components (such as STORAGE, INTERNET, and so forth) and writes the information to a data set. 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 runtime by any of the following methods:

- JOBNAME
- Address space identifiers (ASID)
- Trace option
- IP address
- Port number

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 (see Figure 4 on page 49). For a description of trace options available in the CTIEZB00, see Table 12 on page 52.

Note: Besides specifying the desired TCP/IP traces, you can also change the component trace buffer size. The buffer size can be changed only at TCP/IP initialization.

```
/*******************************************************************/
/*                                                                 */
/*  Communications Server for OS/390                               */
/*  SMP/E Distribution Name: CTIEZB00                              */
/*                                                                 */
/*  MEMBER:  CTIEZB00                                              */
/*                                                                 */
/*     COPYRIGHT = Licensed Materials - Program Property of IBM.   */
/*                 This product contains "Restricted Materials of IBM"*/
/*                 5647-A01 (C) Copyright IBM Corp. 1996, 2000     */
/*                 All rights reserved.                            */
/*                 US Government Users Restricted Rights -         */
/*                 Use, duplication or disclosure restricted       */
/*                 by GSA ADP Schedule Contract with IBM Corp.     */
/*                 See IBM Copyright Instructions                  */
/*                                                                 */
/*     STATUS =  CSV2R10                                           */
/*                                                                 */
/*  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 members only lists those TRACEOPTS */
/*                 value specific to TCP/IP.  For a complete list  */
/*                 of TRACEOPTS keywords and their values see      */
/*                 OS/390 MVS INITIALIZATION AND TUNING REFERENCE. */
/*                                                                 */
/* $MAC(CTIEZB00) PROD(TCPIP): Component Trace SYS1.PARMLIB member */
/*                                                                 */
/*  CHANGE-ACTIVITY =                                              */
/*  CFD List:                                                      */
/*                                                                 */
/*  $L0=D005509  HTCP330 960619 MWS: RAS DCR - Trace Enhancements  */
/*  $D1=D0077    HTCP340 970916 DWJ: RAS DCR - Update options      */
/*  $P1=MV15775  TCPV3R4 971010 DYY: Remove HPNSAPPL, HPNSTCP opts */
/*  $D2=D0109    HTCP340 971026 KDJ: OS/390 Copyright              */
/*  $A1=PQ16720  HTCP350 980520 MWS: Add FIREWALL option (MV17568) */
/*  $G1=D270.14  CSV2R7  980521 HMS: Sysplex Sockets               */
/*  $G2=D270     CSV2R7  980909 MWS: Add jobname, asid, and wtr    */
/*  $H1=D280.9   CSV2R8  981203 MWS: Add IpAddr and Port keywords  */
/*  $H2=MV18932  CSV2R8  990203 MWS: Update default size to 8M     */
/*  $J1=D310.19  CSV2R10 990503 LAN: Added TRACE ROUTE             */
/*  $J2=D310.21  CSV2R10 990806 SAH: Trace enhancements            */
/*  $J3=D310.22  CSV2R10 990810 BDM: Socket API Trace              */
/*  $J4=MV19951  CSV2R10 990907 SAH: Add new trace group names     */
/*  $J5=MV20658  CSV2R10 991213 JPW: Fix typos                     */
/*  $J6=MV21237  CSV2R10 000227 BDM: Fix prolog errors             */
/*  $J7=MV21293  CSV2R10 000306 BDM: Remove ROUTE from ALL         */
/*                                                                 */
/*  End CFD List:                                                  */
/*******************************************************************/
```

Figure 4. SYS1.PARMLIB Member CTIEZB00 (Part 1 of 3)

```
TRACEOPTS
/* ---------------------------------------------------------------- */
/*   ON OR OFF: PICK 1                                              */
/* ---------------------------------------------------------------- */
          ON
/*        OFF                                                       */
/* ---------------------------------------------------------------- */
/*   BUFSIZE: A VALUE IN RANGE 128K 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, STORAGE, and TIMER */
/*   CSOCKET = PFS + SOCKET                                         */
/*   DLC     = CLAW + INTERNET + LCS + VTAM + VTAMDATA              */
/*   IN      = CONFIG + INIT + IOCTL + OPCMDS + OPMSGS              */
/*   LATCH   = SERIAL                                               */
/*   MINIMUM = INIT + OPCMDS + OPMSGS                               */
/*   OETCP   = ENGINE + PFS + QUEUE + TCP                           */
/*   OEUDP   = ENGINE + PFS + QUEUE + UDP                           */
/*   PING    = ARP + ICMP + RAW                                     */
/*   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                  */
/*                                                                  */
/*   OPTIONS: NAMES OF FUNCTIONS OR GROUPS TO BE TRACED:            */
/*                                                                  */
/* ---------------------------------------------------------------- */
```

Figure 4. SYS1.PARMLIB Member CTIEZB00 (Part 2 of 3)

```
/*        OPTIONS(            */
/*                'ALL     '  */
/*               ,'ACCESS  '  */
/*               ,'AFP     '  */
/*               ,'ARP     '  */
/*               ,'CLAW    '  */
/*               ,'CONFIG  '  */
/*               ,'CSOCKET '  */
/*               ,'DLC     '  */
/*               ,'ENGINE  '  */
/*               ,'FIREWALL'  */
/*               ,'ICMP    '  */
/*               ,'IN      '  */
/*               ,'INIT    '  */
/*               ,'INTERNET'  */
/*               ,'IOCTL   '  */
/*               ,'IPADDR(nnn.nnn.nnn.nnn/mmm.mmm.mmm)' */
/*               ,'LATCH   '  */
/*               ,'LCS     '  */
/*               ,'MESSAGE '  */
/*               ,'MINIMUM '  */
/*               ,'NONE    '  */
/*               ,'OETCP   '  */
/*               ,'OEUDP   '  */
/*               ,'OPCMDS  '  */
/*               ,'OPMSGS  '  */
/*               ,'PASAPI  '  */
/*               ,'PFS     '  */
/*               ,'PING    '  */
/*               ,'PORT(ppppp,ooooo,rrrrr,ttttt) ' */
/*               ,'QUEUE   '  */
/*               ,'RAW     '  */
/*               ,'ROUTE   '  */
/*               ,'RW      '  */
/*               ,'SERIAL  '  */
/*               ,'SMTP    '  */
/*               ,'SNMP    '  */
/*               ,'SOCKAPI '  */
/*               ,'SOCKET  '  */
/*               ,'STORAGE '  */
/*               ,'SYSTEM  '  */
/*               ,'TC      '  */
/*               ,'TCP     '  */
/*               ,'TELNET  '  */
/*               ,'TELNVTAM'  */
/*               ,'TIMER   '  */
/*               ,'TN      '  */
/*               ,'UD      '  */
/*               ,'UDP     '  */
/*               ,'VTAM    '  */
/*               ,'VTAMDATA'  */
/*               ,'WORKUNIT'  */
/*               ,'XCF     '  */
/*                )           */
```

Figure 4. SYS1.PARMLIB Member CTIEZB00 (Part 3 of 3)

Table 12 on page 52 describes the available trace options and groups. A group turns on multiple trace options. The group name identifies traces that should be on for a specific problem area. Trace groups provide a way to collect trace data by problem type.

Table 12. Trace Options

Trace Event	Description
ALL	All types of records except ROUTE, SERIAL, STORAGE, and TIMER. **Slow Performance:** Using this option will slow performance considerably, so use with caution.
ACCESS	Trace creation, modification, and manipulation of the Network Access tree, along with results of all Network Access queries.
AFP	Turn 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. **Note:** The information provided differs depending on the type of device. See *OS/390 IBM Communications Server: IP Configuration Guide* for information on what is supported for each device.
CLAW	Shows all control flows for a CLAW device.
CONFIG	Turn on trace for configuration updates.
CSOCKET	Turn on the following trace options: • PFS • SOCKET
DLC	Turn on the following trace options: • CLAW • INTERNET • LCS • VTAM • VTAMDATA
ENGINE	Turn on trace for stream head management. **Note:** The ENGINE and QUEUE options are aliases. If you turn one on, you turn on all related options, and if you turn one off, you turn off all related options.These alias options are only for recording the trace. When formatting the trace, these options can be filtered separately.
FIREWALL	Turn on trace for firewall events.
ICMP	Turn on trace for the ICMP protocol.
IN	Turn on the following trace options: • CONFIG • INIT • IOCTL • OPCMDS • OPMSGS
INIT	Turn 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.
INTERNET	Turn on trace for Internet Protocol layer.
IOCTL	Turn on trace for IOCTL processing.

Table 12. Trace Options (continued)

Trace Event	Description
IPADDR=(list)	Turn on trace by IP address.
LATCH	Turn on the following trace option: • SERIAL
LCS	Shows all control flows for an LCS device.
MESSAGE	Turn on trace for message triple management.
MINIMUM	Turn on the following trace options: • INIT • OPCMDS • OPMSGS
NONE	Turn off all traces but exception traces, which always stay on.
OETCP	Turn on the following trace options: • ENGINE • PFS • QUEUE • TCP
OEUDP	Turn on the following trace options: • ENGINE • PFS • QUEUE • UDP
OPCMDS	Turn on traces of operator commands. **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.
OPMSGS	Turn on message trace for console messages. **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.
PASAPI	Turns on traces for transforms that handle Pascal APIs.
PFS	Turn on trace for the physical file system layer.
PING	Turn on the following trace options: • ARP • ICMP • RAW
PORT=(list)	Turn on trace by port number.
QUEUE	Turn on trace for stream queue management. **Note:** 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 12. Trace Options (continued)

Trace Event	Description
RAW	Turn on trace for the RAW transport protocol.
ROUTE	Trace manipulation of IP Routing Tree
RW	Turn on the following trace options: • ENGINE • PFS • QUEUE • RAW • SOCKET
SERIAL	Turn on trace for lock obtain and release.
SMTP	Turn on the following trace options: • ENGINE • IOCTL • PASAPI • PFS • QUEUE • SOCKET • TCP
SNMP	Turn on trace for SNMP SET requests.
SOCKAPI	Trace Macro and Call Instruction API calls (see “Socket API Traces” on page 69)
SOCKET	Turn on trace for the Sockets API layer.
STORAGE	Turn on trace for storage obtain and release.
SYSTEM	Turn on the following trace options: • INIT • OPCMDS • OPMSGS • SERIAL • STORAGE • TIMER • WORKUNIT
TC	Turn on the following trace options: • ENGINE • PFS • QUEUE • SOCKET • TCP
TCP	Turn on trace for the TCP transport protocol.
TELNET	Turn on trace for TELNET events.
TELNVTAM (an alias for TELNET)	Turn on trace for TELNET events.
TIMER	Turn on trace for TCP timers.

Table 12. Trace Options (continued)

Trace Event	Description
TN	Turn on the following trace options: • PFS • TCP • TELNET • TELNVTAM
UD	Turn on the following trace options: • ENGINE • PFS • QUEUE • SOCKET • UDP
UDP	Turn on trace for UDP transport protocol.
VTAM	Shows all of the non data-path signaling occurring between IF and VTAM. Turn on this option; it will have no effect on mainline performance and will provide helpful information.
VTAMDATA	Shows data-path signaling between IF and VTAM, including a snapshot of media headers and some data. This option can slow performance, so use with caution.
WORKUNIT	Turn on trace for work unit scheduling.
XCF	Turn on trace for XCF events.

Specifying Trace Options After Initialization: After TCP/IP initialization, you must use the TRACE CT command to change the component trace options. All options except for the size of the TCP/IP component trace buffer can be changed using this command. 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. See "Appendix A. Collecting Component Trace Data" on page 513.

Filter TCPIP CTRACE by IP Address

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(192.48.32.37),PORT(80,33))
R 02,OPTIONS=(ENGINE,PFS),END
```

Trace records of type ENGINE or PFS, and for an IP address of 192.48.32.37 and a port number of 80 or 33 will be 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 sixteen). The trace record must match one of the values.

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.

There are five criteria for selecting trace records for recording: TYPE, JOBNAME, ASID, IPADDR and PORT. Each criterion may 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 will 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.

Note: There is an exception for IPADDR and PORT. Some trace records will not have a IP address or a port number. Therefore, the IP address will only be checked if it is nonzero and the port number will be 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
The list of IP address to be filtered. Up to sixteen IP addresses may be specified. The address is specified in dotted decimal format. Submasking may be indicated by using a slash followed by a dotted decimal submask. For example, 192.48.32/255.255.255.0 will allow addresses from 192.48.32.00 to 192.48.32.255. A trace record with a zero IP address is not subject to IP address filtering.

PORT The list of port numbers to be filtered. Up to sixteen port numbers may be specified. The port numbers, specified in decimal, are in the range 0–65535. A trace record with a zero port number is not subject to port number filtering.

Notes:

1. The equal sign (=) after the IPADDR and PORT keywords is optional.
2. The IPADDR and PORT keywords may be specified multiple times in an OPTIONS string. All the values will be saved.
3. All the values in the OPTIONS keyword must be specified in one trace command. The next trace command with an OPTIONS keyword will replace all the options specified.

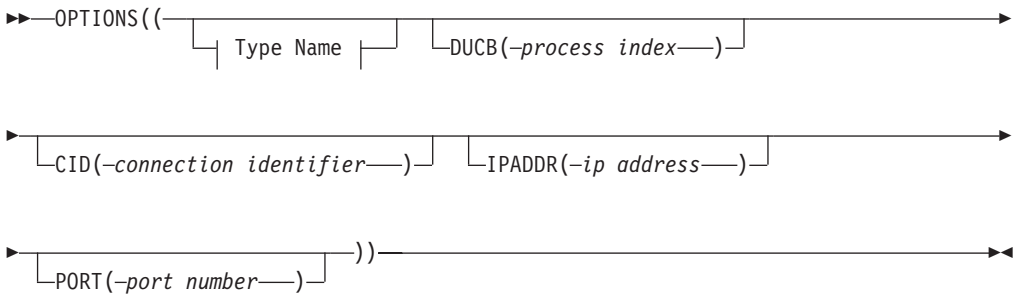
Formatting Trace Records for TCP/IP Stacks

You can format component trace records using IPCS panels or a combination of IPCS panels and the CTRACE command. For a description of the relevant IPCS panels, see “Appendix A. Collecting Component Trace Data” on page 513. For more information about other CTRACE options, refer to the *OS/390 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. For more information on the command and IPCS, refer to the *OS/390 MVS IPCS User's Guide*.



Type Name
The name of a trace type. Only records of these types will be formatted. For a list of types, see Table 12 on page 52.

DUCB A process index for the thread of execution. Up to sixteen 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 sixteen 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 sixteen addresses can be specified. These are in dotted decimal notation. For Example: 192.48.24.57. An IP address of '0' may be used for trace records that do not have an IP address. A subnet mask is indicated by a slash ('/') followed by a dotted decimal value of the subnet mask. For example: 192.48.24/255.255.255.0

PORT A port number. Up to sixteen port numbers may be specified. Please 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. A port number of 0 may be used for trace records that do not have a port number.

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

Figure 5 on page 58 contains the start 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 4 data areas. 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 jobname) 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 4 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 4 data areas in this example. The third one (″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 *OS/390 IBM Communications Server: IP and SNA Codes* for more information.
7. TCP/IP trailer and separator line with the record sequence number (hexadecimal 573E).

```
 COMPONENT TRACE FULL FORMAT
 COMP(SYSTCPIP)SUBNAME((TCPSVT))
 **** 11/03/1999
 SYSNAME   MNEMONIC  ENTRY ID    TIME STAMP     DESCRIPTION
 -------   --------  --------  --------------  -------------
 VIC142    PFS       60010018  14:57:59.207826   Socket IOCTL Exit
HASID..001E      PASID..000E      SASID..001E      USER...OMPROUTE
TCB....007E7A68  MODID..EZBPFIOC  REG14..161D86C0  DUCB...0000000C
CID....0000003A  PORT...    0     IPADDR.000.000.000.000
  ADDR...00000000  14D9EED0  LEN....000000A0  OSI
     +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....         |
=================================================================0000573E
```

Figure 5. Start of Component Trace Full Format

Additional Fields in CTRACE Output

The ERRNO line in Figure 5 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 6 on page 59 is an example. Note the additional fields (TcpState, TpiState, etc.) 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
  SrcIPAddr.197.11.106.1      SrcPort..1925
  DstIPAddr.197.11.106.1      DstPort..389
  FLAGS.....ACK
```

Figure 6. Component Trace Full Format Showing Character Interpretation of Fields

Component Trace for OMPROUTE

TCP/IP Services component trace is also available for use with the OMPROUTE application. “TCP/IP Services Component Trace for OMPROUTE” on page 447.

Packet Trace

Packet trace is a diagnostic method for obtaining traces of IP packets flowing to and from a TCP/IP stack on a CS for OS/390 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 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.

Note: 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 *OS/390 IBM Communications Server: IP Configuration Reference* 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 7 illustrates the overall control and data flow in the IP packet tracing facility.

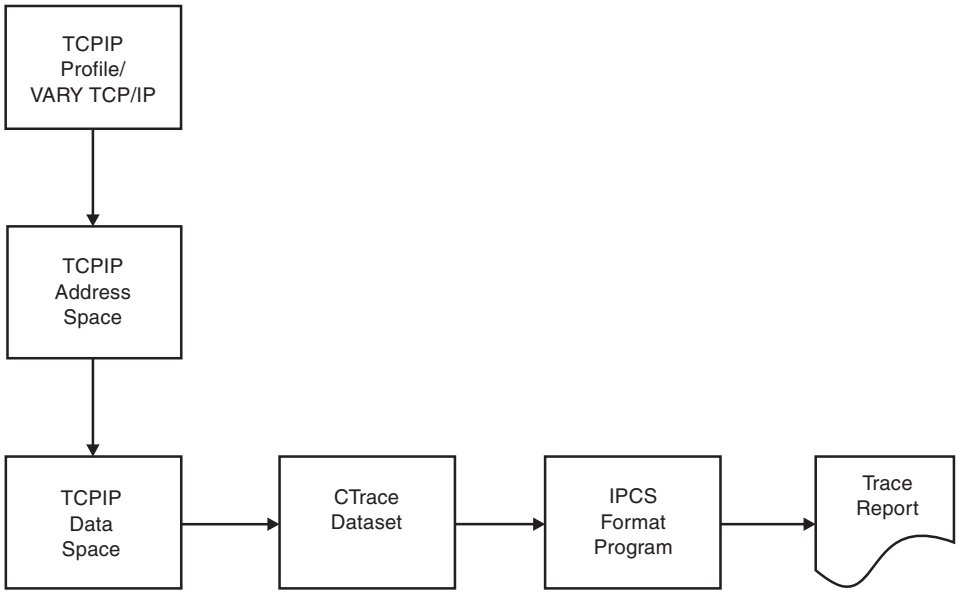


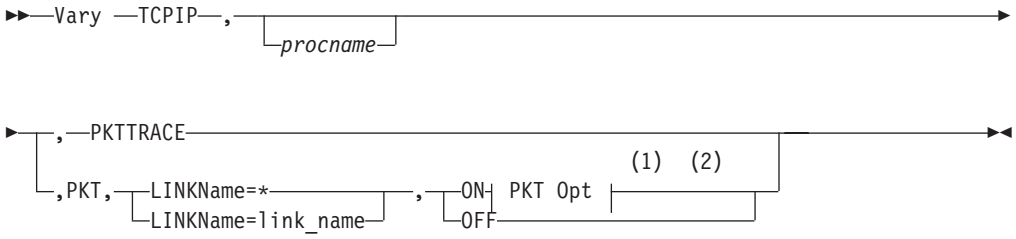
Figure 7. Control and Data Flow in the IP Packet Tracing Facility

Supported Devices

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

Packet Trace Format

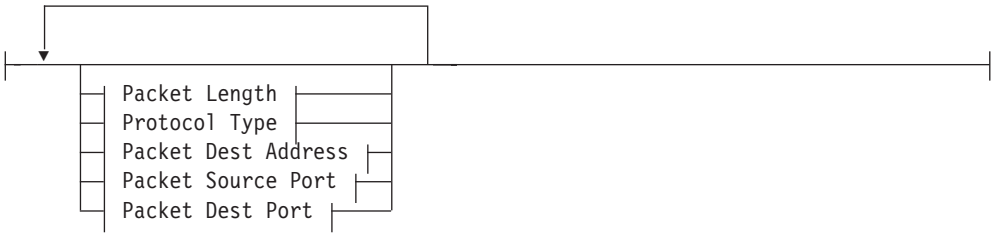
Use the VARY TCPIP PKTTRACE command to set up tracing from an operator console.



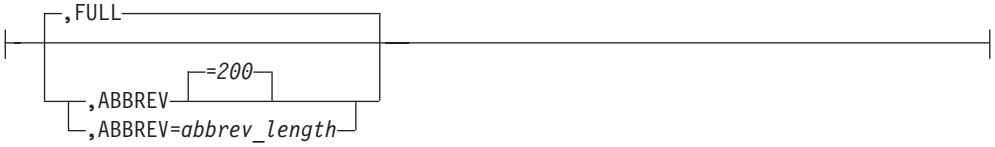
Notes:

1. Each option can be specified only once. The order of options is not important.
2. The MVS TRACE command must also be issued for component SYSTCPDA to activate the packet trace. See “Appendix A. Collecting Component Trace Data” on page 513 for details.

PKT Opt:

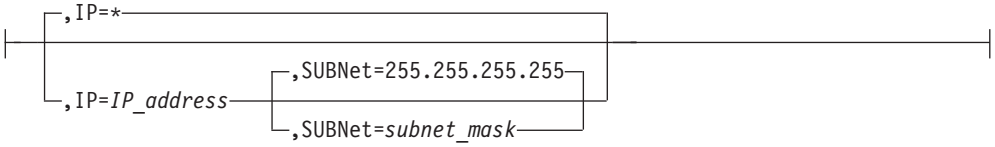


Packet Length:



Protocol Type:

Packet Dest Address:



Packet Source Port:

Packet Dest Port:

procname
Specifies the name of the TCP/IP address space to query for information.

PKTTRACE
Requests PKTTRACE information.

PKT Opt

ABBREV
Specifies that a truncated portion of the IP packet is to be traced. You can specify a length in the range 1—65535 or use the default of 200. The ABBREV parameter can be used to reduce the volume of data stored in the trace file.

DESTport
Specifies a port number that will be compared with the destination port of inbound and outbound packets. The port number is an integer in the range 1—65535. If the destination port of a packet is the same as the specified port number, the packet will be traced. This comparison is only performed for packets using either the TCP or UDP protocol; packets using other protocols are not traced. If the DESTPORT parameter is omitted, there is no checking of the destination port of packets. If an asterisk (*) is specified, packets of any protocol and any source port will be traced.

IP Specifies an IP address that will be compared with both the source and destination addresses of inbound and outbound packets. If either the source or destination address of a packet matches the specified IP address, the packet will be traced. The IP address must be specified in dotted decimal notation. If the IP option is omitted, or an asterisk (*) is specified, all IP addresses will be traced.

LINKName
Specifies the name of the link defined in the preceding LINK statement. If the LINKName parameter is omitted or an asterisk (*) is specified, the PKTTRACE parameters will apply to all links prior to this statement.

To facilitate defining packet tracing when many links are involved, use the PKTTRACE statement with the LINKName=* option to define packet tracing characteristics for the majority of the links. Then use individual PKTTRACE statements with specific LINKName parameters for each link that must be defined differently from the majority.

OFF
Disables packet tracing for the links specified and removes the characteristics defining how they should be traced.

If LINKName=* and all other parameters are defaults, all trace structures are deactivated and removed from all existing links.

If LINKName=* and PROT=UDP, all trace structures for all resources are analyzed; any matches are removed. If no trace structures remain, trace is deactivated for that resource.

If LINKName=*link_name* and there are no other parameters, all trace structures for *link_name* are deactivated and removed.

If LINKName=*link_name*,IPADR=127.0.0.1, that particular trace structure is removed if it is found. If there is only one trace structure, then that structure is removed and trace is deactivated for that resource.

ON
Turns on packet tracing, clears all settings previously defined and refreshes just the default settings.

If you use LINKName=* and all other parameters are defaults, even if the defaults are specified, the command results replaces any existing trace structures for all existing links.

If you use LINKName=*link_name* and another nondefault parameter, the command results are added to any existing trace structures. However, if the existing trace structure for *link_name* is all defaults, the existing trace structure will be discarded.

PROT
Specifies the protocol type to be traced. This can be specified as one of the literals TCP, UDP, or ICMP, or as a number in the range 0—255 (ICMP=1, TCP=6, UDP=17, and RAW=255). If the PROT parameter is omitted or an asterisk (*) is specified, packets of any protocol will be traced.

SRCPort
Specifies a port number that will be compared with the source port of inbound and outbound packets. The port number is an integer in the range 1—65535. If the source port of a packet is the same as the specified port number, the packet will be traced. This comparison is only performed for packets using either the TCP or UDP protocol; packets using other protocols are not traced. If the SRCPORT parameter is omitted, there is no checking of the source port of packets. If an asterisk (*) is specified, packets of any protocol and any source port will be traced.

SUBNet
Specifies a subnet mask that applies to the host and network portions of the IP address specified on the accompanying IP parameter. The subnet mask must be specified in dotted decimal notation and must be specified in conjunction with the IP parameter.

Starting Packet Trace

To start packet trace, use the following command:

```
V TCPIP,,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* are 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. See “Appendix A. Collecting Component Trace Data” on page 513.
The MVS TRACE command must also be issued for component SYSTCPDA to activate the packet trace.

Once you have started packet trace, you can display the status using the onetstat command, as shown in the following example:

```
# onetstat -p TCPCS -d
MVS TCP/IP onetstat IPA      TCPIP Name: TCPCS          12:34:56

  DevName: LOOPBACK          DevType: LOOPBACK       DevNum: 0000
```

```
LnkName: LOOPBACK          LnkType: LOOPBACK       Status: Ready
  NetNum: 1   QueSize: 0  ByteIn:  0000000304    ByteOut: 0000000304
BSD Routing Parameters:
  MTU Size: 00000           Metric: 00
  DestAddr: 8.67.113.22        SubnetMask: 255.255.0.0
Packet Trace Setting:
  Protocol: *        TrRecCnt: 00000000    PckLength: FULL
  SrcPort:   *       DestPort:  *
  IpAddress: *                  Subnet:    ipaddressmask
```

In this example, the packet length (PckLength) is FULL.

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

Modifying Options with Vary

After starting 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,,pkt,abbrev
```

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

```
# onetstat -p TCPCS -d
MVS TCP/IP onetstat CS V2R10  TCPIP Name: TCPCS          12:34:56

  DevName: LOOPBACK          DevType: LOOPBACK       DevNum: 0000
    LnkName: LOOPBACK          LnkType: LOOPBACK       Status: Ready
      NetNum: 1   QueSize: 0  ByteIn:  0000000304    ByteOut: 0000000304
    BSD Routing Parameters:
      MTU Size: 00000           Metric: 00
      DestAddr: 8.67.113.22        SubnetMask: 255.255.0.0
    Packet Trace Setting:
      Protocol: *        TrRecCnt: 00000000    PckLength: 00200
      SrcPort:   *       DestPort:  *
      IpAddress: *                  Subnet:    ipaddressmask
```

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

Socket Data Trace

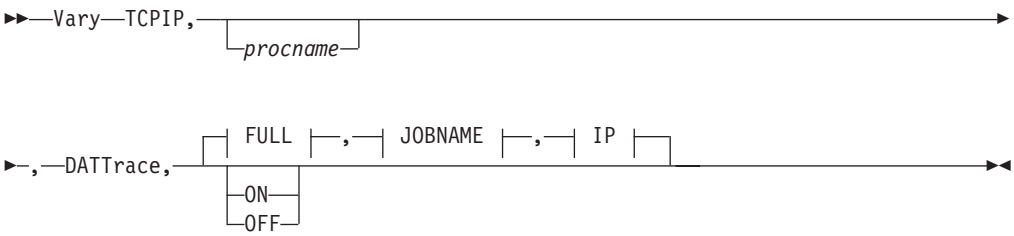
You can use the DATTRACE command to trace socket data (transforms) into and out of the physical file structure (PFS). DATTRACE works with the following Application Programming Interfaces (APIs):

- REXX
- C-sockets
- IMS®
- CICS®
- Native OS/390 UNIX

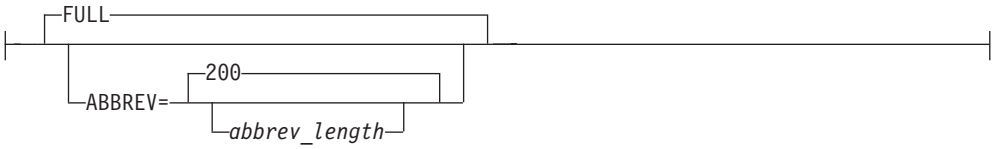
DATTRACE has the following format.

Notes:

1. Be sure to enter the DATTRACE command before the socket is opened.
2. The MVS TRACE command must also be issued for component SYSTCPDA to activate the data trace. See “Appendix A. Collecting Component Trace Data” on page 513 for details.

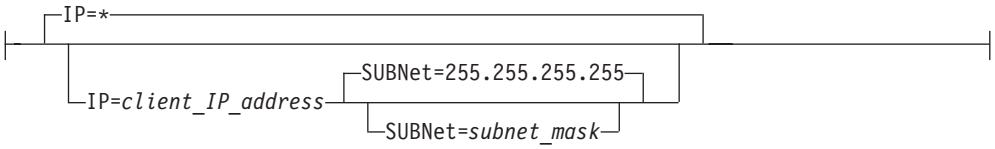


FULL:



JOBNAME:

IP:



Following are descriptions of the DATTRACE parameters:

ON
Turns on socket data tracing, clears all settings previously defined and refreshes just the default settings.

OFF
Turns off socket data tracing.

ABBREV
Specifies that a truncated portion of the IP packet is to be traced. You can specify a length in the range 1—65535 or use the default of 200. The ABBREV parameter can be used to reduce the volume of data stored in the trace file.

FULL
Specifies that the entire IP packet is to be traced

JOBNAME
Specifies the name of the TCP/IP address space to query for information

IP Specifies an IP address that will be compared with both the source and destination addresses of inbound and outbound packets. If either the source or

destination address of a packet matches the specified IP address, the packet will be traced. The IP address must be specified in dotted decimal notation. If the IP option is omitted, or an asterisk (*) is specified, then all IP addresses will be traced.

Note: IP address selection is not recommended for use with DATTRACE.

SUBNET
Specifies a SUBNET mask that applies to the host and network portions of the IP address specified on the accompanying IP parameter. The subnet mask must be specified in dotted decimal notation and must be specified in conjunction with the IP parameter.

Starting Data Trace

You can start data trace for all jobnames using the VARY command:

```
V TCPIP,,DAT
```

Security Note:

1. To use any VARY command, the user must be authorized in RACF. This replaces the old OBEY list authorization.
2. 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.
3. 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. See "Appendix A. Collecting Component Trace Data" on page 513.

Displaying Data Traces

You can use the `NETSTAT` or `onetstat` command to display data traces. Figure 8 shows a data trace for a single entry.

```
Data Trace Setting:
   Jobname: *                   TrRecCnt: 0000000000    Length: FULL
   IpAddr:  *                   SubNet: 255.255.255.255
```

Figure 8. Data Trace: Single Entry

Figure 9 shows a data trace for multiple entries.

```
Data Trace Setting:
   Jobname: MEGA4               TrRecCnt: 0000000000    Length: 200
   IpAddr:  127.0.0.3           SubNet: 255.255.255.255
   Jobname: *                   TrRecCnt: 0000000000    Length: 0
   IpAddr:  127.0.0.9           SubNet: 255.255.255.255
```

Figure 9. Data Trace: Multiple Entries

Formatting Packet Traces Using IPCS

You can format Packet Trace and Data Trace records using IPCS panels or a combination of IPCS panels and the CTRACE command. (For a description of the relevant IPCS panels, see "Appendix A. Collecting Component Trace Data" on page 513.) The following options are available for both Packet Trace and Data Trace:

PACKETTRACE
Select packet trace records

X25 Select X25 records

DATATRACE Select data trace records

LINK (linkname1 . . . linkname10)
One to ten link names

IPADDR (ipaddr1 . . . ipaddr10)
Match on either the source or destination address:

nnn.nnn.nnn.nnn
A dotted IP address. Each nnn in the IP address can be replaced with an asterisk (*) to indicate that the value can be ignored. For example, in 9.67.*.14, the third byte is ignored.

A Class A address

B Class B address

C Class C address

L Loopback address

TYPE (type1...type16)
Use one or more of the following device type keywords:

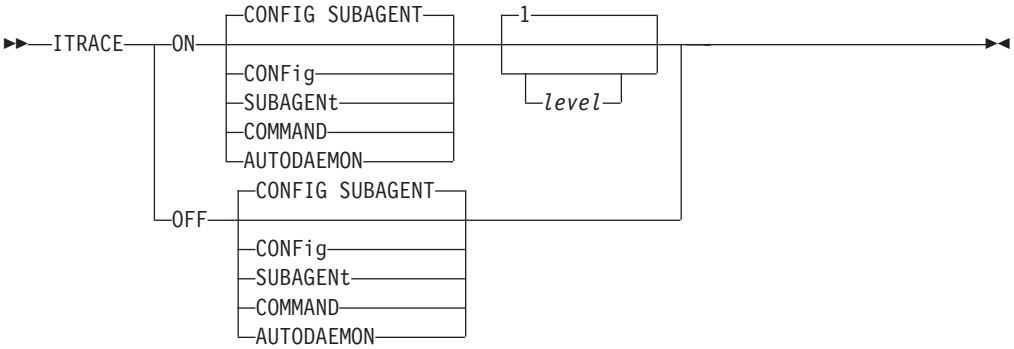
ATM
CDLC
CETIETHERNET
CETI8023
CETIETHERNET8023
CETITOKENRING
CLAWRS6000
CLAWETHERNET
CLAW8023
CLAWETHERNET8023
CLAWTOKENRING
CLAWFDDI
CTC
HIPPI
HYPERCHANNEL
IPAQENET
IUCV
LCSETHERNET
LCSETHERNET8023
LCSFDDI
LCSTOKENRING
LCS8023
LOOPBACK
MPC
OSAENET
OSAFDDI
SNALINK
SNALU0
SNALU62
X25NPSI
VIPA
X25

You can abbreviate the keyword as long as its meaning is clear; for example, you can use CLAWF instead of CLAWFDDI.

Note: Class A, B, C, and Loopback are methods of specifying an IP address and subnet for internet addresses.

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:

Level 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.

Socket API Traces

SOCKAPI is a new option for the TCP/IP CTRACE component SYSTCPIP. The SOCKAPI option 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 may issue. In previous versions of IBM Communications Server for OS/390, the TCP/IP CTRACE options already included a SOCKET trace option. However, 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.

OS/390 provides several socket APIs that can be used by applications. Figure 10 on page 70 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

environment, CICS transactions do not issue an INITAPI 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 INITAPI calls are created.

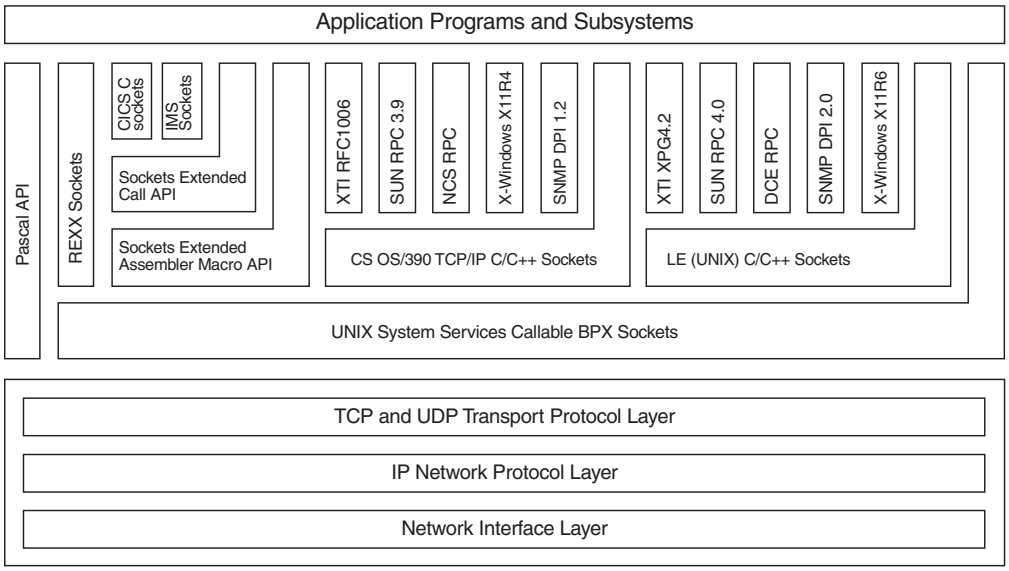


Figure 10. TCP/IP Networking API Relationship on OS/390

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. Some things to keep in mind when using CTRACE:

- Even though 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 security 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).

Optimally, 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 may be that the wrong IP address or port number was coded or used. Both the IP address and port number are formatting options.

Recommended options for optimally capturing the application data follow.

- **Trace only one application.** Use the jobname 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. (You will also get exception records. Exception records are always traced because they are considered unusual events that merit attention.)
- **Use an external writer.** The external writer is recommended:
  - To separate the SOCKAPI trace records from other internal data that exists in a dump (for security and other reasons)

  - To avoid interrupting processing with a dump of the trace data
  - To keep the buffer size from limiting the amount of trace data
  - To avoid increasing the buffer size (which requires restarting TCP/IP)
  - To 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 OS/390 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.

How to Collect the SOCKAPI Trace Option

The existing CTRACE facility for TCP/IP's SYSTCPIP component is used for the SOCKAPI trace option. Collecting the trace is described generally in "Component Trace" on page 47. This section describes how to collect the trace for use by application programmers.

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. The most important option is BUFSIZE because it cannot be changed after TCP/IP is started. All the other options can be changed while TCP/IP is active.

Note: The BUFSIZE is the size of the internal tracing buffer and is not relevant if you are using the external writer.

To get SOCKAPI records, you must specify either the SOCKAPI or ALL option. It is recommended that you use the ALL option only if necessary as the output can very large.

TCP/IP Start Procedure

The CTRACE PARMLIB member may 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. Following is an example of overriding the PARMLIB member name using the sample TCPIPROC start procedure.

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

TRACE Command

To start, modify, or stop the trace after TCP/IP has been started, use the MVS TRACE command. The TRACE command replaces all prior settings except the buffer size (which is set during TCP/IP initialization and cannot be modified). When modifying the options, be sure to specify the SOCKAPI option. Following are some examples showing how to start the trace. The SUB option is the subtrace name, which for TCP/IP is the jobname of the stack (usually this is the TCP/IP start procedure name). In these examples, the subtrace is TCPIPROC, the name of the sample procedure. In the examples, variable fields are in lower case.

- Example 1. Activating the trace with just the SOCKAPI option:

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

- Example 2. Specifying a PARMLIB member which contains the trace options:

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

To stop the trace, you can either use the TRACE CT,OFF command or you can re-issue TRACE CT,ON with different parameters. 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:

```
  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.
```

See *OS/390 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. 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 which specifies the job to run (the external writer) and the trace output data sets. There is an example in "Appendix A. Collecting Component Trace Data" on page 513. Also, see the *OS/390 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 jobname 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 the application programmer sees only the data for his/her application for security reasons.

The IP address and Port options can be used to filter the trace both when collecting the trace and when formatting the trace. In general, it is best to collect all the application records to avoid having to recreate the problem. Once 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 will get all the records for the problem client, and you'll also get 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 which match that fillter will be collected. No means that whether or not a filter is specified, all the calls will be collected (no filtering is done).

Table 13. IP Address/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/Port Filtering Effect on Different Types of Socket API Calls (continued)

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

Where Yes is indicated above, 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 and/or not connected, no filtering will take 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 will be available after an ACCEPT call, it will not be 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 non-zero port. If a BIND is issued for port 0 (meaning an ephemeral port will be assigned by TCP/IP), no filtering will take place for this BIND call.
3. If the socket is bound at the time of the CONNECT, local port filtering will be honored. Otherwise the CONNECT will not be subject to local port filtering.

Monitoring the Trace

Use the MVS command DISPLAY TRACE to check the trace options currently in effect. Here's 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 8 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 *OS/390 MVS System Commands*.

2–4 These are explained in the *OS/390 MVS System Messages, Vol 1 (ABA-ASA)* 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.1 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 will be 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. Note that 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
```

Note: 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.

If you use the external writer, see “External Writer” on page 72, which explains how to capture the trace in a data set.

How to Format 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 which will help to limit the amount of data formatted. In addition to the CTRACE options (start/stop time, etc.) 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
Can be used for SOCKAPI. Specify the IP address in dotted decimal format, with a subnet mask. 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 zero), specify zero for one of the IPADDR values. For example, `IPADDR(0,9.67.113/255.255.255.0)` will format all CTRACE records with an IP address of `000.000.000.000` and will format all CTRACE records with an IP address of `009.067.113.*`, where * is any number from 0 to 255.

PORT Can be used 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 see the TCP/IP CTRACE filtering options in effect. This will also indicate if any records were lost. See "Chapter 6. IPCS Subcommands for TCP/IP" on page 87 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/stop times). For a dump, this command will also show 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.

See the *OS/390 MVS IPCS User's Guide* for more information about CTRACE formatting. See *OS/390 MVS IPCS Commands* for more information about the CTRACE command.

How to Read and Interpret 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, and any API calls that fail before the trace point is reached. (An API call will fail if module EZBSOH03 cannot be located, if EZBSOH03 was unable to obtain storage, etc.). 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 79 below).

The following examples include:

- A SOCKAPI trace record
- Trace records for asynchronous applications
- GETHOSTBYADDR, GETHOSTBYNAME 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 which 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 may also be shown for reference. Parameters are only formatted if they were specified in the call (i.e., 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 *OS/390 IBM Communications Server: IP Application Programming Interface Guide* 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 will fit. Numeric parameter values are in decimal unless followed by a lowercase **x** indicating hexadecimal. Whenever possible, the values are interpreted (ERRNO, etc.) 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     LOCAL  IPADDR..9.67.113.58
 7.  REMOTE PORT..1034      REMOTE IPADDR..9.67.113.58
 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 jobname 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 *OS/390 IBM Communications Server: IP Application Programming Interface Guide*. 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

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 may actually complete synchronously, in which case the Async Complete trace record may 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 may fail synchronously or asynchronously. In this example, the WRITE call error was detected in the synchronous processing, so general purpose register 15 has a non-zero 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. See *OS/390 IBM 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     LOCAL  IPADDR..9.67.113.58
REMOTE PORT..1031      REMOTE 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         LOCAL  IPADDR..0.0.0.0
REMOTE PORT..11007     REMOTE 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 *OS/390 IBM 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     LOCAL  IPADDR..9.67.113.58
REMOTE PORT..0         REMOTE 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
```

GETHOSTBYADDR, GETHOSTBYNAME API Calls: The GETHOSTBYADDR and GETHOSTBYNAME API calls use the HOSTENT structure described in the calls in the *OS/390 IBM Communications Server: IP Application Programming Interface Guide*. As shown in the example, the HOSTENT address is shown on one line and

the contents of the HOSTENT structure are described on separate lines. There may 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.:  Loopback                                          Addr..00005F30
 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
```

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 may 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     LOCAL  IPADDR..9.67.113.58
 REMOTE PORT..0         REMOTE 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     LOCAL  IPADDR..9.67.113.58
 REMOTE PORT..0         REMOTE 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     LOCAL  IPADDR..9.67.113.58
REMOTE PORT..1032      REMOTE 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. (See the *OS/390 IBM Communications Server: IP Application Programming Interface Guide* 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 (AF_INET)
 RETCODE..:  0
```

Default TERMAPI: Usually, an application will end the connection between itself and TCP/IP by issuing the TERMAPI call. But sometimes, the connection will end for another reason, such as the application being cancelled. In this case, TCP/IP will issue 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 far right 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. Below the binary mask, the decimal socket numbers are listed in numerical order. This is a very handy way to check if the mask is coded as it is 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)
  127   96                                 11110111
   95   64     11111111 11111111 10111111 11111111
   63   32     00111011 11111111 11111111 11111101
   31    0     00000000 00000000 00000000 00001111
    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
```

SELECTEX: The SELECTEX call can contain a list of ECBs. The high-order bit on the SELECB adddress 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 8 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, when means the UE exit will be invoked in key 8.

```
================================================================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)
```

How to Correlate 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 will also have to 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 will allow the full data to be correlated.

Figure 11 on page 85 shows the data trace record corresponding to the READ Exit SOCKAPI trace entry in Figure 12 on page 85. 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 time, jobname, ASID, MVS TCB address, data length, port, and IP address:

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 jobname 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 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 11. Data trace record.

The packet trace, on the other hand, does not correlate well with the SOCKAPI

```
================================================================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     LOCAL  IPADDR..9.67.113.58
REMOTE PORT..1040      REMOTE 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 12. SOCKAPI trace record.

trace option. The packet trace will record data being sent/received between the TCP/IP stack and the network. The packet trace data has headers and the data may be segmented or packed.

Chapter 6. IPCS Subcommands for TCP/IP

The IPCS subcommands for TCP/IP are used to format data from IPCS system dumps. This chapter describes the commands (including description, syntax, parameters, and sample output), installation, and execution.

There are two types of subcommands.

- Many of the TCP/IP subcommands work on a specific stack. These subcommands are grouped under the TCPIPCS subcommand to share the TCP (to select the stack) and TITLE options.
- The remaining TCP/IP IPCS subcommands do not require a TCP/IP stack, so they are not under the TCPIPCS subcommand.

Table 14 lists all the IPCS subcommands.

Note: The TCP/IP IPCS commands are not supported for IPCS "active".

Table 14. TCP/IP IPCS Commands - showing the TCPIPCS commands first, followed by the general commands.

Command	Description	Alias
TCPIPCS ALL	**Equivalent to** TCPIPCS STATE TSEB TSDB TSDX DUAF CONFIG ROUTE SOCKET STREAM RAW TCB UDP LOCK TIMER STORAGE	
TCPIPCS API	Display control blocks for Sockets Extended Assembler Macro and Pascal APIs	
TCPIPCS CONFIG	Display device configuration information	TCPIPCS CNFG TCPIPCS CONF
TCPIPCS CONNECTION	Display active or all connections.	TCPIPCS CONN
TCPIPCS DETAIL	**Equivalent to** TCPIPCS TSEB TSDB TSDX DUAF	TCPIPCS CBS
TCPIPCS DU	**Equivalent to** TCPIPCS DUAF DUCB	
TCPIPCS DUAF	Summarize DUCBs	TCPIPCS DUCBS
TCPIPCS DUCB	Find and Format DUCBs	
TCPIPCS FIREWALL	Display information about Firewall filters and tunnels	
TCPIPCS FRCA	Display state information about FRCA connections and objects	
TCPIPCS HASH	Display TCP/IP data stored in hash tables	
TCPIPCS HEADER	Display dump Header info	TCPIPCS HDR
TCPIPCS HELP	Display syntax help for TCPIPCS command	TCPIPCS ?
TCPIPCS LOCK	Display locks	TCPIPCS LOCKSUM
TCPIPCS MAP	Display storage map	
TCPIPCS MTABLE	Display module table	
TCPIPCS POLICY	Display service policy data	

Table 14. TCP/IP IPCS Commands - showing the TCPIPCS commands first, followed by the general commands. (continued)

Command	Description	Alias
TCPIPCS PROFILE	Display TCP/IP configuration data in the format of a profile dataset	TCPIPCS PROF
TCPIPCS PROTOCOL	Invokes RAW, TCB, UDP	
TCPIPCS RAW	Display Raw control blocks	TCPIPCS MRCB TCPIPCS RAWSUM TCPIPCS RCB
TCPIPCS ROUTE	Display routing information	TCPIPCS RTE
TCPIPCS SOCKET	Display socket information	TCPIPCS SCB TCPIPCS SOCKSUM
TCPIPCS STATE	Display general stack information	TCPIPCS
TCPIPCS STORAGE	Display TCP/IP storage usage	TCPIPCS STOR
TCPIPCS STREAM	Display Streams information	TCPIPCS SKSH TCPIPCS STREAMS
TCPIPCS SUMMARY	**Equivalent to** TCPIPCS DUAF CONFIG SOCKET	
TCPIPCS TCB	Display TCP protocol control blocks	TCPIPCS MTCB TCPIPCS TCBSUM
TCPIPCS TELNET	Display Telnet information	
TCPIPCS TIMER	Display information about Timers	TCPIPCS TIMESUM
TCPIPCS TRACE	Display TCP/IP CTrace information	TCPIPCS TCA
TCPIPCS TREE	Display information about data stored in Patricia trees	TCPIPCS TREESUM
TCPIPCS TSDB	Format TSDB	
TCPIPCS TSDX	Format TSDX	
TCPIPCS TSEB	Format TSEB	
TCPIPCS UDP	Display UDP control blocks	TCPIPCS MUCB TCPIPCS UCB TCPIPCS UDPSUM
TCPIPCS VMCF	Display information about VMCF and IUCV users	
TCPIPCS XCF	Display information about XCF links and dynamic VIPA	
ERRNO	Interpret error numbers	
ICMPHDR	Format an ICMP header	
IPHDR	Format an IP header	
SETPRINT	Set destination so the IPCS subcommand output will be sent to a user ID and/or the printer	
SKMSG	Format a stream message	

Table 14. TCP/IP IPCS Commands - showing the TCPIPCS commands first, followed by the general commands. (continued)

Command	Description	Alias
TCPHDR	Format a TCP header	
TOD	Convert a S/390 64-bit time-of-day timestamp to a readable date and time	
UDPHDR	Format UDP header	

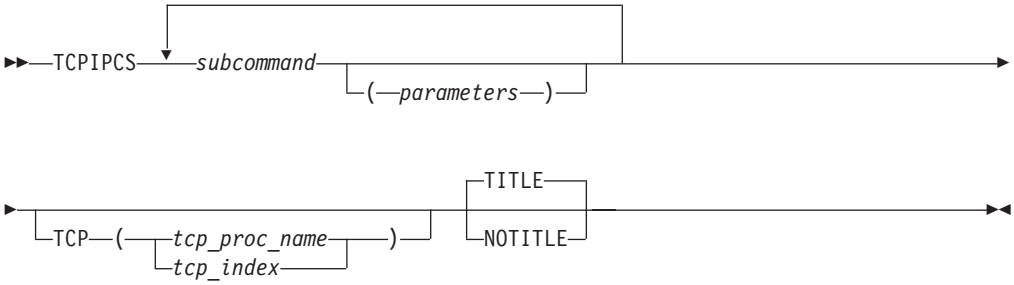
TCPIPCS

The following describes the TCPIPCS command.

Command Syntax

The command syntax for all TCPIPCS subcommands includes an option to specify the TCP stack and whether the title will be displayed.

Note: For an explanation about syntax diagrams, see "Appendix E. How to Read a Syntax Diagram" on page 547.



Parameters

The command syntax parameters for the TCPIPCS command is 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 place-holder if any keyword parameters will be specified. If two or more variable parameters are specified, they must be enclosed in parentheses.
- To distinguish among the variable parameters, a parameter is assumed to be one of the following:
  - An index or small number if it is 4 digits or less, begins with 0–9, 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 4 digits, begins with 0–9, 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, command name).
- Keyword parameters may 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 will be processed. 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 which matches the release of the TCPIPCS command. Once 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 may be selected. 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 will get a message indicating the stack is not the same version and release as the TCPIPCS command `Selected TCP/IP is not V2R10`.

tcp_proc_name

TCP/IP procedure name.

tcp_index

TCP/IP stack index (1–8).

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 dataset 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, and 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 handy when you are processing lots of commands on the same dump and don't care to see the title information repeated.

Note: If you specify multiple keywords from the set {TITLE, NOTITLE}, only the last one will be 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 available TCPIPCS Subcommands.

TCPIPCS API

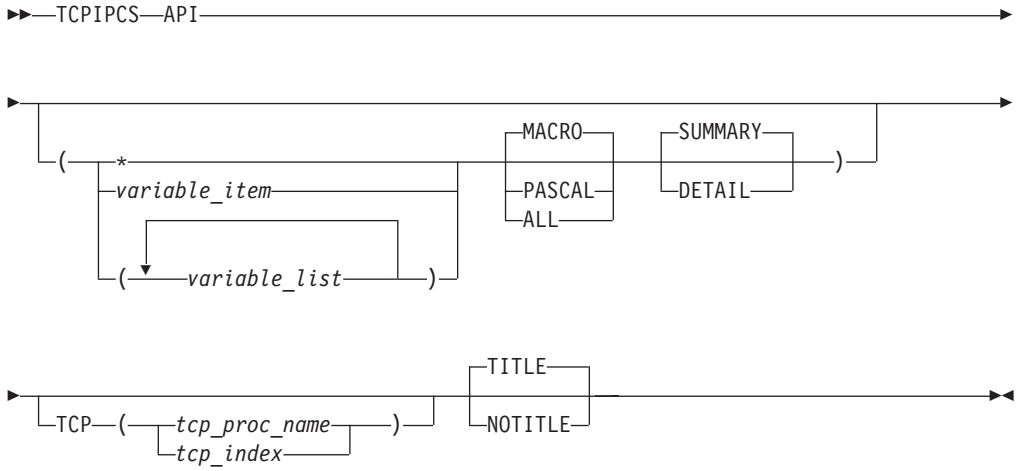
Display information about the connects in the Sockets Extended Assembler Macro Application Programming Interface (Macro API) and the Pascal API.

Note: The Macro API is the base for the CALL Instruction API, the CICS C API, and the CICS EZACICAL API. See the *OS/390 IBM Communications Server: IP Application Programming Interface Guide* for more information about the native TCP/IP APIs.

Some API control blocks are in the application address space, which may not be available in the dump. If the application address space is available, the API control blocks will be formatted.

Syntax

Following is the syntax of the TCPIPCS API subcommand:



Parameters

If no parameters are specified, only information about the Macro API is summarized. Following are the parameters for the TCPIPCS API subcommand:

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, separated by a blank space, within parentheses:

Variable parameters are:

jobname
Displays only the API control blocks for this job name. The job name may be a TCP/IP application name or a stack name. Must be from 1–8 characters.

ASCB_address
Displays the API control blocks with this address space control block (ASCB) address. An IPCS symbol name may be specified for the address. The address is specified as 1–8 hexadecimal digits. If an address begins with digit A –F, prefix the address with a zero to avoid the address being interpreted as a symbol name or as a character string.

ASID_number
Displays the API control blocks with this Address Space Identifier (ASID). The ASID is a hexadecimal number containing 1–4 digits.

In addition to the variable parameters described above, the following keyword parameters may be specified:

MACRO
Only display information for Macro APIs. MACRO is the default.

PASCAL
Only display information for Pascal APIs.

ALL
Display information for both APIs.

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 89 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {MACRO, PASCAL, ALL}, only the last one will be used.
2. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a sample output of the TCPIPCS API subcommand.

```
TCPIPCS API
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
```

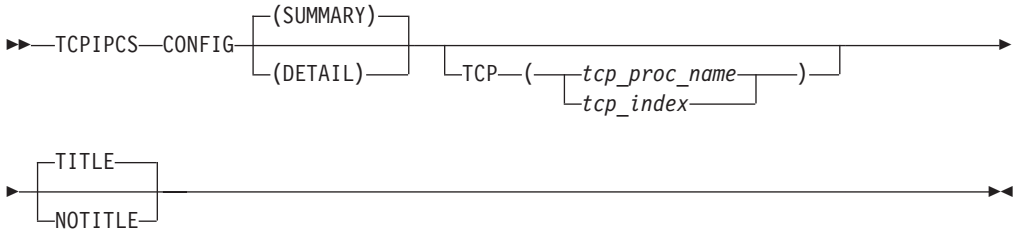
```
13391C00  1 TCPSVT    V2R10   1323B000 1323B0C8 07DE 04041405  Active
13391C80  2 TCPSVT2   V2R10   00000000 00000000 07E8 00000000  Down Stopping
13391D00  3 TCPSVT1   V2R10   12FC3000 12FC30C8 0080 94FF755F  Active
13391D80  4 TCPSVT3   V2R10   00000000 00000000 0059 00000000  Down Stopping

   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 API Analysis

Macro API Analysis:

CSVT at 13336120

TSCA at 13236A58

Macro API Analysis for TCPSVT Asid=07DE Tcb@=007E19D8:

  SDST for ASID 07DE at 7F5B0798

Macro API Analysis for ASCHINT Asid=0032 Tcb@=007E1748:

  SDST for ASID 07DE at 7F41C798

Macro API Analysis for ASCHINT Asid=0032 Tcb@=007E14B8:

  SDST for ASID 07DE at 7F3AB798

Macro API Analysis for ASCHINT Asid=0032 Tcb@=007E1320:

  SDST for ASID 07DE at 7F3A5798

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS CONFIG

Invocation of this subcommand displays 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

Following is the syntax of the TCPIPCS CONFIG subcommand:



Parameters

Following are the parameters for the TCPIPCS CONFIG subcommand:

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 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a sample output of the TCPIPCS CONFIG subcommand.

```
TCPIPCS CONFIG
Dataset: IPCS.A594094.DUMPB
Title:   TCPCS    V2RA: Job(TCPCS   ) EZBIPLIF(HTCP380 99.137)+
         001022 S0C4/00000010 SRB P=0024,S=0024,H=0024.

The address of the TSAB is: 08CEA280

Tseb     SI Procedure Version Status       Tsdb     Tsdx     Asid TraceOpts

08CEA2C0  1 TCPCS     V2R10   Active       0885D000 0885D0C8 0024 9FFFFF7F

   1 defined TCP/IP(s) were found
   1 active  TCP/IP(s) were found

   1 TCP/IP(s) for CS     V2R10  were found

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

Analysis of Tcp/Ip for TCPCS    . Index:1

Configuration control block summary

IPMAIN found at 08759410

Dif@     DeviceName       Next     Prev     DevR DevW Protocol
7F6DA888 LOOPBACK         7F577888 00000000 **** **** Loopback
7F577888 VDEV1            7F576888 7F6DA888 **** **** VIPA
7F576888 VDEV2            7F576108 7F577888 **** **** VIPA
7F576108 CTC093           7F55E888 7F576888 0AA1 0AA0 CTC
7F55E888 CTC092           7F55E108 7F576108 0D02 0D03 CTC
7F55E108 IUTSAMEH         00000000 7F55E888 **** **** MPC

Pif@     LinkName  Next     Prev     DeviceName  Protocol  Dif@     Lif@
7F6E43A8 LOOPBACK  7F577608 00000000 LOOPBACK    Loopback  7F6DA888 7F6E42A8
7F577608 VLINK1    7F577388 7F6E43A8 VDEV1       VIPA      7F577888 7F576008
7F577388 VLINK2    7F577108 7F577608 VDEV2       VIPA      7F576888 7F56D088
7F577108 CTCAA0    7F56D408 7F577388 CTC093      CTC       7F576108 7F55E008
7F56D408 CTCD2     7F56D188 7F577108 CTC092      CTC       7F55E888 7F55DF08
7F56D188 TOVTAM    00000000 7F56D408 IUTSAMEH    MPC       7F55E108 7F55DE08

Lif@     LinkName         Next     Prev     Pif@     IpAddr
7F576008 VLINK1           7F56D088 00000000 7F577608 009.067.116.094
7F56D088 VLINK2           7F55E008 7F576008 7F577388 009.067.113.094
7F55E008 CTCAA0           7F55DF08 7F56D088 7F577108 009.067.116.201
7F55DF08 CTCD2            7F55DE08 7F55E008 7F56D408 009.067.116.203
7F55DE08 TOVTAM           7F6E42A8 7F55DF08 7F56D188 009.067.116.240
7F6E42A8 LOOPBACK         00000000 7F55DE08 7F6E43A8 127.000.000.001

Configuration Summary
```

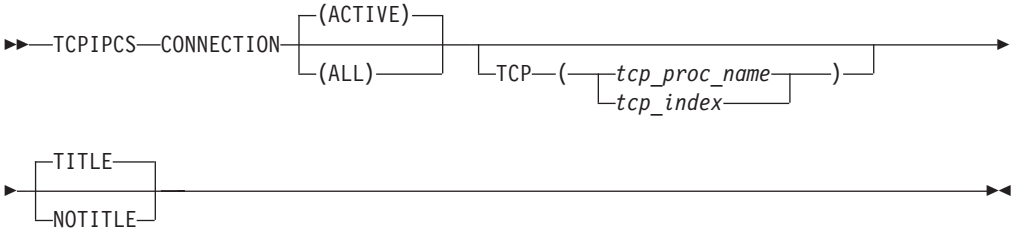
```
Lif@     LinkName         DeviceName       DevR DevW IpAddr
7F576008 VLINK1           VDEV1            **** **** 009.067.116.094
7F56D088 VLINK2           VDEV2            **** **** 009.067.113.094
7F55E008 CTCAA0           CTC093           0AA1 0AA0 009.067.116.201
7F55DF08 CTCD2            CTC092           0D02 0D03 009.067.116.203
7F55DE08 TOVTAM           IUTSAMEH         **** **** 009.067.116.240
7F6E42A8 LOOPBACK         LOOPBACK         **** **** 127.000.000.001

Analysis of Tcp/Ip for TCPCS    completed
```

TCPIPCS CONNECTION

Invocation of this command displays information about TCP, UDP and raw connections. The information includes the user ID, connection ID, local IP address, foreign IP address, the connection state (for TCP connections only), and the protocol name (for raw connections only).

Syntax

Following is the syntax of the TCPIPCS CONNECTION subcommand:



Parameters

Following are the parameters for the TCPIPCS CONNECTION subcommand:

ACTIVE
Display only active connections. This is the default. Note that the number of connections reported for each protocol includes both inactive and active connections. So the total may be higher than the displayed (active) connections.

ALL
Display all connections, regardless of state.

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {ACTIVE, ALL}, only the last one will be used.

Sample Output

The following is a sample output of the TCPIPCS CONNECTION subcommand.

Note: For the sample output shown, the default option is ACTIVE, so only active connections are shown. There are 10 total TCP connections, of which 8 are active. There is one UDP connection which is not active. Both raw connections are active.

```
TCPIPCS CONNECTION
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


TCP Connections:

Userid   Conn     Local Socket           Foreign Socket         State
FTPD1    0000000E 0.0.0.0..21            0.0.0.0..0             Listening
TCPCS    00000012 0.0.0.0..23            0.0.0.0..0             Listening
TCPCS    0000000B 0.0.0.0..1025          0.0.0.0..0             Listening
BPXOINIT 00000013 0.0.0.0..10007         0.0.0.0..0             Listening
TCPCS    00000011 127.0.0.1..1025        127.0.0.1..1026        Established
TCPCS    00000010 127.0.0.1..1026        127.0.0.1..1025        Established
USER15   00000019 127.0.0.1..1027        127.0.0.1..1028        Established
TCPCS    00000018 127.0.0.1..1028        127.0.0.1..1027        Established

10 TCP connections

1 UDP connections

RAW Connections:

Userid   Conn     Local Socket           Foreign Socket         Protocol
USER15   00000017 9.67.113.14            224.0.0.5              OSPFIGP
TCPCS    00000008 0.0.0.0                0.0.0.0                RAW

2 RAW connections

Analysis of Tcp/Ip for TCPCS completed
```

TCPIPCS DUAF

Invocation of this command displays 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:

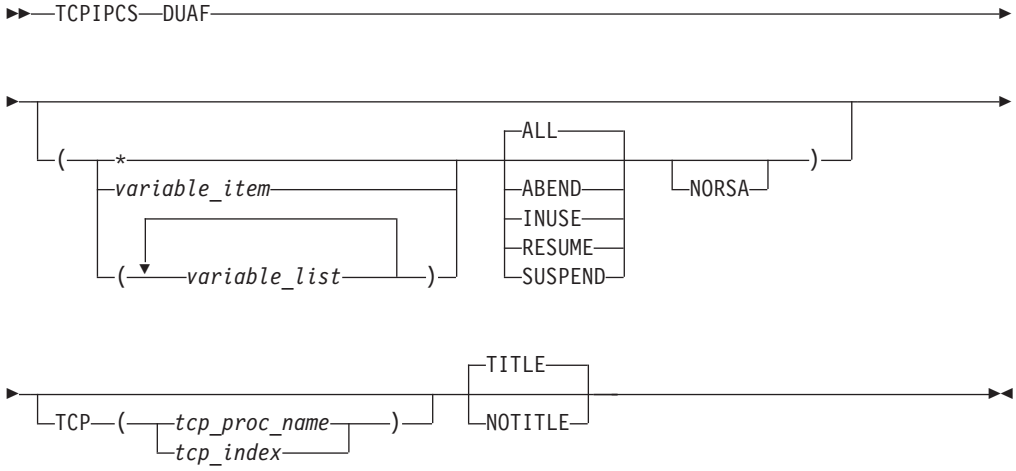
Ab The DUCB has ABENDed.
Iu The DUCB is in use.
Re The DUCB is in resume state.
Su The DUCB has been suspended.

The DUCB status may 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

Following is the syntax of the TCPIPCS DUAF subcommand:



Parameters

If no parameters are specified, all active DUCBs are summarized.

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, separated by a blank space, within parentheses:

jobname
Displays only the DUCBs with this job name. The job name may be a TCP/IP application name or a stack name. Must be from 1–8 characters.

DUCB_address
Displays the DUCB with this address. An IPCS symbol name may be specified for the address. The address is specified as 1–8 hexadecimal digits. If an address begins with digit A through 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 1–4 digits. The lowest index is zero.

ASCB_address
Displays the DUCB with this address space control block (ASCB) address. An IPCS symbol name may 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 1–4 digits.

In addition to the variable parameters described above, the following keyword parameters may be specified:

ALL
Display information for all active DUCBs. This is the default.

ABEND
Only display information for DUCBs that ABENDed.

INUSE
Only display information for DUCBs currently being used

RESUME
Only display information for DUCBs that are resumed.

SUSPEND
Only display 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 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {ALL, ABEND, INUSE, RESUME, SUSPEND}, only the last one will be used.

Sample Output

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
```

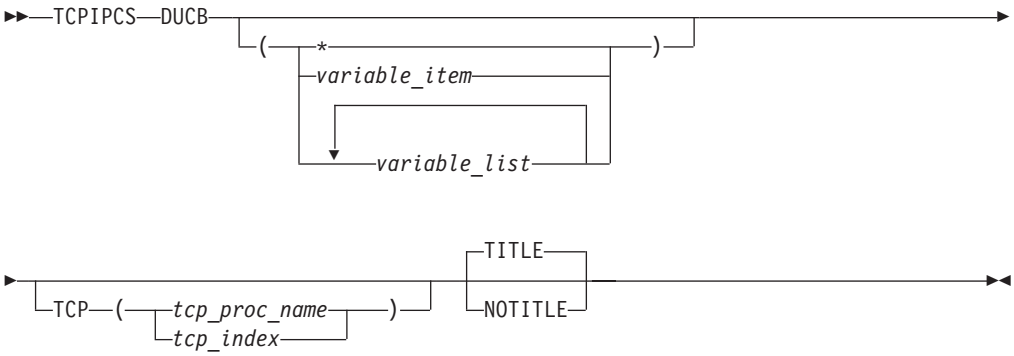
```
 4536 bytes were used

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

Invocation of this command displays 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. In the output, the DUAT is summarized. Then the contents of each DUCB is 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

Following is the syntax of the TCPIPCS DUCB subcommand:



Parameters

Following are the parameters for the TCPIPCS DUCB subcommand:

If no parameters are specified, all DUCBs are displayed.

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, separated by a blank space:

jobname
Displays only the DUCBs with this job name. The job name may be a TCP/IP application name or a stack name. Must be from 1–8 characters.

DUCB_address
Displays the DUCB with this address. An IPCS symbol name may 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 hexadecimal number containing 1–4 digits. The lowest index is zero.

ASCB_address
Displays the DUCB with this address space control block address (ASCB). An IPCS symbol name may 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 1–4 digits.

TCP, TITLE, NOTITLE
See "Parameters" on page 89 for a description of these parameters.

Sample Output

The following is a sample output of the TCPIPCS DUCB subcommand.

```
TCPIPCS DUCB(2E)
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

DUCB Detail Analysis


Dispatchable Unit Control Block: DUCB46
 EZBDUCB: 12762000
 +0000  DUCB_EYE.................. DUCB
 +0004  DUCB_LENGTH............... 0100
 +0006  DUCB_VERSION.............. 0002
 +0008  DUCB_TOKEN................ 12762000  2E1407DE  1000002E  00000000
 +0018  DUCB_DUSA................. 12762100
 +001C  DUCB_AVAIL_CHAIN.......... 1128E000
 +0020  DUCB_DUAEP................ 12FC1198
 +0026  DUCB_ASID................. 07DE
 +0028  DUCB_ASCB................. 00F7C280
```

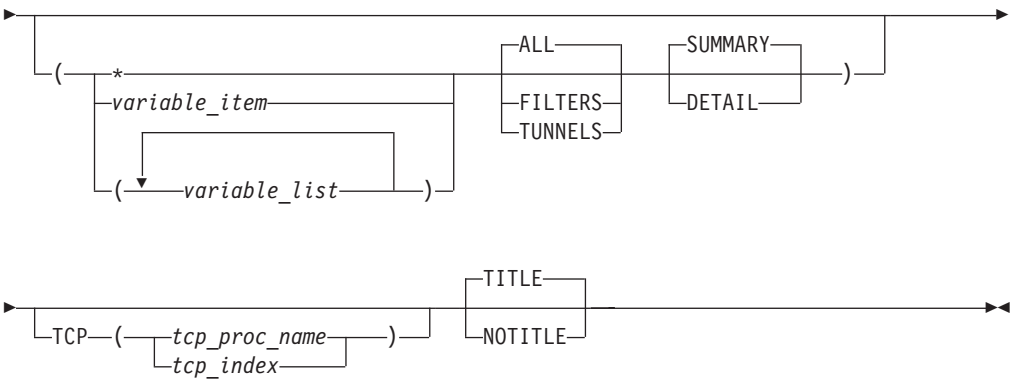
```
 +002C  DUCB_ATCB................. 00000000
 +0030  DUCB_ITCVT................ 1323B3C8
 +0034  DUCB_LOCKSHELDCOUNT....... 00000000
 +0038  DUCB_LOCKS_TABLE.......... 12762184
 +003C  DUCB_LOCKS_SUSPENDED...... 00000000
 +0040  DUCB_LOCKS_SUSPENDED_NEXT. 7FFAFAF1
 +0044  DUCB_SUSPENDTOKEN......... 800014F1  0296D7B0
 +004C  DUCB_JOBNAME.............. WEBSTCP
 +0054  DUCB_TSAB................. 13391BC0
 +0058  DUCB_TSEB................. 13391C00
 ...
Register Save Area: RSA46@1
Module: EZBITDUM
 EZBRSA: 127623E8
 +0000  RSA_DUSA. 12762100  RSA_PREV. 00000000  RSA_NEXT. 12762448
        RSA_R14.. 9320C62E  RSA_R15.. 11B6B2B4

 +0014  RSA_R0... 1333AFE0  RSA_R1... 12762430  RSA_R2... 1333AFE4
        RSA_R3... 1333AFC0  RSA_R4... 00000000

 +0028  RSA_R5... 00000008  RSA_R6... 12762000  RSA_R7... 00000000
        RSA_R8... 80000000  RSA_R9... 9320DCA0

 +003C  RSA_R10.. 12762430  RSA_R11.. 1333AFD8  RSA_R12.. 9320C588
        RSA_AR13. 1333AFDC  RSA_AR14. 1333AFE0

 +0058  RSA_AR15. 1333AFE4  RSA_AR0.. 11DA9D48  RSA_AR1.. C5E9C2E3
        RSA_AR2.. C3D7E3D4  RSA_AR3.. 12762100

 +006C  RSA_AR4.. 127623E8  RSA_AR5.. 127627C0  RSA_AR6.. 91B6CE08
        RSA_AR7.. 11B88CA6  RSA_AR8.. 1333AFDC

 +0080  RSA_AR9.. 127626AC  RSA_AR10. 00000000  RSA_AR11. 12762554
        RSA_AR12. 00000000

Parm(12762430):  12762000 1333AFD8 1333AFDC 1333AFE0 1333AFE4

Dynamic Area of RSA46@1
Module: EZBITDUM
 127623E8  12762100  00000000  12762448  9320C62E  | ............l.F. |
    +0010  11B6B2B4  1333AFE0  12762430  1333AFE4  | .......\.......U |
    +0020  1333AFC0  00000000  00000008  12762000  | ...{............ |
    +0030  00000000  80000000  9320DCA0  12762430  | ........l....... |
    +0040  1333AFD8  9320C588  12762000  1333AFD8  | ...Ql.Eh.......Q |
    +0050  1333AFDC  1333AFE0  1333AFE4  11DA9D48  | .......\...U.... |
...
```

TCPIPCS FIREWALL

Display information about Firewall filters or tunnels.

Syntax

Following is the syntax of the TCPIPCS FIREWALL subcommand:



Parameters

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

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, 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 may 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 may 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, the following keyword parameters may be specified:

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

FILTERS
Only display information for Firewall filters.

TUNNELS
Only display 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 89 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {ALL, FILTERS, TUNNELS}, only the last one will be used.
2. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

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:
```

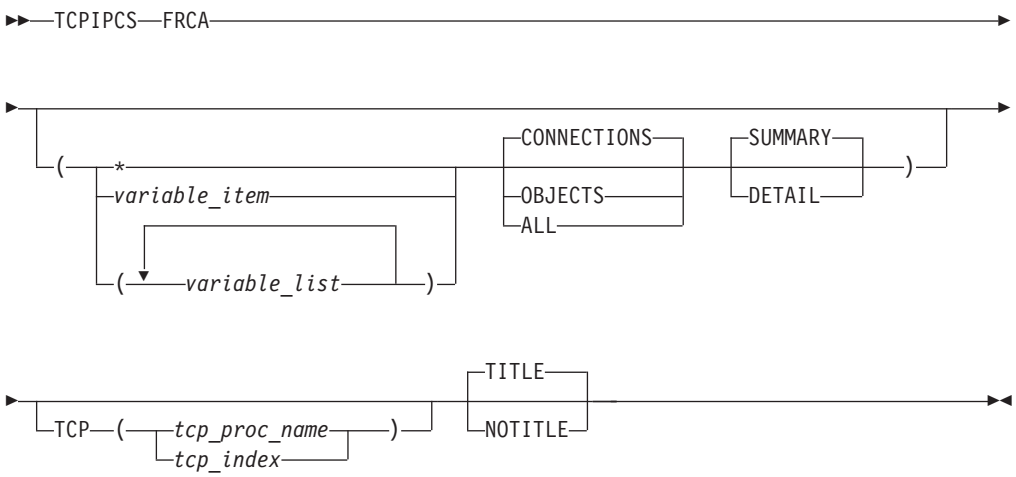
```
Name       Src@            Dst@            Policy   Format
0000000510:0000000508:0000000507:0000000516:0000000517:0000000506:0000000003
           9.67.116.36     9.67.116.47     00000021 00000033
0000000504:0000000508:0000000507:0000000504:0000000503:0000000506:0000000004
           9.67.116.36     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

Display information about the Fast Response Cache Accelerator (FRCA) connections or about cached objects.

Syntax

Following is the syntax of the TCPIPCS FRCA subcommand:



Parameters

If no parameters are specified, only FRCA connections are summarized.

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, 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 may 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 may 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 may be a TCP/IP application name or a stack name. The jobname is 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, the following keyword parameters may be specified:

CONNECTIONS
Only display information for FRCA connections. CONNECTIONS is the default.

OBJECTS
Only display 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 89 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {CONNECTIONS, OBJECTS, ALL}, only the last one will be used.
2. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a 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
```

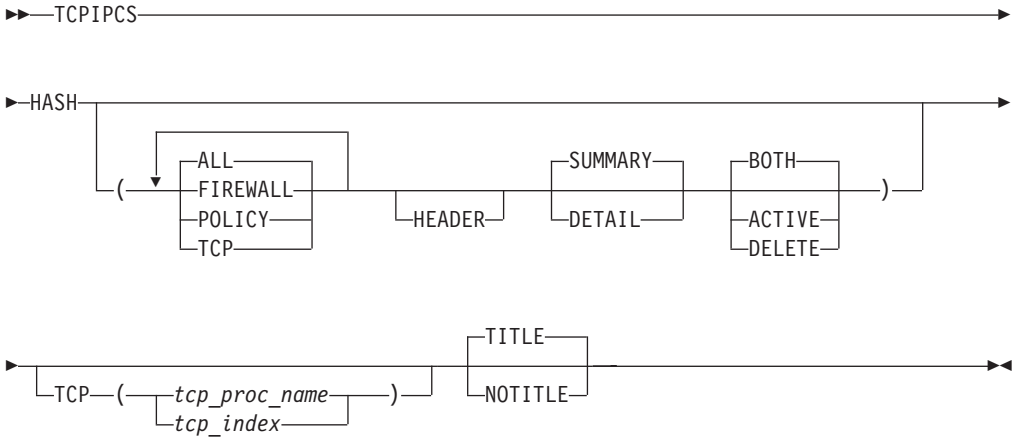
```
FRCA Server Connections

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

Display information about the structure of TCP/IP hash tables.

Syntax

Following is the syntax of the TCPIPCS HASH subcommand:



Parameters

The following keyword parameters may be specified:

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

FIREWALL
Only display structure of Firewall hash tables.

POLICY
Only display structure of Service Policy hash tables.

TCP
Only display structure of TCP 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
Only display active table elements.

DELETE
Only display logically deleted table elements.

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {ALL, FIREWALL, POLICY, TCP}, all of them will be used.
2. If you specify multiple keywords from the set {BOTH, ACTIVE, DELETE}, only the last one will be used.
3. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a sample output of the TCPIPCS HASH subcommand.

```
TCPIPCS HASH ( HEADER  ACTIVE TCP )
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 Hash Table Analysis

TCP Index Table

Hash Table Header at 7F61AB88
  Instance            : 2
  Active entries      : 6,131
  Hash buckets        : 62,533
  User free routine   : 92D2E7A6
  Element queue       : 12B57D90

Bucket# Bucket@  Element@ Status  User@
      0 7F61AB88 7F5FC020 Active  7F604108
    530 7F61CCA8 7F5FC0A0 Active  7F605108
   6536 7F634408 7F5FC2E0 Active  7F612508
   7080 7F636608 7F5FC0C0 Active  7F605908
  10083 7F6421B8 7F5FC100 Active  7F606108
  13086 7F64DD68 7F5FC120 Active  7F606508
```

```
 ...
 62263 7F70DEF8 7F324E60 Active  7F0BE508
 62264 7F70DF08 7F324E80 Active  7F263108
 62265 7F70DF18 7F602820 Active  7F47F108
 62266 7F70DF28 7F324EC0 Active  7F0FD108

6131 elements in TCB Index Table

Analysis of Tcp/Ip for TCPSVT completed
```

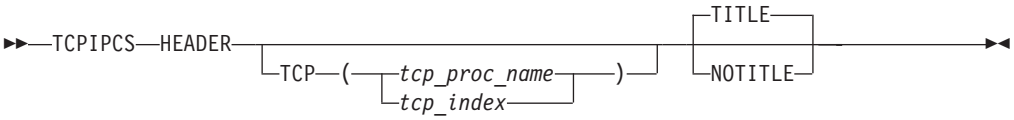
TCPIPCS HEADER

The TCPIPCS HEADER command displays information from the system dump header and, in some cases, if a DUCB has ABENDed, the DUCB will be 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 may or may not be available. If the DUCB address is available, the DUCB will be displayed. To find DUCBs that ABENDed, use the TCPIPCS DUAF (* ABEND) command.

Syntax

Following is the syntax of the TCPIPCS HEADER subcommand:



Parameters

The following keyword parameters may be specified:

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Sample Output

The following is a 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
```

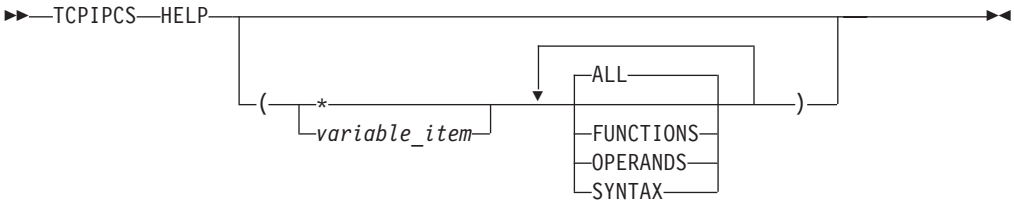
```
                                MVS Diagnostic Worksheet

 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

Display TCPIPCS usage and syntax information.

Syntax

Following is the syntax of the TCPIPCS HELP subcommand:



Parameters

If no parameters are specified, the function, operand, and syntax information is displayed for all TCPIPCS commands.

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the TCPIPCS subcommand names.

In addition to the variable parameters described above, the following keyword parameters may be specified:

ALL
Display information for all TCPIPCS commands. ALL is the default.

FUNCTIONS
Only display function information.

OPERANDS
Only display operand information.

SYNTAX
Only display syntax information.

Note: If you specify multiple keywords from the set {ALL, FUNCTION, OPERANDS, SYNTAX}, all of them will be used.

Sample Output

The following is a 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 LOCK

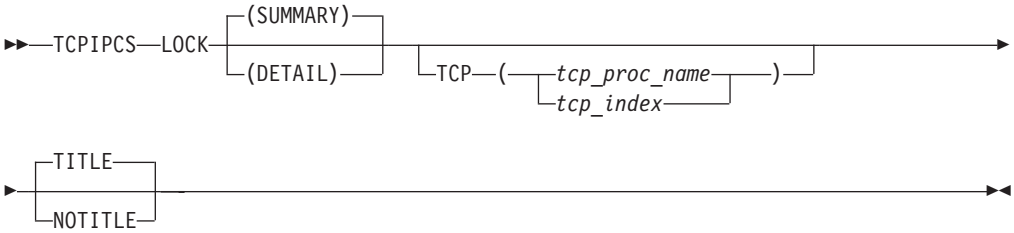
The TCPIPCS LOCK command scans the dump for information about the current locks that are defined and held.

Only non-zero statistics are reported.

Note: The DUCB lock table entries may 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 moment the dump was obtained, the lockword counters may have been updated but the DUCB has not yet been updated.

Syntax

Following is the syntax of the TCPIPCS LOCK subcommand:



Parameters

Following is the parameter for the TCPIPCS LOCK subcommand:

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 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a 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

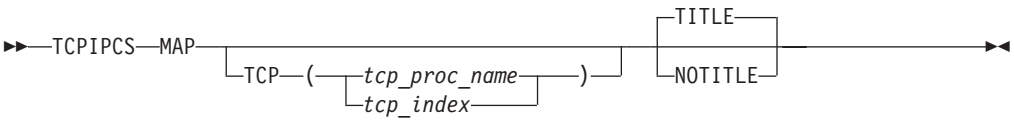
Invocation of this command displays a mapping of TCP/IP storage. This command is useful for finding overlays and abandoned storage.

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 quotes. 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. There may be multiple references, so additional references are continued on a separate line.

Note: Large dumps with many control blocks can take considerable time to process.

Syntax

Following is the syntax of the TCPIPCS MAP subcommand:



Parameters

Following is the parameter for the TCPIPCS MAP subcommand:

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Sample Output

The following is a sample output of the TCPIPCS MAP subcommand.

```
TCPIPCS MAP
Dataset: IPCS.MV20767.DUMPA
Title:   VERIFY MV20758


The address of the TSAB is: 08DD36F8
```

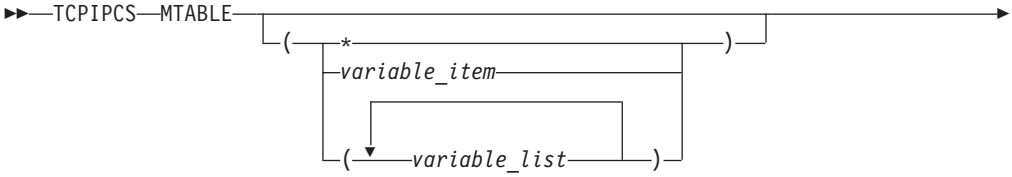
```
Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

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
```

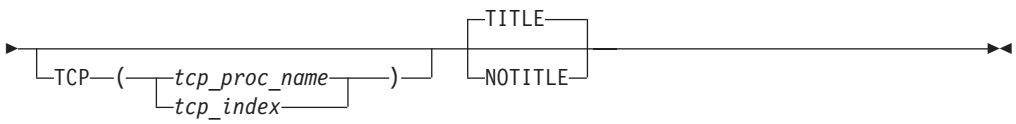
TCPIPCS MTABLE

Invocation of this subcommand accesses the module tables and displays the module entry point address, name, compile date and time, PTF number, and load module name. The entries are listed first in entry-point-address order and then listed again in module-name order.

Syntax

Following is the syntax of the TCPIPCS MTABLE subcommand:





Parameters

Following are the parameters for the TCPIPCS MTABLE subcommand. If no parameters are specified, all displayable modules are displayed.

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1-32 of the following variable parameters may be repeated, 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 may 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 89 for a description of these parameters.

Sample Output

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
```

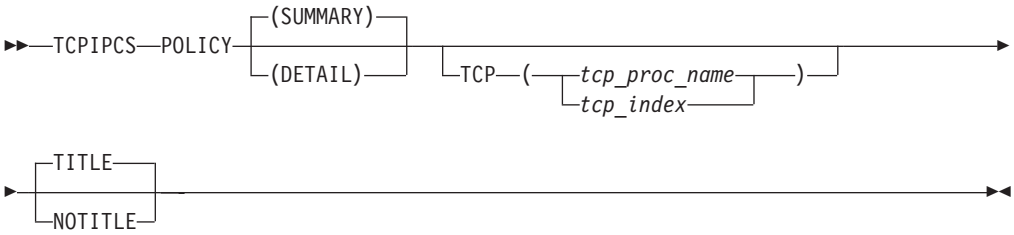
```
EZBXFINI 12D9B808 1999/10/08 00:37:29 HTCP50A  EZBTIINI 0051

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

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS POLICY

Display information about service policies.

Syntax

Following is the syntax of the TCPIPCS POLICY subcommand:



Parameters

Following is the parameter for the TCPIPCS POLICY subcommand:

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 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a 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
```

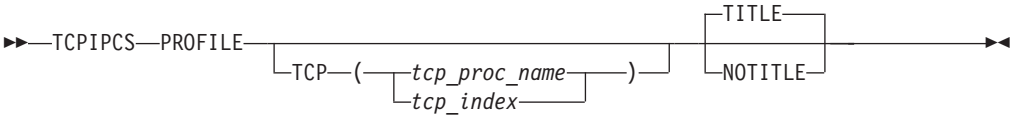
```
   8 TCP/IP(s) for CS     V2R10 found

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

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

Invocation of this subcommand shows the active configuration information at the time of the dump, in the form of profile dataset statements. This profile will not necessarily match the profile used to start TCP/IP because the start-up profile would not include the dynamic changes, additions, or deletions made via commands. All the defaults that are in effect are displayed in addition to explicit settings.

Syntax

Following is the syntax of the TCPIPCS PROFILE subcommand:



Parameters

Following is the parameter for the TCPIPCS PROFILE subcommand:

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Sample Output

The following is a sample output of the TCPIPCS PROFILE subcommand.

```
TCPIPCS PROFILE NOTITLE

  ;
  ; Profile generated on 2000/03/13 at 20:23:54
  ;
  ; Dump Dataset  : IPCS.MV21018.DUMPA
  ; Dump Time     : 2000/02/02 22:45:56.893646
  ; TCP/IP Jobname: TCPSVT
  ;

  ;
  ; For informational purposes, both GATEWAY and
  ; BEGINRoutes statements may be generated in this
  ; reconstructed profile.
  ;
  ; GATEWAY and BEGINRoutes statements cannot both
  ; be specified in a real profile/obeyfile dataset.
  ;

 ARPAGE 20

 ASSORTedparms IGNORERedirect VARSUBNETTING SOURCEVIPA
 ENDASSORTedparms

 AUTOLog 5
  FTPMVS   JOBNAME FTPMVS1
  FTPUNIX  JOBNAME FTPUNIX1
  OSNMPD   JOBNAME OSNMPD
  CBSAMPLE JOBNAME CBSAMPLE
  PORTMAP  JOBNAME PORTMAP
  WEBSTCP  JOBNAME WEBSTCP
  REXECD   JOBNAME REXECD
  OMPROUTE JOBNAME OMPROUTE
  NAMED    JOBNAME NAMED
 ENDAUTOLog

 BEGINRoutes
  ROUTE 202.51.52.153 HOST 155.155.155.2 LBRSAMEHOST MTU 1500
        MAXImumretransmittime 120 MINImumretransmittime 0.5
        ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
        NODELAYAcks
  ROUTE 202.51.52.0 255.255.255.0 155.155.155.2 LBRSAMEHOST MTU 1500
        MAXImumretransmittime 120 MINImumretransmittime 0.5
        ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
        NODELAYAcks
  ...
  ROUTE 155.155.155.3 HOST 155.155.155.2 LBRSAMEHOST MTU 1500
        MAXImumretransmittime 120 MINImumretransmittime 0.5
        ROUNDTRIPGain 0.125 VARIANCEGain 0.25 VARIANCEMultiplier 2
        NODELAYAcks
 ENDRoutes

 BSDRoutingparms TRUE
   LMBR2EC53 8096 20 255.255.240.0 0.0.0.0
   LEBRZ1355 8096 20 255.255.255.0 0.0.0.0
   LBRVIPA1 DEFAULTSize 1 255.255.255.0 0.0.0.0
   LBRSAMEHOST DEFAULTSize 1 255.255.0.0 0.0.0.0
 ENDBSDRoutingparms

 DEVice MBR2EC53 MPCPTP NOAUTORESTART
 LINK LMBR2EC53 MPCPTP MBR2EC53 IFSPEED 4500000 CHECKSUM

 DEVice EBRZ1355 MPCPTP NOAUTORESTART
 LINK LEBRZ1355 MPCPTP EBRZ1355 IFSPEED 4500000 CHECKSUM

 DEVice BRVIPA1 VIRTual 0000 NOAUTORESTART
 LINK LBRVIPA1 VIRTual 0 BRVIPA1
```

```
DEVice IUTSAMEH MPCPTP NOAUTORESTART
LINK LBRSAMEHOST MPCPTP IUTSAMEH IFSPEED 4500000 CHECKSUM

GATEWAY
  202.51.52.153 155.155.155.2 LBRSAMEHOST 1500 HOST
                MAXImumretransmittime 120 MINImumretransmittime 0.5
                ROUNDTRIPGain 0.125 VARIANCEGain 0.25
                VARIANCEMultiplier 2 NODELAYAcks
  202.51.52.0 155.155.155.2 LBRSAMEHOST 1500 0 MAXImumretransmittime
              120 MINImumretransmittime 0.5 ROUNDTRIPGain 0.125
              VARIANCEGain 0.25 VARIANCEMultiplier 2 NODELAYAcks
  ...
  155.155.155.3 155.155.155.2 LBRSAMEHOST 1500 HOST
                MAXImumretransmittime 120 MINImumretransmittime 0.5
                ROUNDTRIPGain 0.125 VARIANCEGain 0.25
                VARIANCEMultiplier 2 NODELAYAcks

GLOBALCONFig NOTCPStatistics

HOME
  197.51.155.1 LBRVIPA1
  155.155.155.1 LBRSAMEHOST
  201.51.53.155 LMBR2EC53
  216.51.55.155 LEBRZ1355

IPCONFig ARPTO 1200 DATAGRamfwd(NOFWDMULTipath) FIREWALL SOURCEVIPA
         VARSUBNETTING NOSYSPLEXRouting IGNORERedirect
         REASSEMBLytimeout 60 TTL 64 PATHMTUDISCovery NOMULTIPATH
         NODYNAMICXCF

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

KEEPAliveoptions INTerval 120 SENDGarbage FALSE
ENDKEEPaliveoptions

PKTTRACE FULL LINKNAME=LOOPBACK PROT=* IP=* SRCPort=* DESTport=*
PKTTRACE ON LINKNAME=LOOPBACK
...
PKTTRACE FULL LINKNAME=LBRSAMEHOST PROT=* IP=* SRCPort=* DESTport=*
PKTTRACE ON LINKNAME=LBRSAMEHOST

PORT 31010 TCP XTISRV NODELAYAcks
PORT 31000 UDP EZAIMS34 NODELAYAcks
...
PORT 7 TCP MISCSRV NODELAYAcks
PORT 7 UDP MISCSRV NODELAYAcks

SACONFig COMMUNity public AGENT 161 ENABLED SETSENAbled

SMFCONFIG NOTCPINIT NOTCPTERM NOFTPCLIENT NOTN3270CLIENT NOTCPStatistics

SMFPARMS 0 0 0

SOMAXCONN 10

STOP MBR2EC53
START EBRZ1355
START IUTSAMEH

TCPCONFIG INTerval 120 UNRESTRICTLowports TCPRCVBufrsize 16384
          TCPSENDBufrsize 16384 TCPMAXRCVBufrsize 262144 SENDGarbage
          FALSE
```

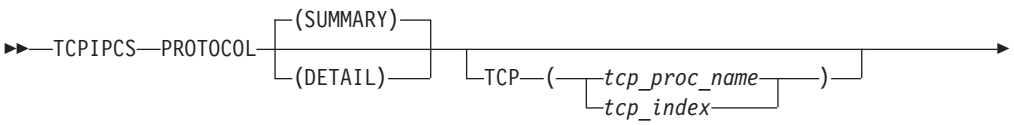
```
UDPCONFIG UNRESTRICTLowports UDPCHKsum UDPRCVBufrsize 65535
          UDPSENDBufrsize 65535 UDPQueuelimit

TELNETPARMS
  PORT 623
  CLIENTAUTH NONE
  MAXRECEIVE 65536
  MAXVTAMSENDQ 50
  SCANINTERVAL 1800
  SSLTIMEOUT 5
  TIMEMARK 10800
  TKOSPECLU 30
ENDTELNETPARMS

BEGINVTAM
  PORT 623
  ALLOWAPPL TSO* DISCONNECTABLE
  ALLOWAPPL * DISCONNECTABLE
  DEFAULTLUS
    TN000000..TN060000
  ENDDEFAULTLUS
  LUGROUP LUGRP1
    B10000..B19999
  ENDLUGROUP
  LUGROUP LUGRP2
    B20000..B29999
  ENDLUGROUP
  TELNETDEVICE 3278-3-E NSX32703
  TELNETDEVICE 3278-4-E NSX32704
  TELNETDEVICE 3278-5-E NSX32705
  TELNETDEVICE 3279-3-E NSX32703
  TELNETDEVICE 3279-4-E NSX32704
  TELNETDEVICE 3279-5-E NSX32705
  HNGROUP AHNGROUP
    A.A
  ENDHNGROUP
ENDVTAM

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS PROTOCOL

Display information from TCP, UDP, and raw protocol control blocks.

Syntax

Following is the syntax of the TCPIPCS PROTOCOL subcommand:



Parameters

Following is the parameter for the TCPIPCS PROTOCOL subcommand:

SUMMARY

Formats the MTCB, MUDP and MRCB contents. Lists all the TCBs, UDPs and RCBs in separate cross-referenced tables. SUMMARY is the default.

DETAIL
In addition to the SUMMARY display, DETAIL formats the contents of the TCB(s), UDP(s), and RCB(s).

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a 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
```

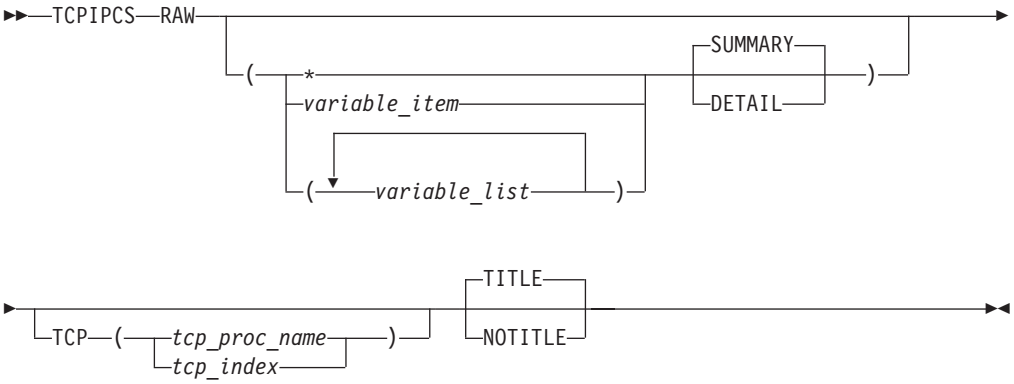
```
 7F5FD108 00000059 DHCP3    WLOIDLE  0.0.0.0         6001
 7F5FCB08 0000004B CBSAMPLE WLOIDLE  0.0.0.0         30001
 7F5FC908 00000048 MISCSRV  WLOIDLE  0.0.0.0           19
 7F5FC708 00000047 MISCSRV  WLOIDLE  0.0.0.0           19
....
 7F5F6B08 00000017 MISCSRV  WLOIDLE  0.0.0.0            7
 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

Invocation of this command displays the Master Raw Control Block (MRCB) and any Raw protocol Control Blocks (RCBs) defined in the MRCB hash table.

Syntax

Following is the syntax of the TCPIPCS RAW subcommand:



Parameters

If no parameters are specified, all raw connections are summarized.

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, separated by a blank space, within parentheses:

Variable parameters are:

jobname
Displays only the API control blocks for this job name. The job name may be a TCP/IP application name or a stack name. Must be 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 may 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 digits.

In addition to the variable parameters described above, the following keyword parameters may be specified:

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 RCB(s).

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a sample output of the TCPIPCS RAW subcommand.

```
TCPIPCS RAW
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
```

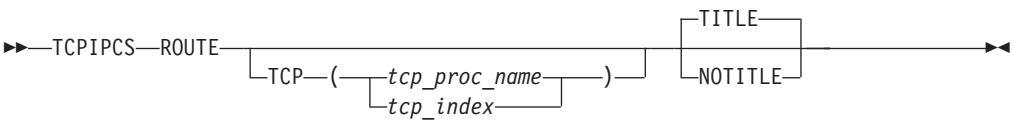
```
RCB      ResrcID  ResrcNm  TpiState   DestAddr            ProtocolId
7F5513C8 00000062 OMPROUTE WLOIDLE    129.11.208.108      89
7F712088 00000008 TCPSVT   WLOIDLE    0.0.0.0             255

2 RCB(s) FOUND
2 RCB(s) FORMATTED



Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS ROUTE

Invocation of this subcommand displays the routing control blocks. Each routing table entry is formatted to display the address device name, type, protocol, destination IP address, gateway IP address, and the physical interface control block address.

Syntax

Following is the syntax of the TCPIPCS ROUTE subcommand:



Parameters

Following is the parameter for the TCPIPCS ROUTE subcommand:

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Sample Output

The following is a sample output of the TCPIPCS ROUTE subcommand.

```
TCPIPCS ROUTE
Dataset: IPCS.MV21018.DUMPA
Title:   STORAGE SHORTAGE


The address of the TSAB is: 130B2560

Tseb     SI Procedure Version Tsdb     Tsdx     Asid TraceOpts Status

130B25A0  1 TCPSVT    V2R10   12250000 122500C8 003E 9CFF755F  Active
130B2620  2 TCPSVT1   V2R10   11A15000 11A150C8 004D 9CFF755F  Active
130B26A0  3 TCPSVT2   V2R10   11A0D000 11A0D0C8 004A 9CFF755F  Active

   3 defined TCP/IP(s) were found
   3 active  TCP/IP(s) were found

   3 TCP/IP(s) for CS     V2R10 found

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

Analysis of Tcp/Ip for TCPSVT.  Index: 1

TCPIP Route Analysis

Routes in Search Table

Rte@     DeviceName       Type            Protocol        DestinationAddr
 GatewayAddr     Pif@
```

```
7F55D1A8 LEBRZ1355        Host           OSPF           216.51.56.113
 216.51.55.202   7F5DD188
7F4C95E8 LEBRZ1355        Host           OSPF           216.51.56.202
 216.51.55.202   7F5DD188
7F5CA208 LEBRZ1355        Host           Configuration  216.51.55.155
 0.0.0.0         7F5DD188
7F55A548 LEBRZ1355        Host           OSPF           216.51.55.202
 0.0.0.0         7F5DD188
7F55A6A8 LEBRZ1355        Direct         OSPF           216.51.55.0
 0.0.0.0         7F5DD188
7F5CC228 LBRSAMEHOST      Host           Configuration  202.51.52.153
 155.155.155.2   7F5DB188
7F5CB848 LBRSAMEHOST      Network        Configuration  202.51.52.0
 155.155.155.2   7F5DB188
7F5CBEA8 LBRSAMEHOST      Network        Configuration  202.51.50.0
 155.155.155.2   7F5DB188
7F5CBB88 LBRSAMEHOST      Network        Configuration  202.51.51.0
 155.155.155.2   7F5DB188
7F4C9488 LEBRZ1355        Host           OSPF           202.202.202.202
 216.51.55.202   7F5DD188
7F4BE248 LEBRZ1355        Host           OSPF           201.201.201.201
 216.51.55.202   7F5DD188
7F5CC928 LBRSAMEHOST      Host           Configuration  200.51.155.2
 155.155.155.2   7F5DB188
7F5CCEA8 LBRSAMEHOST      Host           Configuration  200.51.153.1
 155.155.155.2   7F5DB188
7F5CCC48 LBRSAMEHOST      Host           Configuration  200.51.156.1
 155.155.155.2   7F5DB188
7F5CC568 LBRSAMEHOST      Host           Configuration  200.51.155.3
 155.155.155.2   7F5DB188
7F4E3A88 LEBRZ1355        Host           OSPF           200.51.113.2
 216.51.55.202   7F5DD188
7F4BC0E8 LEBRZ1355        Network        OSPF           200.22.64.0
 216.51.55.202   7F5DD188
7F5CB488 LBRSAMEHOST      Network        Configuration  202.51.54.0
 155.155.155.2   7F5DB188
7F4B3D08 LEBRZ1355        Host           OSPF           199.11.87.164
 216.51.55.202   7F5DD188

7F4BF668 LEBRZ1355        Host           OSPF           198.66.20.49
 216.51.55.202   7F5DD188
...
7F55A128 LEBRZ1355        Network        OSPF           113.0.0.0
 216.51.55.202   7F5DD188
7F4E2D48 LEBRZ1355        Network        OSPF           129.66.0.0
 216.51.55.202   7F5DD188

Routes in Update Table

Rte@     DeviceName       Type           Protocol       DestinationAddr
 GatewayAddr     Pif@
7F55D1A8 LEBRZ1355        Host           OSPF           216.51.56.113
 216.51.55.202   7F5DD188
7F4C95E8 LEBRZ1355        Host           OSPF           216.51.56.202
 216.51.55.202   7F5DD188
7F5CA208 LEBRZ1355        Host           Configuration  216.51.55.155
 0.0.0.0         7F5DD188
7F55A548 LEBRZ1355        Host           OSPF           216.51.55.202
 0.0.0.0         7F5DD188
7F55A6A8 LEBRZ1355        Direct         OSPF           216.51.55.0
 0.0.0.0         7F5DD188
7F5CC228 LBRSAMEHOST      Host           Configuration  202.51.52.153
 155.155.155.2   7F5DB188
7F5CB848 LBRSAMEHOST      Network        Configuration  202.51.52.0
 155.155.155.2   7F5DB188
7F5CBEA8 LBRSAMEHOST      Network        Configuration  202.51.50.0
 155.155.155.2   7F5DB188
```

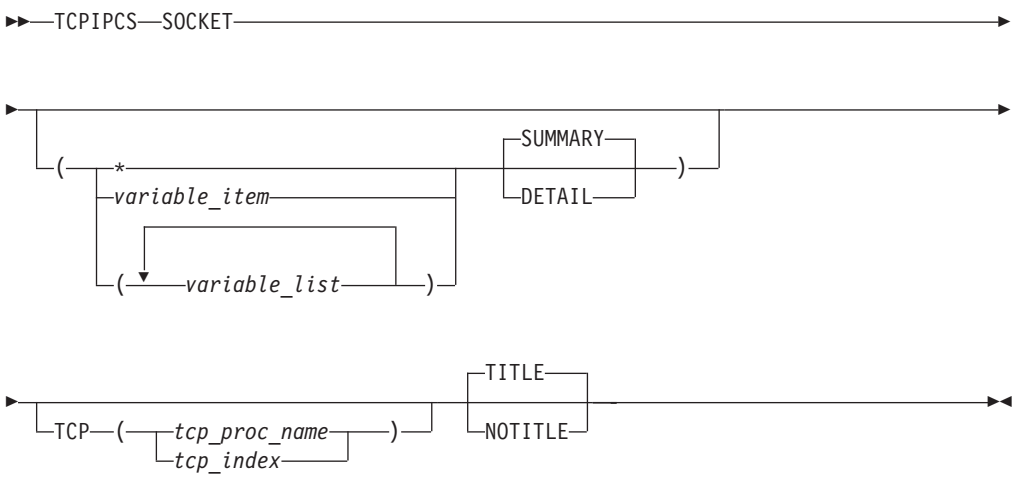
```
 ...
7F5ADD48 LEBRZ1355        Subnetwork      OSPF            51.51.128.0
 216.51.55.202   7F5DD188
7F55A128 LEBRZ1355        Network         OSPF            113.0.0.0
 216.51.55.202   7F5DD188
7F4E2D48 LEBRZ1355        Network         OSPF            129.66.0.0
 216.51.55.202   7F5DD188

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS SOCKET

Invocation of this command displays information from AF_UNIX socket control blocks.

Syntax

Following is the syntax of the TCPIPCS SOCKET subcommand:



Parameters

If no parameters are specified, all sockets are summarized.

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, separated by a blank space, within parentheses:

Variable parameters are:

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 may 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 may be specified:

SUMMARY
Summarizes the sockets. SUMMARY is the default.

DETAIL
In addition to the SUMMARY display, DETAIL formats the contents of the SCB(s).

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a 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
```

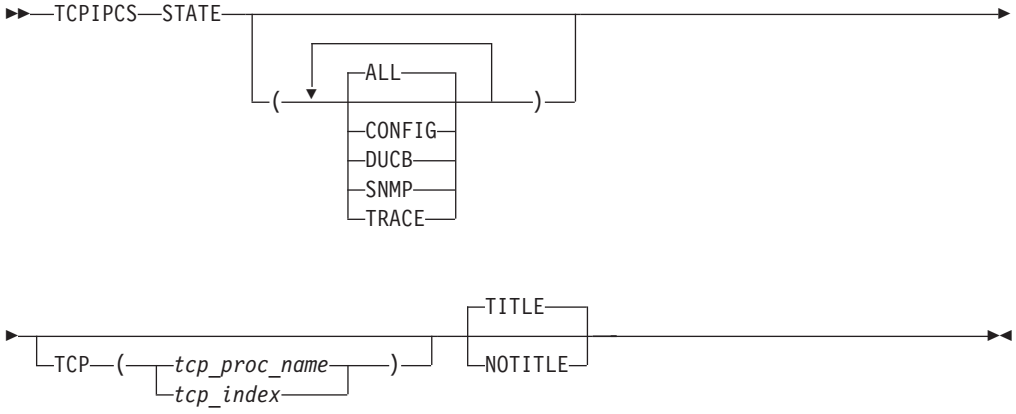
```
Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS STATE

Invocation of this subcommand provides an overall view of TCP/IP. Major control block addresses, subtasks, storage usage, dispatchable units, trace and configuration are displayed.

Syntax

Following is the syntax of the TCPIPCS STATE subcommand:



Parameters

The following keyword parameters may be specified:

ALL
Display all state information. ALL is the default.

CONFIG
Only display configuration state information.

DUCB
Only display DUCB state information.

SNMP
Only display SNMP and CONFIG information. (SNMP information makes sense only in the context of the configuration, so the configuration information will also be displayed.)

TRACE
Only display trace state information.

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {ALL, CONFIG, DUCB, SNMP, TRACE}, all of them will be used.

Sample Output

The following is a sample output of the TCPIPCS STATE subcommand.

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

```
TCPIP State

TCPIP Status:
 Procedure:     TCPSVT
 Version:       V2R10
 Status:        Active
 Asid:          0051
 Started:       1999/10/23 21:19:31
 Ended:         1999/10/24 00:08:07
 Active:        2.80 hours

Major Control Blocks
 TSEB:          12E89BF8       TSDB:          12B57000
 TSDX:          12B570C8       TCA:           12A2FE50
 ITCVT:         12B573C8       ITSTOR:        12B575C8
 DUAF:          12B56010       MRCB:          7F77EEA8
 MTCB:          12E6D290       MUCB:          7F70F268
 IPMAIN:        12A26410       Streams_root:  7F78BD88
 TosMains:      12E6CE70       MIB2:          12E6A448
 CdCb:          12E6A408       User:          13D092D0
 Conf:          12E6C8F0       Stks:          137CB0B0
================================================================================
TCPIP Subtasks
Task      Tcb       FirstRB   EotECB    StopEcb   CmpCode   RsnCode   RTWA
EZBTCPIP  006EC928  006FD0C8  806FDDC0            00000000  00000000  00000000
EPWPITSK  006EC698  006ECB30  00000000            00000000  00000000  00000000
EZBITTUB  006EC408  006EC870  00000000  806EC870  00000000  00000000  00000000
........  006EC270  006D56F8  00000000  806EC178  00000000  00000000  00000000
EZACDMSM  006D5520  006D5A08  00000000  806D5A08  00000000  00000000  00000000
EZBTTMST  006E0E88  006D5488  006E40E0            00000000  00000000  00000000
================================================================================
Storage Cache Information
  Total CSA Allocated:                 177,579K
  Total CSA Elements:                    21,469
  Cache Delay:                                0 seconds
  Scan Delay:                               120 seconds
  Total cache allocated:                 48,416
  Total cache elements:                      11
  Total freed elements:                       2
  Last cache scan time:     1999/10/24 04:06:12

CSM Status
  ECSA Storage:           OK
  Data Space Storage:     OK
  Fixed Storage:          OK
Alet: 01FF000B               Dspname: 00000CSM

================================================================================
Dispatchable Unit Status
  DUCB Initializations:                10,375K
  DUCB Expansions:                   2,180,028
  Percent DUCB expansions:                  21 %
  Last DUCB scan time:     1999/10/24 04:04:20
DU 1000001C at 12AB3000 for TCPSVT   indicates abend: S4C5 74BE2500.
DU 10000020 at 12ABF000 for TCPSVT   indicates abend: S4C5 74BE2500.
DU 10000029 at 12ADA000 for TCPSVT   indicates abend: S878 00000008.
DU 10000035 at 12AFE000 for TCPSVT   indicates abend: S4C5 74BE2500.
DU 10000039 at 12B0A000 for TCPSVT   indicates abend: S4C5 74BE2500.

   1 DUAT control block(s) were found in the DUAF at 12B56010
  82 Dispatchable units were found.
   5 DU(s) indicate abend
================================================================================

CTrace Status:
   Member Name  : CTIEZB01
   Buffer Size  : 104,858K
```

```
   Options      : Init Socket AFP XCF Access PFS API Engine RAW
                  UDP TCP ICMP ARP Route CLAW LCS Internet
                  Message WorkUnit SockAPI Config SNMP IOCTL
                  FireWall VtamData Telnet Vtam
   Asid List    : ()
   JobNameList  : ()
   Xwriter      : Disconnected
   Trace Count  : 513,412K
   Lost Count   : 0
   Lost Time    : 1900/01/01 00:00:00
   Wrap Count   : 2,256
   Wrap Time    : 1999/10/24 04:07:51

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS STORAGE

Invocation of the command displays the TCP/IP storage summary referenced in common cached storage.

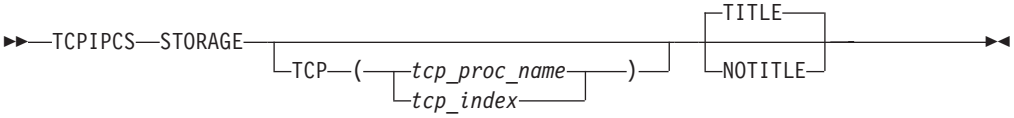
Under the heading Storage Summary, a ″c″ in column ″c″ indicates the address is on the cache queue. A ″p″ in column ″p″ indicates that the control block is part of a pool.

Cache storage has twelve bytes from offset 4 overlaid with a chain pointer and time stamp. This may show incorrect data for cached control blocks.

Note: 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

Following is the syntax of the TCPIPCS STORAGE subcommand:



Parameters

Following is the parameter for the TCPIPCS PROFILE subcommand:

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Sample Output

The following is a 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
```

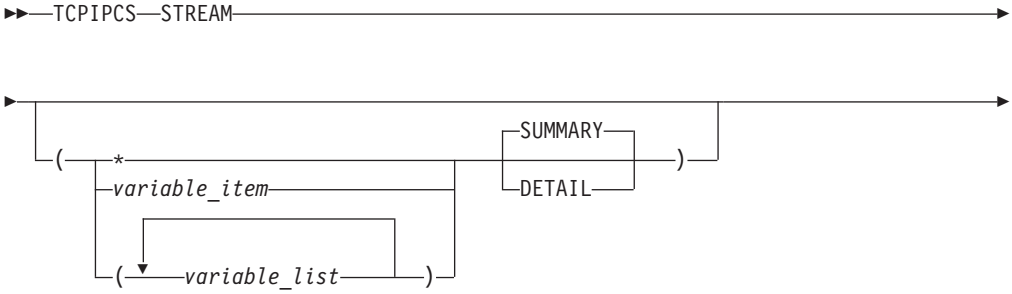
```
num_freed                      2 elements freed during last scan
scan_time    1999/10/24 04:06:12 time of last scan
dsa_init              10,375,262 # of DUCB initializations
dsa_exp                2,180,028 # of DUCB expansions
The control block at 008AC010 (Prev: 00000000) has already been added
...
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
```

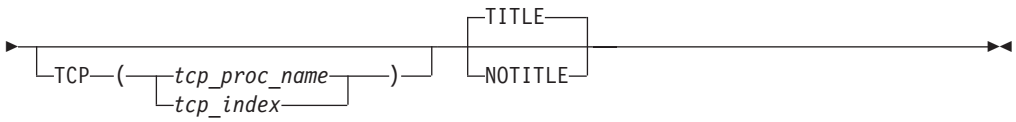
TCPIPCS STREAM

Invocation of this command displays the stream control blocks.

Syntax

Following is the syntax of the TCPIPCS STREAM subcommand:





Parameters

If no parameters are specified, all stream control blocks are summarized.

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, 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 may 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, the following keyword parameters may be specified:

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 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a sample output of the TCPIPCS STREAM subcommand.

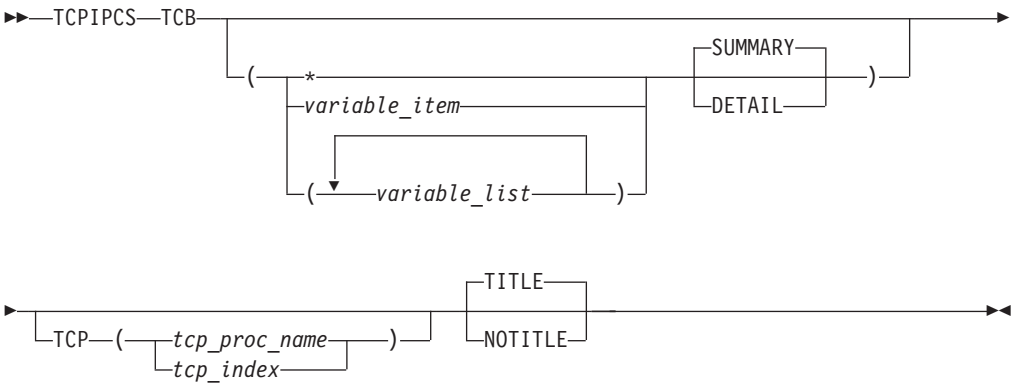
```
TCPIPCS STREAM
Dataset: IPCS.A594094.DUMPM
Title:   TCPSVT   V2R10: Job(TCPSVT  ) EZBITSTO(HTCP50A 99.281)+
         00077A S4C5/74BE2500 SRB P=0051,S=0051,H=0051

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

Invocation of this command displays the Master Transmission Control Block (MTCB) and any Transmission protocol Control Blocks (TCBs) defined in the TCP hash table.

Syntax

Following is the syntax of the TCPIPCS TCB subcommand:



Parameters

If no parameters are specified, all TCP control blocks are summarized.

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, separated by a blank space, within parentheses:

Variable parameters are:

jobname
Displays only the TCBs with this job name. The job name may be a TCP/IP application name or a stack name. A jobname 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 may 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, the following keyword parameters may be specified:

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 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a 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

================================================================================
```

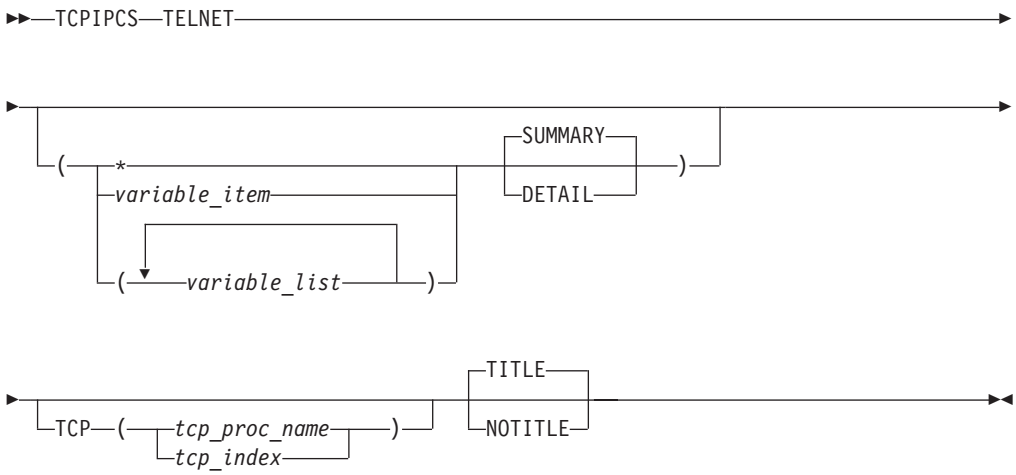
```
Analysis of Tcp/Ip for TCPSVT.  Index: 1


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    LuNa
 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

Invocation of this subcommand displays either the address or address and contents of Telnet control blocks. These include the TCMA, TCFG, TPDB, and, optionally, the TKCB and CVB for a selected session. A partial TCFG that is being built is also displayed, if found.

Syntax

Following is the syntax of the TCPIPCS TELNET subcommand:



Parameters

If no parameters are specified, all TCP control blocks are summarized.

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, separated by a blank space, within parentheses:

Variable parameters are:

LUname
Displays only the session control blocks for the eight-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 sixteen digits, it is padded on the right with zeros.

In addition to the variable parameters described above, the following keyword parameters may be specified:

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 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a 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
```

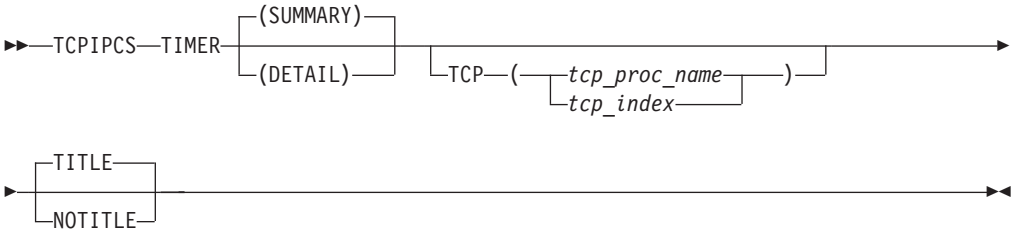
```
Tpdb@     Port Tcfg@    Prof Tkcb@    Token             Cvb@     LUname
7F59D8A0   623 7F5A6068 CURR 00000000 00000000 00000000 00000000
7F59D4E0   625 7F59D620 CURR 00000000 00000000 00000000 00000000

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS TIMER

Invocation of this command displays the timer control blocks.

Syntax

Following is the syntax of the TCPIPCS TIMER subcommand:



Parameters

Following is the parameter for the TCPIPCS TIMER subcommand:

SUMMARY
Displays the contents of the timer control blocks. The timer queue elements (TQE's) and timer id's (TID's) 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 are fully displayed.

TCP, TITLE, NOTITLE
See "Parameters" on page 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a 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
```

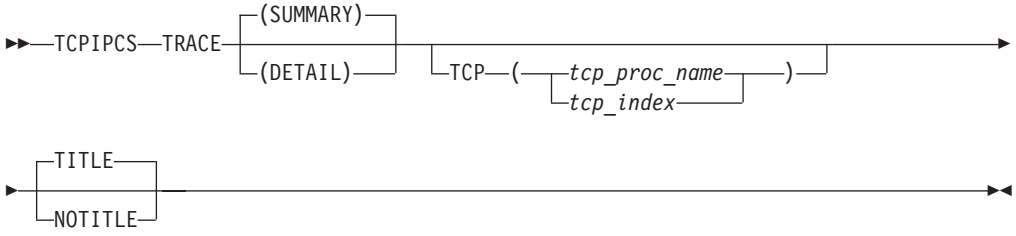
```
Timer tables at 086D8F80

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

Invocation of this command displays information about CTrace.

Syntax

Following is the syntax of the TCPIPCS TRACE subcommand:



Parameters

Following is the parameter for the TCPIPCS TRACE subcommand:

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 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a sample output of the TCPIPCS TRACE subcommand.

```
TCPIPCS TRACE
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

Trace Control Area
 TCA: 11469E50
 +0000  TCAACRONYM.......... TCA
 +0006  TCAVERSION.......... 0005
 +0008  TCASIZE............. 0002F188
 +000C  TCAFTBE............. 1146A1D8
 +0010  TCACURTBE........... 11471B40
 +0014  TCACURENT........... 030A044E
 +0018  TCATABSZ............ 06400000
 +001C  TCANUMBF............ 00000640
...

Event Trace Statistics for SYSTCPIP
  Size of the Trace Control Area . . . . 192904
  Size of the trace buffer . . . . . . . 102400K
  Size of a trace segment. . . . . . . . 64K
  Number of trace segments . . . . . . . 1600
  Maximum trace record size. . . . . . . 14,336
  Number of trace records requested. . . 854,878,442
  Number of trace records recorded . . . 710,809,778
  Number of trace segments filled. . . . 3,928,777
  Average records per segment. . . . . . 180
  Average records per table. . . . . . . 288,000
  Trace status . . . . . . . . . . . . . Active
  XWriter status . . . . . . . . . . . . Disconnected
  Number of buffers written. . . . . . . 846
  Lost record count. . . . . . . . . . . 0
  Lost record time . . . . . . . . . . . 1900/01/01 00:00:00.000000
  Trace table wrap count . . . . . . . . 2,456
  Trace table wrap time. . . . . . . . . 2000/03/22 12:33:48.626806
  Average records per wrap . . . . . . . 289,417

Data Trace Statistics for SYSTCPDA
  Size of the trace buffer . . . . . . . 204800K
  Size of a trace segment. . . . . . . . 64K
```

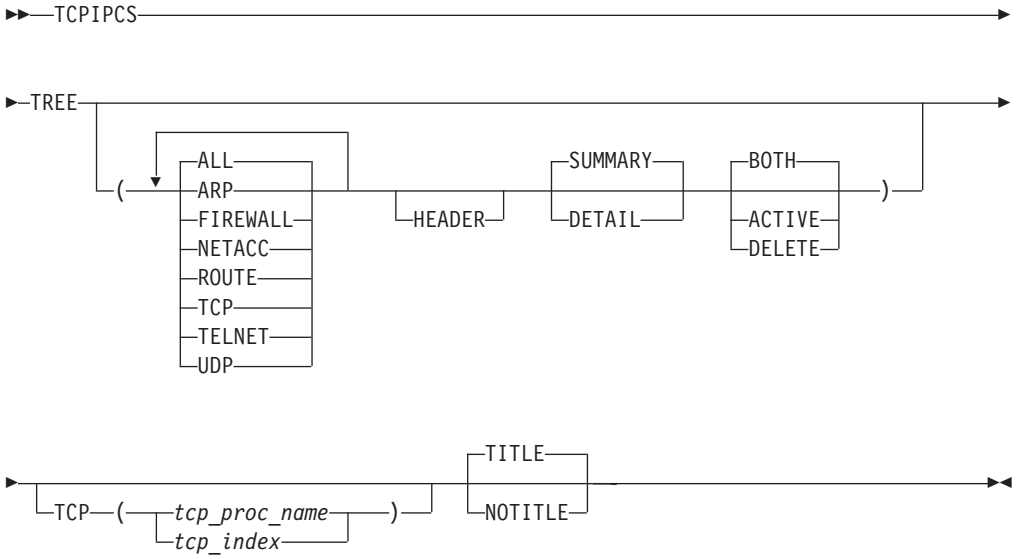
```
  Number of trace segments . . . . . . . 3200
  Number of trace records requested. . . 21,596,667
  Number of trace records recorded . . . 21,515,003
  Number of trace segments filled. . . . 68,826
  Number of lost records . . . . . . . . 0
  XWriter status . . . . . . . . . . . . Disconnected

Tseb_Trace_Opts: 04041405
Options: XCF TCP Internet Message VtamData Vtam

1600 SYSTCPIP Trace Buffer Elements were found
   0 SYSTCPIP Trace Buffer Elements were formatted
3200 SYSTCPDA Trace Buffer Elements were found
   0 SYSTCPDA Trace Buffer Elements were formatted


Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS TREE

Invocation of this command displays the structure of TCP/IP Patricia trees.

Syntax

Following is the syntax of the TCPIPCS TREE subcommand:



Parameters

The following keyword parameters may be specified:

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

ARP
Only display structure of ARP trees.

FIREWALL
Only display structure of Firewall trees.

NETACC
Only display structure of NetAccess trees.

ROUTE
Only display structure of route trees.

TCP
Only display structure of TCP trees.

TELNET
Only display structure of Telnet trees.

UDP
Only display structure of UDP trees.

HEADER
Display tree header information. Not displayed by default.

SUMMARY
Displays 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
Only display active tree nodes.

DELETE
Only display logically deleted tree nodes

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {ALL ARP, FIREWALL, NETACC, ROUTE, TCP, TELNET, UDP}, all of them will be used.
2. If you specify multiple keywords from the set {BOTH, ACTIVE, DELETE}, only the last one will be used.
3. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a sample output of the TCPIPCS TREE subcommand.

```
TCPIPCS TREE (UDP TELNET HEADER)
Dataset: IPCS.A594094.DUMPL
Title:   TCPIPR10 V2R10: Job(RUNNV16S)         (              )+
         ?????? S0E0/00000028 SRB P=0048,S=0048,H=0048
...
TCPIP Tree Analysis

Telnet LU Name Tree

Tree root Header at 7F7B4A68
  Instance            : 13
  Total elements      : 1
  Deleted elements    : 1
  Rebuild count       : 0
  Maximum key length  : 8 bytes
  Rebuild threshold   : 50
  Rebuild time        : 1900/01/01 00:00:00.000000

Node@     Bit Parent   LChild   RChild   Key      Element  Status
7F7B4A98    0 00000000 7F7B4A98 7F7B4A98 16644D48 00000000 Deleted

0 active elements in Telnet LU Name Tree
```

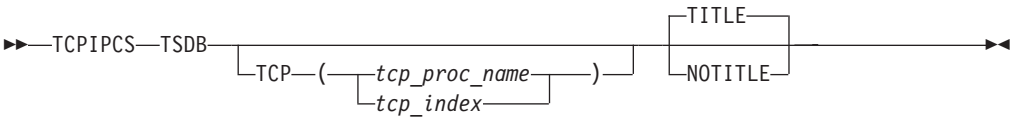
```
UDP DMUX Tree

Tree root Header at 7F7BE208
  Instance            : 2
  Total elements      : 1
  Deleted elements    : 0
  Rebuild count       : 0
  Maximum key length  : 6 bytes
  Rebuild threshold   : 20
  Rebuild time        : 1900/01/01 00:00:00.000000

Node@     Bit Parent   LChild   RChild   Key      Element  Status
7F7BE238    0 00000000 7F7BE238 7F7BE238 16644D48 7F630108 Active

1 active elements in UDP DMUX Tree

UDP SNMP Tree

Tree root Header at 7F7BE1A8
  Instance            : 3
  Total elements      : 1
  Deleted elements    : 0
  Rebuild count       : 0
  Maximum key length  : 6 bytes
  Rebuild threshold   : 20
  Rebuild time        : 1900/01/01 00:00:00.000000

Node@     Bit Parent   LChild   RChild   Key      Element  Status
7F7BE1D8    0 00000000 7F7BE1D8 7F7BE1D8 16644D48 7F630108 Active

1 active elements in UDP SNMP Tree

Analysis of Tcp/Ip for TCPIPR10 completed
```

TCPIPCS TSDB

Invocation of this subcommand displays the TSDB server data block.

Syntax

Following is the syntax of the TCPIPCS TSDB subcommand:



Parameters

Following is the parameter for the TCPIPCS TSDB subcommand:

TCP, TITLE, NOTITLE

See “Parameters” on page 89 for a description of these parameters.

Sample Output

The following is a 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
```

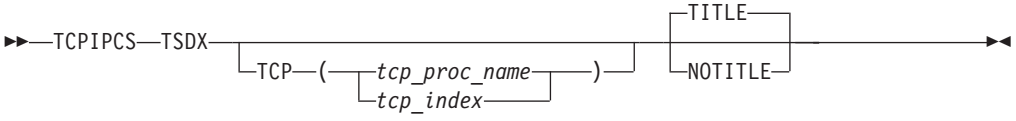
```
13391C00  1 TCPSVT    V2R10   1323B000 1323B0C8 07DE 04041405  Active
13391C80  2 TCPSVT2   V2R10   00000000 00000000 07E8 00000000  Down Stopping
13391D00  3 TCPSVT1   V2R10   12FC3000 12FC30C8 0080 94FF755F  Active
13391D80  4 TCPSVT3   V2R10   00000000 00000000 0059 00000000  Down Stopping

   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

Invocation of this subcommand displays the TSDX server data extension.

Syntax

Following is the syntax of the TCPIPCS TSDX subcommand:



Parameters

Following is the parameter for the TCPIPCS TSDX subcommand:

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Sample Output

The following is a sample output of the TCPIPCS TSDX subcommand.

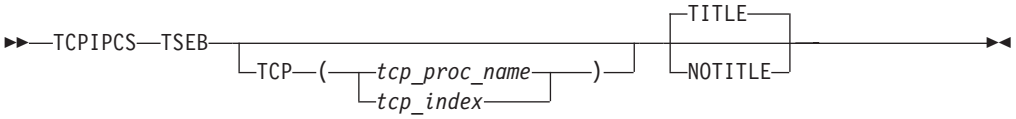
```
TCPIPCS TSDX
Dataset: IPCS.MV21381.DUMPA
Title:   SLIP DUMP ID=TC


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

Invocation of this subcommand displays the TSEB server anchor block.

Syntax

Following is the syntax of the TCPIPCS TSEB subcommand:



Parameters

Following is the parameter for the TCPIPCS TSEB subcommand:

TCP, TITLE, NOTITLE

See "Parameters" on page 89 for a description of these parameters.

Sample Output

The following is a 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
```

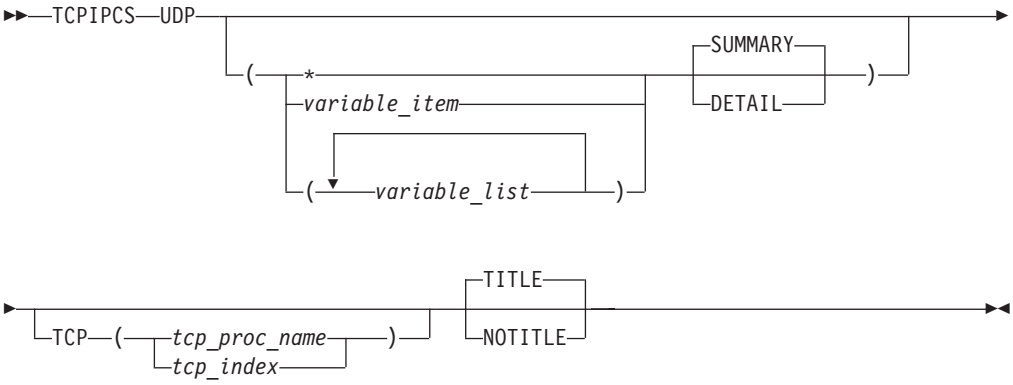
```
+0050  TSEB_BGRP_Q@..... 1320A608
+0054  TSEB_DUAF..... 130A4010
+0059  TSEB_TOKENID..... 000015
+005C  TSEB_TCMTPTR..... 132072F0
+0060  TSEB_EZBITCOM_LEN..... 00007D28
+0064  TSEB_CS390_VERSION.... 020A
+0068  TSEB_CSMFREE..... 1320A548
+006C  TSEB_TCPIP_STOKEN..... 00001F78  00000008
+0074  TSEB_CSMPACK..... 1320A588
+0078  TSEB_SOCA..... 140BAEB8
+007C  TSEB_CSMPACKQDIO..... 1320A5C8

Analysis of Tcp/Ip for TCPSVT completed
```

TCPIPCS UDP

Invocation of this command displays the Master UDP Control Block (MUCB) and any UDP Control Blocks (UCBs) defined in the UDP Patricia tree.

Syntax

Following is the syntax of the TCPIPCS UDP subcommand:



Parameters

If no parameters are specified, all UDP control blocks are summarized.

* An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, separated by a blank space, within parentheses:

Variable parameters are:

jobname
Displays only the UDP control blocks with this job name. The job name may be a TCP/IP application name or a stack name. A jobname 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 may 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, the following keyword parameters may be specified:

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 UCB(s).

TCP, TITLE, NOTITLE
See “Parameters” on page 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a sample output of the TCPIPCS UDP subcommand.

```
TCPIPCS UDP
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


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
 +0090  UIPWRQ@.. 7F712828  UIPRDQ@.. 7F7127E8  UMI@..... 00000000
```

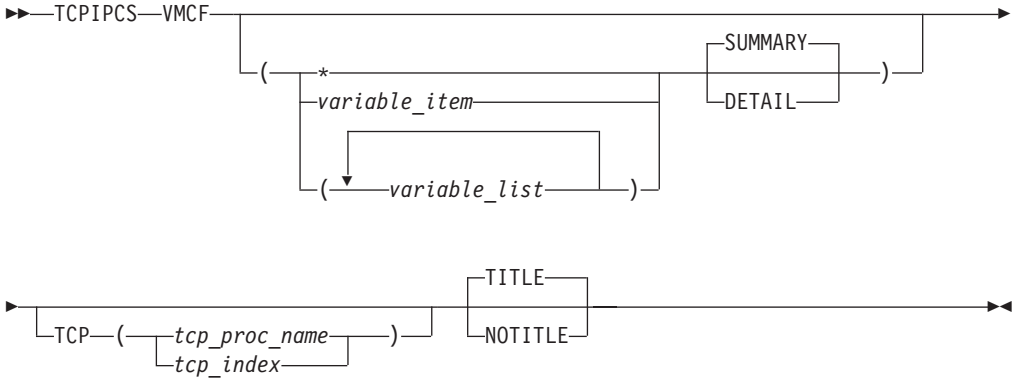
```
USMI@.... 00000000  UUCBSID@. 00000000
 +00A4  UPAUTLL@. 00000000  USNAWRQ@. 00000000  USNARDQ@. 00000000
 +00B0  UDMUX_TOKEN.                 7F781248  00000002
 +00B8  USNMP_TOKEN.                 7F7811E8  00000003
 +00C0  UDMULTI@. 00000000

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
7F5FCB08 0000004B CBSAMPLE WLOIDLE  0.0.0.0         30001
...
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 VMCF

Invocation of this command displays information about VMCF (Virtual Machine Communication Facility) and IUCV (Inter-User Communication Vehicle) users.

Syntax

Following is the syntax of the TCPIPCS VMCF subcommand:



Parameters

If no parameters are specified, all VMCF control blocks are summarized.

***** An asterisk is used as a place-holder if no variable parameters will be specified.

variable_item
Any one of the following variable parameters.

variable_list
From 1–32 of the following variable parameters may be repeated, separated by a blank space, within parentheses:

Variable parameters are:

user_id
Displays only the VMCF control block associated with this user ID. Specifed 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 may 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 1–4 hexadecimal digits.

In addition to the variable parameters described above, the following keyword parameters may be specified:

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 89 for a description of these parameters.

Note: If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a 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
```

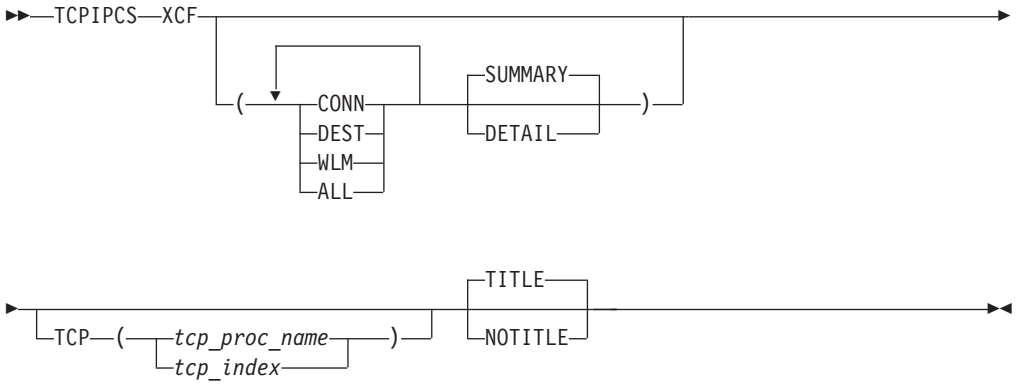
```
  TNF Manager Area    : 00008FE0
  SMSG Id             : 0

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

Analysis of Tcp/Ip for TCPCS completed
```

TCPIPCS XCF

Invocation of this command produces a cross-system coupling facility (XCF) analysis report.

Syntax

Following is the syntax of the TCPIPCS XCF subcommand:



Parameters

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

The following keyword parameters may be specified:

ALL
Display all optional information.

CONN
Only display connection hash table optional information. CONN is the default.

DEST
Only display destination hash table optional information.

WLM
Only display workload manager 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 89 for a description of these parameters.

Notes:

1. If you specify multiple keywords from the set {ALL, CONN, DEST, WLM}, all of them will be used.
2. If you specify multiple keywords from the set {SUMMARY, DETAIL}, only the last one will be used.

Sample Output

The following is a 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
================================================================================
----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

The ERRNO command searches for the name and description of constants used for ERRNO, ErrnoJr, module ID, reason code, and ABEND reason code.

Syntax

Following is the syntax of the ERRNO command:

Parameters

Following are the parameters of the ERRNO command:

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 in one to eight hexadecimal digits ending with a period. The value at that address will be interpreted.

hhhhhhhh
An ERRNO, ERRNO junior, reason code, ABEND code, or module ID in one to eight hexadecimal digits.

hhhhhhhhx
An ERRNO, ERRNO junior, or a module ID in one to eight hexadecimal digits followed by the letter x.

ddddddddn
An ERRNO, ERRNO junior, or a module ID in one to eight 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 will try to interpret the name as a hexadecimal value.

Sample Output

Following are some sample outputs of the ERRNO command.

- **Example 1:** Reason code by hexadecimal value.

```
Command ===>  errno r 74be72e9

    ReasonCode: 74BE72E9
      Module: EZBITSTO  ErrnoJr: 29417 JRCMNOCSM
      Description: Cache Manager encountered a CSM storage shortage
```

- **Example 2:** Reason code by address, where the value at address 07093F98 is 74717273. Note that 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
```

- **Example 3:** Errno in decimal.

```
    Command ===> errno e 129n

    Errno: 00000081(129) : ENOENT
      Description: No such file, directory, or IPC member exists
```

- **Example 4:** ErrnoJr in hexadecimal.

```
    Command ===> errno j 6c

    ErrnoJr: 0000006C(108) : JRFILENOTTHERE
      Description: The requested file does not exist
```

- **Example 5:** Abend code in decimal.

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

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

- **Example 6:** Module ID in hexadecimal.

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

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

- **Example 7:** Module name.

```
    Command ===> errno ezbifinb

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

- **Example 8:** ERRNO name.

```
    Command ===> errno ebadf

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

- **Example 9:** ErrnoJr name.

```
    Command ===> errno   jrmaxuids

    ErrnoJr: 00000013(19) : JRMAXUIDS
      Description: The maximum number of OpenMVS user IDs is exceeded
```

- **Example 10:** ABEND reason name.

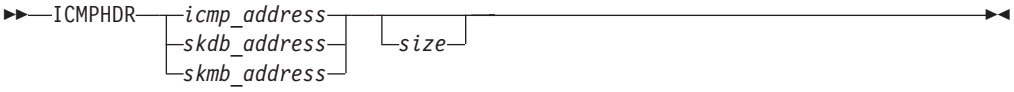
```
Command ===> errno tcpbadentrycode

Abend Reason Code: 00000401
  Module: Unknown  Reason: TcpBadEntryCode
  Description: Bad Entry code to module
```

ICMPHDR

Invocation of the ICMPHDR command displays the ICMP header fields.

Syntax

Following is the syntax of the ICMPHDR command:



Parameters

Following are the parameters of the ICMPHDR subcommand:

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 1–3 hexadecimal digits.

Sample Output

Following is a sample output of the ICMPHDR command.

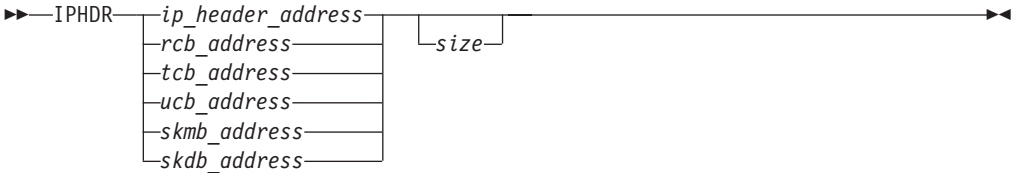
```
ICMPHDR 08D0A0D0

ICMP Header at 08D0A0D0

08D0A0D0  03032AFE  00000000  45000047  00600000  | .............-.. |
   +0010  401180DB  7F000001  09437127  040700A1  |  ..."..........  |
   +0020  0033CD23  00000000  00000000  4500003C  | .................|
   +0030  00010000  40067CB9  7F000001            | .... .@."...     |

  Type                : UNREACH
  Code                : PORT
  Checksum            : 2AFE
  Misc. Header        : 0
  Misc. Data          : 45000047 00600000 401180DB 7F000001
```

IPHDR

Invocation of the IPHDR command displays the IP header fields.

Syntax

Following is the syntax of the IPHDR subcommand:



Parameters

Following are the parameters of the IPHDR subcommand:

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 1–3 hexadecimal digits.

Sample Output

The following is a sample output of the IPHDR command.

```
IPHDR 8D0A0D8

IPv4 Header at 08D0A0D8

08D0A0D8  45000047  00600000  401180DB  7F000001  | .....-.. ..."... |
   +0010  09437127                                | ....              |

  IP Version          : 4
  Header Size         : 20
  Precedence          : Routine
  TOS                 : NormalService
  Packet Size         : 71
  ID Number           : 96
  Fragment            : ^DontFragment ^MoreFragment
  Offset              : 0
  TTL                 : 64
  Protocol            : UDP
  Checksum            : 80DB
  Source              : 127.0.0.1
  Destination         : 9.67.113.39

UDP Header at 08D0A0EC
```

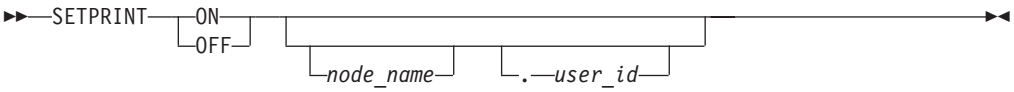
```
08D0A0EC  040700A1  0033CD23                    | ...~....        |

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

SETPRINT

Invocation of the SETPRINT command may 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 will be accumulated (not displayed at the terminal) until you exit IPCS. When you exit IPCS, the IPCSPRNT data set will be sent to the specified node.

Syntax

Following is the syntax of the SETPRINT command:



Parameters

Following are the parameters of the SETPRINT command:

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 will be sent.

user_id
The user ID on the TSO or VM system to which the output will be 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:

- **Example 1:** Allocating IPCSPRNT when it is already allocated.

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

- **Example 2:** Freeing IPCSPRNT when it is already freed.

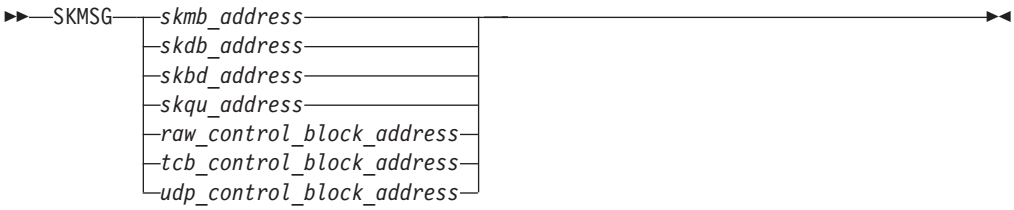
```
setprint off
 BLS21060I PRINT file not open
 IKJ56247I FILE IPCSPRNT NOT FREED, IS NOT ALLOCATED
```

SKMSG

Invocation of the SKMSG command displays the SKMSG fields.

Syntax

Following is the syntax of the SKMSG command:



Parameters

Following are the parameters of the SKMSG command:

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.

Sample Output

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
```

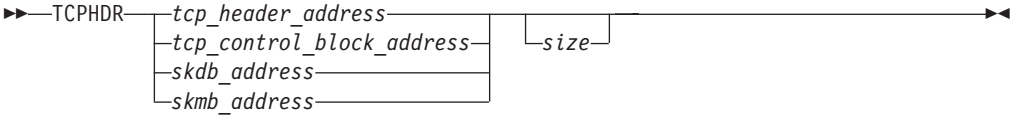
```
   +001C  base..... 15DE5000  size..... 00001000  flag..... 00000000
   +0028  ref...... 00000005

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

Invocation of the TCPHDR command displays the TCP header fields.

Syntax

Following is the syntax of the TCPHDR command:



Parameters

Following are the parameters of the TCPHDR subcommand:

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 1–3 hexadecimal digits.

Sample Output

The following is a sample output of the TCPHDR command.

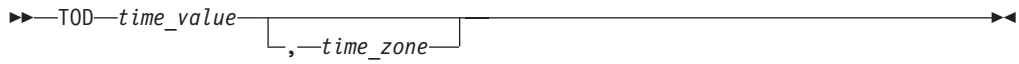
```
TCPHDR 7F522108

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

Invocation of the TOD command formats a hexadecimal time-of-day value into a readable date and time.

Syntax

Following is the syntax of the TOD command:



Parameters

Following are the parameters of the TOD subcommand:

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 8 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 an 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

The following are sample outputs of the TOD command.

- **Example 1:** 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
```

- **Example 2:** 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

Invocation of the UDPHDR command displays the UDP header fields.

Syntax

Following is the syntax of the UDPHDR command:

Parameters

Following are the parameters of the UDPHDR subcommand:

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 will format 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.

Sample Output

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 has two parts:

1. Install the members of the target data sets.
2. Optionally, connect the TCP/IP panels to your existing ISPF panels.

Table 15 shows the target data sets that contain the data necessary to set up the TCP/IP IPCS subcommands. You need to copy the members to another data set or concatenate the target data sets into the DDNAME statements shown. Note that the target data sets contain other members, so you may not want to simply concatenate the target dataset. 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
SYS1.SEZAHELP	EZBIPCSH	SYSHELP	TCPIPCS HELP command text
SYS1.PARMLIB	EZAIPCSP	PARMLIB	IPCS verbexit mappings
SYS1.SEZAINST	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)	SYSEXEC	REXX execs
SYS1.SEZAMIG	EZBDGIPC	STEPLIB or IPCS TASKLIB	Load module
SYS1.SEZAPENU	EZBD* (approximately 170 members)	ISPPLIB	ISPF panels
SYS1.SEZAPENU	EZBDKEYS	ISPTLIB	ISPF key lists

Table 15. Target Data Sets for TCP/IP IPCS Subcommands (continued)

Target Data Set	Members	DD Name	Description
SYS1.SEZAMNU	EZBF* (6 members)	ISPMLIB	ISPF messages

To use the panel interface to the TCP/IP IPCS subcommands, you can either invoke the panels via 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 via 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 will invoke 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

The TCP/IP IPCS subcommands can be entered as an IPCS command or by using the panels provided by TCP/IP. 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 13 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   CS/OS390    - VTAM & 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 13. IPCS Primary Option Menu

3. Select option 4, `COMMAND`.
4. Type the TCP/IP IPCS subcommand. Figure 14 on page 163 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 14. IPCS Subcommand Entry panel with a TCP/IP IPCS subcommand entered.

There are two ways to invoke the TCP/IP IPCS panels.

- One way is to 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
  ```

- The second way to invoke the TCP/IP IPCS panels is to select the option provided in the installation section above.

Either way you invoke the panels, you should see the main menu for the TCP/IP IPCS commands shown in Figure 15. Select an option and the panels will 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 will be invoked using the current default dump data set.

```
                          TCP/IP Analysis Menu
Command ===>
(C) Copyright IBM Corporation 1998,2000. 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   . - LOCK, MAP, STORAGE, TIMER
    5.  Execution . . - DUAF, DUCB, TRACE
    6.  Interfaces  . - API, SOCKET, STREAM
    7.  Structures  . - HASH, TREE
    8.  Functions . . - FIREWALL, FRCA, POLICY, TELNET, VMCF, XCF
    9.  Headers . . . - ICMPHDR, IPHDR, SKMSG, TCPHDR, UPDHDR
    10. Converters  . - ERRNO, SETPRINT, TOD


F1=Help    F2=Select  F3=Exit    F9=Swap    F12=Cancel
```

Figure 15. Main menu for TCP/IP IPCS Subcommands.

Part 3. Diagnosing CS for OS/390 Components

Chapter 7. 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 may 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 *OS/390 IBM Communications Server: IP Configuration Reference*. For information on using the client-related LPR, LPQ, and LPRM commands, refer to the *OS/390 IBM Communications Server: IP User's Guide*.

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 may 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, recreate the abend or wait for it to occur again. For information about analyzing dumps produced during LPD processing, refer to *OS/390 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)

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.

One or more of the following problems can cause 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

If a timeout, hang, or wait occurs, do one or more of the following.

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 may 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.

Garbled Data

If garbled data is the problem, do one or more of the following:

1. Check 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.

Truncated Or Missing Print Data

If truncated or missing print data is the problem, do one of the following:

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 may 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.

LPR Works with Some Options Only

If the LPR command works with some options, but not with others, do one or more of the following:

- If some print requests do not work with certain LPR options, check the LPR trace for error messages.
- 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 *OS/390 IBM Communications Server: IP User's Guide*.

LPR Client Traces

This section provides information about activating LPR client traces. It also provides samples of trace output with explanations of selected messages.

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, to start the LPR client with trace on, enter the following command:

```
LPR filename (Printer p1 Host h1 TRACE
```

Client Trace Output

LPR trace output is sent to SYSOUT and can be displayed on the LPR client console. Figure 16 on page 171 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 16. 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 16. 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 17 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 which shows that this job was rejected, see Figure 22 on page 183.) 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 *OS/390 IBM Communications Server: IP Configuration Reference*.

The trace was produced using the following command:

```
LPR test (p TIANNA h 9.67.113.60 filter x TRACE
```

issued through TSO by user ID TCPUSR4.

```
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 17. 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 22 on page 183.)

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 18 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 18. 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 19 on page 175 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 19. 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 *OS/390 IBM 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 20 on page 176 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 *OS/390 IBM 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 20. 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 errors, data corruption, and so on, 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 and 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 *OS/390 IBM Communications Server: SNA Messages*.

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.

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.

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 21 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 21. 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 21. 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 21. 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 21. 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 Not ification Other external interrupt received (8688)
 21
EZB0622I Terminated by external interrupt
```

Figure 21. 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 may 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 will scan 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 22 on page 183 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 17 on page 173 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 22. Example of a 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 22. Example of a 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 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 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 23 on page 186 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 23. Example of a 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 23. Example of a 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 23. Example of a 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 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.

Note: 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 8. Diagnosing File Transfer Protocol (FTP) Problems

This chapter describes how to diagnose problems with the CS for OS/390 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 213 and “Documenting FTP Client Problems” on page 218.

FTP Server

This section describes the following topics:

Structural Overview

The OS/390 UNIX FTP server has a multiprocess design. The main FTP daemon performs initialization and configuration functions, and then waits for incoming client connections. When an incoming client connection is received, a new process is forked, and a second load module, which provides the actual session support for the client connection, is loaded into the new address space.

The socket connection information and configurable variables are passed to the new address space so the variables in the client session program will have the same value as the variables in the main daemon process at the time the new client connection is received. (In other words, the client session program has the *current* settings of all variables, not the initial settings.)

After the new client process has been created, the main daemon process and all client processes are totally independent from one another. Any changes to the variables in the main daemon process, or in other client processes, are not reflected in any other process.

After the new client process is created, the client is prompted for a user ID and password. The C Runtime Library function *seteuid()* sets the effective user ID for the new address space to the client user ID.

Definitions and Setup

This section describes the definitions and setup for the FTP server.

Start Procedure

The start procedure for the FTP server is EZAFTPAP (alias FTPD) in the *hlq*.SEZAINST data set. Changes might be necessary to customize the start procedure for your MVS host system. The following should be kept 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 runtime library is not in the MVS link list, it must be included on the STEPLIB DD statement.
- If the FTP server will be 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 will 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 *OS/390 IBM 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 *OS/390 IBM Communications Server: IP Configuration Reference* for more information about the FTP.DATA data set.

TCPIP.DATA Data Set

The TCPIP.DATA data set provides information to the FTP server, such as the high-level qualifier to be used for configuration data sets, whether messages are to be written in uppercase or mixed case, and which DBCS translation tables are to be used. For more information about the TCPIP.DATA data set, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.

Error Exit Codes

OS/390 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 sessions 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 OS/390 UNIX FTP

This section explains the MVS and HFS naming conventions.

MVS Naming Conventions

MVS data set names used with all FTP commands sent to the OS/390 UNIX FTP server must meet MVS data set naming conventions as follows:

- Data set names can be no 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 current working directory and the path name cannot be longer than 44 characters. The current working directory can be displayed by issuing the PWD command.
- Each qualifier in a data set name, or each member name for a partitioned data set, must conform to the following:
  - It must be no longer than eight characters.

  - It must begin with a letter or the special characters $, @, or #.
  - It can 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

The following list describes some naming conventions you should know about when using HFS files with the OS/390 UNIX FTP server:

- The HFS name is case-sensitive.
- If a name begins with a single quote, specify `QUOTESOVERRIDE FALSE` in FTP.DATA, or use the `SITE NOQUOTESOVERRIDE` command.
- Names may contain imbedded blanks for special characters. Be aware that:
  - The subcommand is interpreted as:

    ```
    <ftp_subcommand><one blank space><pathname><new_line>
    ```

  - Some FTP clients may 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 *OS/390 IBM Communications Server: IP User's Guide*.
- The START and SITE parameters have additional restrictions on the path name used with SBDATACONN. Refer to *OS/390 IBM Communications Server: IP Configuration Reference* and *OS/390 IBM Communications Server: IP User's Guide*.
- When specifying a OS/390 UNIX FTP subcommand with a file name containing special characters, some FTP clients may truncate trailing blanks, compress multiple internal blanks, or interpret special characters to have special meanings. Special specification of the file name such as enclosing in double or single quotes, or escaping special characters, may be necessary to get the client to send the file name to the server correctly. Refer to your client documentation to see if this is necessary.

Common OS/390 UNIX FTP Problems

This section describes some common OS/390 UNIX FTP problems.

FTP Daemon Initialization Problems

The following problems may be encountered 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 job daemon job is running (for example, FTPD1), verify that TCP/IP is running. If it is not, start TCP/IP. The FTP initialization will complete 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. Following are the 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 trace 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 jobname. 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 syslogd output to verify that configuration values are coming from your intended FTP.DATA file and that no errors were encountered.
- Check if your FTP.DATA file has sequence numbers. If it does, any statement with an optional parameter omitted will pick 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 problem is that 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 parm
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 is fewer than eight characters, ensure that the AUTOLOG and PORT statements in the *hlq*.PROFILE.TCPIP data set specify the FTP background jobname. 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
```

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 *OS/390 UNIX System Services Planning*, or refer to the RACF publications.
- Other

  Check the system and language environment (LE) library levels in the EZAFT09I and EZAFT5II messages in syslogd output. The former should read "Version 2 Release 6"; the latter should read `Version 2 Release 4, or higher`.

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 may 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 OS/390 search order.

If your system is not configured to display exit codes, look in 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 and 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 last FTP entry in the trace reads `RA0`*`nnn`* `pass: termid is ....`

Connection Terminated by the Server After User Enters Any Subcommand: If the server terminates a connection after the user enters a subcommand, either one or both of the following events may occur:

- FTP server address space shows an exit code of `0000`.
- Last entry for the client session is `RX`*`nnnn`* `Server thread terminates rc = -2.` and 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, increase it, if necessary, and recycle the FTP daemon.

Password Validation Fails; Session Continues: If password validation fails and the session continues, you may receive one of the following two replies from the FTP server:

- `550 error processing PASS command: <reason>`

  If you receive this message, do one or more of the following:

  – Ensure all libraries 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 OS/390 UNIX shell.

- `PASS command failed - getpwnam() error: <reason>`

  If you receive this message, ensure that the login user ID has an OMVS segment defined or that a default OMVS segment is established.

Anonymous Logon Fails: If an anonymous logon fails:

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 will be 1 (one).

If ANONYMOUS is set in FTP.DATA, and the STARTDIRECTORY is HFS, and ANONYMOUSLEVEL is 2 or 3, verify the required executable files have been installed in the anonymous user's root directory. SYSLOGD will have error messages if the required executable files are not installed in the anonymous user's home directory. For information on setting up the anonymous user's root directory, see *OS/390 IBM 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 on the OS/390 UNIX environment and its security considerations, refer to *OS/390 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 2 or 3, verify 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 one of the following is coded: ANONYMOUS or ANONYMOUS/USERID/PASSWORD, the user will be prompted to enter an e-mail address as a password. Verify the email address entered by the user is consistent with the requirements of the EMAILADDRCHECK statement in FTP.DATA. If ANONYMOUS/USERID is coded, then the user must provide the password for USERID. See the *OS/390 IBM Communications Server: IP Configuration Reference* for more information on 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. 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. Following is an 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, ensure that FTP has BPX.DAEMON authority.

Data Transfer Problems

This section describes various problems involving data transfer.

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 V2R10 level or higher.
  - For proxy transfers, both servers and the client must be V2R10 or higher.
- Ensure that the user did not attempt an operation which 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 may only be transferred between the same types of load library. 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 PDS. 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-Local directory might be a load library` reply to the client when a CWD command is processed which 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 will NOT be used to transfer any files between the two directories.
  - Ensure that the load module(s) 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 membernames 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 will furnish the IEBCOPY SYSPRINT output as messages to either the console (in the server's case) or the terminal session (in the client's case). If no IEBCOPY errors are detected by FTP, then the IEBCOPY SYSPRINT output is provided with the debug 1 trace.

If the MVS load module fails to transfer:

- 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 1 trace to determine if IEBCOPY was successfully invoked (some client environments, particularly REXX scripts running under the UNIX system services shell, are not fully authorized to call IEBCOPY). If so, examine the IEBCOPY SYSPRINT output (described above) to see if IEBCOPY reported any errors.
- If ″allocation failure″ messages or replies are seen then:
  - If the dataset 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 dataset name appears, or if the dataset 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 will regulate whether or not FTP attempts to mount addtional temporary storage to complete a load module temporary file allocation request.

If the MVS load module transfer hangs, then most likely the system is 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 on 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, 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 PRIMARY and SECONDARY both either specified or not specified? If either PRIMARY or SECONDARY are specified, neither of the values will be 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 will be. The VOLUME and UNIT value of the STAT command will tell you where the data sets are being allocated. (If neither volume or unit are shown by the STAT command, data sets will be allocated on the system default SYSDA DASD.) Does the server system have sufficient space where the data sets will be allocated to allocate the data set? The SITE QDISK command will provide information about the space available at the server system.
- Is the destination at the server site 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 200. 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 both 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, 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 will be 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 both 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 for 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 should be 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 will override 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.
   - Check for variables overridden by a client. The VM and MVS FTP clients both 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:

- Issue the **DIR** command to display the data set. Can the server find the data set to list it?
- Is the MVS data set at the server in the catalog? The server can only locate cataloged MVS data sets.
- 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:

- Is the data set protected by a security system, such as RACF or HFS permission bits or a retention period?
- Is the data set “in use” at the server site by another program or user?
- Was the data set available to the system, or was it migrated or on an unmounted volume?
- Did the data set or member exist?

The following problems apply to MVS data sets only:

- Did the specified path name follow MVS data set naming conventions?
- Was the requested data set a supported data set organization (PS, PDS, or PDS member) on a supported device type (DASD or tape)?
- 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?

Data Transfer Terminated: Check for the following problems:

- 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.

- 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?
- Did the client send an ABOR command?
- 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 will attempt 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 cataloged if a store operation terminates abnormally`, the server will catalog 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 will be 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 will keep the data set even if the conditional disposition has been 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 terminates abnormally`, the server will delete 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.

The client you are using must support the SDIR command.

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 RESOLVER_CONFIG environment variable or /etc/resolv.conf takes precedence over DD:SYSTCPD or *jobname*.TCPIP.DATA.

DB2 Query Support

This section describes how to use FTP server DB2® query support and how to diagnose SQL problems.

How to use FTP Server SQL Support: 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 OS/390 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: The following two tables shows some symptoms and possible causes of SQL problems. Table 16 on page 203 shows problems that generate a reply beginning with 55x.

Table 16. SQL Problems Generating 55x Replies

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 will fail.
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 DBZPLAN 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. Can't 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 *OS/390 IBM Communications Server: IP and SNA Codes*.		

Table 17 shows other SQL problems.

Table 17. Other SQL Problems

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 will indicate 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): If the server is in JESINTERFACELEVEL=1 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 following:

- Is the job name correct? The job name must be the user ID followed by a single character.
- Was the job output spooled to the hold queue? The server will only be able to retrieve job output that is in the hold queue.

For example: If JESINTERFACELEVEL=2 then make sure 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. For example, if the JESJOBNAME=USER1* and the job submitted was USER2A, then 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 *OS/390 IBM Communications Server: IP User's Guide* 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 JTRACE, JDUMP, or UTRACE option on the MODIFY command.

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, 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 OS/390 search order.

Messages and Trace Entries

If messages and trace entries are lost, 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 and/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).

Diagnosing FTP Server Problems with Traces

Several types of traces are available to aid in debugging OS/390 UNIX FTP server problems. Tracing of OS/390 UNIX FTP server initialization is controlled by the TRACE start parameter or the TRACE keyword in FTP.DATA. (See “Start Tracing” on page 206.) When initialization is complete, tracing can be controlled by options on the MODIFY command. (See “Controlling the FTP Server Traces with MODIFY” on page 207.)

You can trace activity for all clients connecting to the FTP server. When tracing for all clients, you can choose to trace general FTP activity, or JES-related activity, or

both. You also can choose the level of detail to be included in the trace log. See “Tracing Activity for All Clients” on page 207.

Alternatively, you can trace all activity for a single user ID. See “Tracing Activity for One User ID” on page 207.

Where to Find Traces

The OS/390 UNIX FTP server sends its trace entries to syslogd. 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 of OS/390 UNIX FTP trace entries (general or JES-related) are written to the same HFS file.

Note: The TRACE parameter and MODIFY options are issued to the FTP daemon and affect all client sessions that connect to the OS/390 UNIX FTP server while tracing is active.

Refer to the *OS/390 IBM Communications Server: IP Configuration Guide* for more information about syslogd.

Start Tracing

This section discusses two 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 or the TRACE statement in FTP.DATA to begin tracing during FTP daemon initialization. This continues tracing for all FTP events (except JES-related 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 command to end it. Refer to “Controlling the FTP Server Traces with MODIFY” on page 207.

Note: When you issue a MODIFY command to end tracing, tracing does not occur for any subsequent client sessions; however, tracing will continue for any sessions that were already connected.

Start Tracing After FTP Initialization: After initialization, you can enable tracing using an MVS MODIFY command to the FTP server listener process. Refer to “Controlling the FTP Server Traces with MODIFY” on page 207. Already established FTP connections are not affected by a MODIFY command. Only FTP connections that are established after the MODIFY command was issued will be subject to tracing.

Stop Tracing

You stop global tracing using the MODIFY command (for example: `F FTPD1,NOTRACE`). This works assuming you started your FTP listener process by way

of a PROCLIB member with the name FTPD, which means the process to modify is FTPD1. You stop tracing for a single user using the command `F FTPD1,NOUTRACE`. Already 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.

Tracing Activity for All Clients

Use the following options to control general and JES tracing for all user IDs. Both general and JES tracing can be active at the same time.

- The TRACE option begins the general trace for the FTP server. The general trace provides information, such as logical paths followed and error conditions encountered, for all clients connecting to the FTP server.
- The NOTRACE option ends general tracing.
- The JTRACE option begins the JES trace for the FTP server. The JES trace provides information, such as logical paths followed and error conditions encountered for JES-related activity, for all clients connecting to the FTP server.
- The NOJTRACE option ends JES tracing.

The general trace and JES trace are global traces, that is, they provide trace data for all user IDs. Additional options control the recording of detailed data, such as parameter lists and storage areas, in the trace logs.

- The DUMP option specifies that additional detailed data is to be included whenever the general trace (TRACE) is active.
- The NODUMP option specifies that detailed data is to be excluded from the trace log. This is the default.
- The JDUMP option specifies that additional detailed data is to be included whenever the JES trace (JTRACE) is active.
- The NOJDUMP option specifies that detailed data is to be excluded from the JES trace log. This is the default.

Tracing Activity for One User ID

Trace data for a specified user ID includes both general and JES-related activity and includes data such as parameter lists and storage areas. User tracing is controlled by the following options:

- The UTRACE option begins tracing for the specified user ID. Only one user ID can be traced at a time. All other tracing options in effect are suspended when the user trace is started and resumed when the user trace is stopped. No new trace options can be entered while the user trace is in effect, with the exception that a new user trace can be entered to change the user ID that is to be traced.
- The NOUTRACE option ends tracing for a specified user ID. Any global tracing options (for the general or JES traces) that were in effect when user tracing was begun are resumed when user tracing is ended.

Controlling the FTP Server Traces with MODIFY

The general trace for the FTP server can be started for all user IDs during initialization by specifying the TRACE parameter either as a start option in the FTP server start procedure or in the FTP.DATA data set.

Alternatively, tracing options can be started or stopped after initialization by issuing one of the following MVS MODIFY commands to the FTP server jobname.

Note: The *jobname* is the name associated with the FTP daemon background job. It is documented in EZYFT41I message in the FTP services log. If you started the OS/390 UNIX server using a proc named FTPD, the jobname to use for the MODIFY command is probably FTPD1. As client sessions

connect to the FTP server, the session process adapts the trace options currently active. These options remain in effect for the life of the client session process, regardless of subsequent MODIFY commands issued to the FTP daemon.

Following are some examples of the MODIFY command used to control OS/390 UNIX FTP server traces.

- `MODIFY ftp_server_jobname,TRACE`

  Starts the FTP server general trace for all user IDs. This option can be issued anytime, except when the user trace is active.

- `MODIFY ftp_server_jobname,NOTRACE`

  Stops the FTP server general trace for all user IDs. This option can be issued anytime, except when the user trace is active.

- `MODIFY ftp_server_jobname,JTRACE`

  Starts the FTP server JES trace for all user IDs. This option can be issued anytime, except when the user trace is active.

- `MODIFY ftp_server_jobname,NOJTRACE`

  Stops the FTP server JES trace for all user IDs. This option can be issued anytime, except when the user trace is active.

- `MODIFY ftp_server_jobname,UTRACE=user_id`

  Starts the FTP server trace for the specified user ID.

  If a previous user trace was in effect, the user ID previously being traced will no longer be traced. The new user ID will be traced instead.

  If other trace options were in effect when the user trace was started, they will be suspended until the user trace is stopped.

  Any MODIFY commands to start other traces will be ignored while the user trace is in effect.

- `MODIFY ftp_server_jobname,NOUTRACE`

  Stops the FTP server user trace.

  Restores any global trace options (for the general or JES traces) that were in effect at the time the UTRACE option was entered to start tracing a specified user ID.

- `MODIFY ftp_server_jobname,DUMP`

  Causes detailed data, such as parameter lists and storage areas, to be included in the general trace log.

  This option can be issued before or after general tracing has been started. If it is issued while general tracing is active, it takes effect immediately. This option cannot be entered while a user trace is active.

- `MODIFY ftp_server_jobname,NODUMP`

  Causes detailed data to be excluded from the general trace log.

  This is the default.

  This option can be issued before or after general tracing has been started. If issued while general tracing is active, it takes effect immediately. This option cannot be entered while a user trace is active.

- `MODIFY ftp_server_jobname,JDUMP`

  Causes detailed data, such as parameter lists and storage areas, to be included in the JES trace log.

This option can be issued before or after JES tracing has been started. If issued while JES tracing is active, it takes effect immediately. This option cannot be entered while a user trace is active.

- MODIFY *ftp_server_jobname*,NOJDUMP

  Causes detailed data to be excluded from the JES trace log.

  This is the default.

  This option can be issued before or after general tracing has been started. If issued while general tracing is active, it takes effect immediately. This option cannot be entered while a user trace is active.

Trace Examples and Explanations

Figure 24 shows an example of a trace of an OS/390 UNIX FTP server. (Timestamps have been removed for clarity.) A trace explanation is given in "Trace Output Explanation" on page 212.

```
1  ftpd[33554440]: EZYFT18I Using catalog '/usr/lib/nls/msg/C/ftpdmsg.cat' for FTP messages.
2  ftpd[33554440]: EZY2697I IBM FTP CS V2R10 13:59:35 on 03/05/98
3  ftpd[33554440]: EZY2693I Unable to open DD:SYSFTPD : EDC5129I
     No such file or directory.
   ftpd[33554440]: EZY2693I Unable to open //'FTPD.FTP.DATA' : EDC5049I
     The specified file name could not be located.
   ftpd[33554440]: EZY2693I Unable to open /etc/ftp.data : EDC5129I
     No such file or directory.
   ftpd[33554440]: EZY2693I Unable to open SYS1.TCPPARMS(FTPDATA): EDC5067I
     An attempt was made to open a nonexistent file for read.
   ftpd[33554440]: EZY2693I Unable to open //'TCPIP.FTP.DATA' : EDC5049I
     The specified file name could not be located.
4  ftpd[33554440]: EZY2638I Using FTP configuration defaults.
5  ftpd[33554440]: EZY2693I Unable to open 'FTPD.SRVRFTP.TCPXLBIN' : EDC5049I
     The specified file name could not be located.
   ftpd[33554440]: EZY2693I Unable to open 'TCPIP.SRVRFTP.TCPXLBIN' : EDC5049I
     The specified file name could not be located.
   ftpd[33554440]: EZY2693I Unable to open 'FTPD.STANDARD.TCPXLBIN' : EDC5049I
     The specified file name could not be located.
   ftpd[33554440]: EZY2693I Unable to open 'TCPIP.STANDARD.TCPXLBIN' : EDC5049I
     The specified file name could not be located.
6  ftpd[33554440]: EZYFT26I Using 7-bit conversion derived from 'ISO8859-1'
     and 'IBM-1047' for the control connection.
7  ftpd[33554440]: EZYFT33I Unable to open DDNAME 'SYSFTSX' for the data connection:
     EDC5129I No such file or directory.
   ftpd[33554440]: EZY2693I Unable to open 'FTPD.SRVRFTP.TCPXLBIN' : EDC5049I
     The specified file name could not be located.
   ftpd[33554440]: EZY2693I Unable to open 'TCPIP.SRVRFTP.TCPXLBIN' : EDC5049I
     The specified file name could not be located.
   ftpd[33554440]: EZY2693I Unable to open 'FTPD.STANDARD.TCPXLBIN' : EDC5049I
     The specified file name could not be located.
   ftpd[33554440]: EZY2693I Unable to open 'TCPIP.STANDARD.TCPXLBIN' : EDC5049I
     The specified file name could not be located.
8  ftpd[33554440]: EZYFT32I Using the same translate tables for the control
     and data connections.
   ftpd[33554440]: DM0685 main: __ipdspx returned TCPIP
9  ftpd[33554440]: EP4982 set_dbcs: __ipdbcs() returned 0 parms from LOADDBCSTABLES statement(s)
10 ftpd[33554441]: DM0794 main: attempt OS/390 registration...
   ftpd[33554441]: DM0814 main: back from ifaedreg.  rc = 0
   ftpd[33554441]: DM0880 main: NLSPATH environment variable not defined. Using default nslpath.
   ftpd[33554441]: S.0885 main: Using /usr/lib/nls/msg/%L/%N for nlspath.
   ftpd[33554441]: DM0912 main: LANG environment variable not defined.  Using default lang value.
   ftpd[33554441]: S.0917 main: Using C for lang value.
```

Figure 24. Example of a Trace of an OS/390 UNIX FTP Server (Part 1 of 4)

```
11  ftpd[33554441]: DM0929 main: a2e table for control connection
   ftpd[33554441]: 09647307  00010203  372D2E2F  1605150B  0C0D0E0F   *................*
   ftpd(33554441): 09647317  10111213  3C3D3226  18193F27  1C1D1E1F  *................*

.
.
.
   ftpd[33554441]: DM0931 main: e2a table for control connection
   ftpd[33554441]: 09647407  00010203  1A091A7F  1A1A1A0B  0C0D0E0F   *................*
   ftpd[33554441]: 09647417  10111213  1A0A081A  18191A1A  1C1D1E1F   *................*

.
.
.
   ftpd[33554441]: DM0934 main: a2e table for data connection
   ftpd[33554441]: 09647107  00010203  372D2E2F  1605150B  0C0D0E0F   *................*
   ftpd[33554441]: 09647117  10111213  3C3D3226  18193F27  1C1D1E1F   *................*

.
.
.
   ftpd[33554441]: DM0936 main: e2a table for data connection
   ftpd[33554441]: 09647207  00010203  1A091A7F  1A1A1A0B  0C0D0E0F   *................*
   ftpd[33554441]: 09647217  10111213  1A0A081A  18191A1A  1C1D1E1F   *................*

.
.
.
   ftpd[33554441]: DM0944 main: msgcat for server msgs is; 157655072
12  ftpd[33554441]: DM0991 main: ret code from fndmembr() for FTCHKIP is: 4
   ftpd[33554441]: DM1003 main: user exit FTCHKIP not found. Bypassing fetch().
   ftpd[33554441]: EZYFT09I system information for MVSJ: OS/390 version 02 release 05.00 (9021)
   ftpd[33554441]: DM1034 main: __librel returns 22040000
   ftpd[33554441]: EZYFT51I OS/390 version 2, release 04, modification 0000.
13  ftpd[33554441]: DM1066 main: Initialization parameter values:
   ftpd[33554441]: DM1068 ..localsite values ------------------------
   ftpd[33554441]: DM1070 ....asatrans..................0
   ftpd[33554441]: DM1072 ....automount.................1
   ftpd[33554441]: DM1074 ....autorecall................1
   ftpd[33554441]: DM1076 ....blocksize.................6233
   ftpd[33554441]: DM1078 ....ckpt_interval.............0
   ftpd[33554441]: DM1080 ....conddisp..................C
   ftpd[33554441]: DM1082 ....dataclass.................
   ftpd[33554441]: DM1084 ....db2name...................DB2
   ftpd[33554441]: DM1086 ....dcbdsn....................
   ftpd[33554441]: DM1088 ....destuser..................
   ftpd[33554441]: DM1090 ....directory.................27
   ftpd[33554441]: DM1092 ....directorymode.............0
   ftpd[33554441]: DM1094 ....filetype..................1
   ftpd[33554441]: DM1096 ....imbedrdw..................0
   ftpd[33554441]: DM1098 ....jeslrecl..................80
   ftpd[33554441]: DM1100 ....jesrecfm..................128
   ftpd[33554441]: DM1102 ....lrecl.....................256
   ftpd[33554441]: DM1104 ....mgmtclass.................
   ftpd[33554441]: DM1106 ....migratevol................MIGRAT
   ftpd[33554441]: DM1108 ....primary...................1
   ftpd[33554441]: DM1110 ....quote_override............1
   ftpd[33554441]: DM1112 ....spread....................0
   ftpd[33554441]: DM1114 ....recfm.....................80
   ftpd[33554441]: DM1116 ....retpd.....................-1
   ftpd[33554441]: DM1118 ....secondary.................1
```

Figure 24. Example of a Trace of an OS/390 UNIX FTP Server (Part 2 of 4)

```
ftpd[33554441]: DM1120 ....spacetype.................3
ftpd[33554441]: DM1122 ....sqlcol....................N
ftpd[33554441]: DM1124 ....storclass.................
ftpd[33554441]: DM1126 ....trailing_blanks...........0
ftpd[33554441]: DM1128 ....ucshostcs.................IBM-1047
ftpd[33554441]: DM1130 ....ucssub....................0
ftpd[33554441]: DM1132 ....ucstrunc..................0
ftpd[33554441]: DM1134 ....unitname..................
ftpd[33554441]: DM1136 ....unit_is_tape..............0
ftpd[33554441]: DM1138 ....volser....................
ftpd[33554441]: DM1140 ....wraprecord................0
ftpd[33554441]: DM1142 ..other values ----------------------------
ftpd[33554441]: DM1144 ....smfappe...................0
ftpd[33554441]: DM1146 ....smfdel....................0
ftpd[33554441]: DM1148 ....smfjes....................0
ftpd[33554441]: DM1150 ....smflogn...................0
ftpd[33554441]: DM1152 ....smfren....................0
ftpd[33554441]: DM1154 ....smfretr...................0
ftpd[33554441]: DM1156 ....smfsql....................0
ftpd[33554441]: DM1158 ....smfstor...................0
ftpd[33554441]: DM1160 ....acceptanonymous...........0
ftpd[33554441]: DM1162 ....auto_tape_mount...........1
ftpd[33554441]: DM1164 ....DB2plan...................EZAFTPMQ
ftpd[33554441]: DM1166 ....inactivetime..............300
ftpd[33554441]: DM1168 ....jespgto...................600
ftpd[33554441]: DM1170 ....jobname...................FTPD1
ftpd[33554441]: DM1172 ....runtimedebug..............1
ftpd[33554441]: DM1174 ....server_port...............621
ftpd[33554441]: DM1176 ....startDirectory............1
ftpd[33554441]: DM1179 main: daemon's code page is: IBM-1047
ftpd[33554441]: DM1182 main: daemon's locale is: C
ftpd[33554441]: EZY2700I Using port FTP control (621)
ftpd[33554441]: EZY2701I Inactivity time is 300
ftpd[33554441]: SD0280 accept_client: socket()
ftpd[33554441]: SD0371 ....hostname..................MVSJ.tcp.raleigh.ibm.com
ftpd[33554441]: SD0376 accept_client: assigned socket 6
ftpd[33554441]: SD0385 accept_client: setsockopt()
ftpd[33554441]: SD0398 accept_client: bind()
ftpd[33554441]: SD0416 accept_client: listen()
ftpd[33554441]: EZY2702I Server-FTP: Initialization completed at 13:59:37 on 03/05/98.
ftpd[33554441]: EZYFT41I Server-FTP: process id 33554441, server job name FTPD1
ftpd[33554441]: SD0469 accept_client: prepare to accept another client
14 ftpd[33554441]: SD0488 accept_client: calling selectex for socket 6
ftpd(33554441): SD0406 accept_client: accept()
ftpd(33554441): SD0424 accept_client: accepted client on socket 7
15 ftpd(33554441): SD0475 accept_client: new session for 9.67.43.72 port 2056
ftpd(33554441): SD0363 accept_client: prepare to accept another client
16 ftpd(33554441): SD0382 accept_client: calling selectex for socket 6
17 ftpd(83886096): SD0794 spawn_ftps: my pid is 83886096 and my parent's is 33554441
ftpd(83886096): SD0733 setup_new_pgm: issuing execv
ftps(83886096): RX0708 undo charvars: msgcat is; 154448120
ftps(83886096): RX0715 undo charvars: replycat is; 154452688
ftps(83886096): SR1178 setup_client_sn: entering setup_client_sn with socket 7
```

Figure 24. Example of a Trace of an OS/390 UNIX FTP Server (Part 3 of 4)

```
   ftps(83886096): RX0304 main: ftp server processing entered for socket 7
   ftps(83886096): RX0311 main: no LANG for child
   ftps(83886096): RX0315 main: child's codeset is: IBM-1047
   ftps(83886096): RX0318 main: child's locale is: C
18  ftps(83886096): RX0356 main: ret code from fndmembr() for FTCHKCMD is: 0
   ftps(83886096): RX0362 main: chkcmdexit successfully loaded
   ftps(83886096): RX0372 main: ret code from fndmembr() for FTCHKPWD is: 0
   ftps(83886096): RX0378 main: chkpwdexit successfully loaded
   ftps(83886096): RX0388 main: ret code from fndmembr() for FTCHKJES is: 0
   ftps(83886096): RX0394 main: chkjesexit successfully loaded
   ftps(83886096): RX0420 main: SMF EXIT not specified in configuration parameters,
       so did not look for FTPSMFEX.
   ftps(83886096): RX0527 init_thread_site_vars: init_thread_site_vars routine entered.
   ftps(83886096): PR0188 parse_cmd: entering parse_cmd.
   ftps(83886096): SR0612 get_command: select rc is 1
   ftps(83886096): SR0649 get_command: received 13 bytes
19  ftps(83886096): PR0308 Parse_cmd: command line: USER user31

.
.
.
   ftps(83886096): SR0612 get_command: select rc is 1
   ftps(83886096): SR0649 get_command: received 6 bytes
   ftps(83886096): PR0308 Parse_cmd: command line: QUIT
   ftps(83886096): PR0325 parse_cmd: calling user exit FTCHKCMD with:
        rc 0, numparms 3, userid 'USER31', cmd 'QUIT' args '0/'
   ftps(83886096): PR0345 parse_cmd: return from FTCHKCMD with rc: 0.
   ftps(83886096): RM1485 quit: quit routine entered.
   ftps(83886096): SR0538 end_session: Ending session 000D086C
20 ftps(83886096): RX0500 Server thread terminates rc = 99.
   ftpd[33554441]: SD0469 accept_client: prepare to accept another client
21  ftpd[33554441]: DF0193 mvs_command_handler: routine entered.
   ftpd[33554441]: DF0205 mvs_command_handler: console command hex code: 40; parm:
   ftpd[33554441]: DH0234 pgmstpd: SIGTERM signal received
   ftpd[33554441]: EZY2714I FTP server shutdown in progress
   ftpd[33554441]: DH0271 pgmstpd: off to ifaeddrg_byaddr...
   ftpd[33554441]: DH0274 pgmstpd: back from ifaeddrg_byaddr.  rc = 0
   ftpd[33554441]: EZYFT59I FTP shutdown complete.
```

Figure 24. Example of a Trace of an OS/390 UNIX FTP Server (Part 4 of 4)

Trace Output Explanation: Following are short descriptions of the numbered items in the trace.

1 Indicates the message catalog used by the FTP server, as determined by the NLSPATH and LANG environment variables. If the catalog cannot be opened, message EZYF501I displays, and the default messages are used.

2 The version and release of the FTP server

3 The FTP daemon begins its search for an FTP.DATA file. The trace shows what happened at each step in the search.

4 No FTP.DATA file was found, so configuration defaults are used.

5 The FTP daemon begins to search for a translate table data set (TCPXLBIN file) to be used for the control connection. If there had been an FTP.DATA file with a valid CTRLCONN statement, this search would not occur.

6 Conversion to be used for the control connection

7 Begin search for translation tables for the data connection.

8 Conversion to be used for the data connection

9 No parameters for the TCPIP.DATA LOADDBCSTABLES statement means

no DBCS data transfer can be done with this FTP server. If languages were specified on a LOADDBCSTABLES statement, they would be listed following this trace entry.

10 Note that the process ID changes as the FTP daemon forks to the background.

11 The initial translate tables to be used for the control and data connections. (a2e=ascii-to-ebcdic; e2a=ebcdic-to-ascii) These can be changed during a client session by a site command.

12 The daemon looks for the FTCHKIP user exit. Trace will indicate whether or not it is loaded.

13 The configuration values the FTP daemon will run with

14 The daemon waits for command or new client connection.

15 The client IP address and port

16 After forking a new address space, the daemon goes back to wait for a command or new connection.

17 New server address space for this client session

18 Server address space loads remaining user exists if they exist.

19 Entire command line is reflected (after conversion to EBCDIC).

Note: The PASS command line is not reflected.

20 Normal end of client session

21 The daemon has received a STOP command.

Documenting Server Problems

If the problem is not caused by any the common errors described in this section, collect the following documentation before calling the IBM Support Center. Documentation is divided into two categories: essential and helpful but not essential.

- Essential:
  - Precise description of problem, including expected results and actual results
  - OS/390 UNIX FTP server dump (for abends)
  - OS/390 UNIX FTP server traces (refer to “Diagnosing FTP Server Problems with Traces” on page 205 for information on collecting FTP server traces)
- Helpful:
  - FTP client output
  - 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 recreate the problem

FTP Client

This section describes the procedures to follow when diagnosing problems with the FTP client.

Execution Environments

The FTP client can run in any of the following environments:

- Interactive (under the TSO or the OS/390 UNIX shell)
- Batch (under TSO only)
- REXX exec (under TSO or the OS/390 UNIX shell)

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 following command to redirect output from a TSO command line to a data set: **ftp 9.68.100.23 > 'USER27.FTPOUT'**. When run under the OS/390 UNIX shell, both input and output can be redirected.

Note: When redirecting output under OS/390 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.

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**.

Setup

You can use an FTP.DATA data set to customize configuration parameters. For information about the FTP.DATA data set used by the FTP client, refer to *OS/390 IBM Communications Server: IP User's Guide*. Use the FTP client LOCSTAT command to display the name of the FTP.DATA 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 *OS/390 IBM Communications Server: IP Configuration Reference*. The OS/390 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 native MVS data sets and HFS files. For more information, see "Name Considerations for OS/390 UNIX FTP" on page 192.

Common Problems

This section describes some common problems with the FTP client.

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
```

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.

Note: The FTP client always uses the OS/390 UNIX resolver, even when FTP is invoked from TSO.

Use the host IP address instead of the hostname on the FTP command or see "Chapter 15. Diagnosing Dynamic Domain Name Server (DDNS) Problems" on page 295 for information on diagnosing name-server problems.

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 **CTRACE** 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 the *OS/390 IBM Communications Server: IP User's Guide*.

Check if your FTP.DATA file has sequence numbers. If it does, any statement with an optional parameter omitted will pick 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.

Data Transfer Problems

Most of the data transfer problems that apply to a server also apply to a client. (See "Data Transfer Problems" on page 197.) In addition, a problem occurs when an FTP client is invoked under TSO and a TYPE U 2 or UCS2 command is invoked. When this happens, the following message is displayed:

```
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>
```

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 displays, 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 RESOLVER_CONFIG environment variable or /etc/resolv.conf takes precedence over DD:SYSTCPD or *jobname*.TCPIP.DATA.

DB2 Query Support

This section describes how to use the FTP client DB2 query support and how to diagnose SQL problems.

How to use FTP Client SQL Support

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

The following two tables shows some symptoms and possible causes of SQL problems. Table 18 shows problems that generate a reply beginning with 55x.

Table 18. SQL Problems Generating 55x Replies

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 will fail.
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.

Table 18. SQL Problems Generating 55x Replies (continued)

Reply	Output file	Possible causes
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 DBZPLAN statement to specify this plan name. • User does not have execute privilege for the DB2 plan being used by the FTP server.
EZA2740E: SQL query not available. Can't 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 *OS/390 IBM Communications Server: IP and SNA Codes*.		

Table 19 shows other SQL problems.

Table 19. Other SQL Problems

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 and 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 will indicate the amount of SQL activity through FTP and the approximate time when each query was processed.

Diagnosing FTP Client Problems with Tracing

You can activate tracing on startup with the -d or CTRACE command line options. Alternatively, you can activate tracing by toggling tracing on or off during an FTP session with the DEBUG command.

The DEBUG (or DEBUG 1) and DEBUG 2 commands activate two levels of tracing. DEBUG 2 includes trace entries generated by level 1 and additionally dumps internal storage. It can generate large amounts of data, so it is recommended that you redirect output to a file when using DEBUG 2.

When running FTP interactively or from a REXX exec, all tracing goes to the terminal unless output has been redirected. When running FTP from a TSO batch job, all tracing goes to SYSOUT.

If you experience problems using DD names to refer to files in FTP transfers:

1. 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 proper, such as for a file that already exists the disposition should not be NEW.
2. 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 will be sent to the remote host for processing as-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 not valid file name.
3. To find attempts to access files by DDNAME, look for DD: in FTP trace output.

```
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
```

Note: By using DDNAME support, the user is assuming responsibility for correctly allocating and deallocating the DDNAMEs being used!

Documenting FTP Client Problems

If the problem is not caused by any the common errors described in this section, collect the following documentation before calling the IBM Software Support Center. Documentation is divided into two categories: essential and helpful but not essential.

- Essential:
  - Precise problem description, including client console, expected results, and actual results
- Helpful:
  - Client trace at the DEBUG 1 level. *This information can be extremely helpful.* (You do not need to generate DEBUG 2 level tracing unless it is requested by IBM service.)
  - Output from the client LOCSTAT command
  - FTP.DATA data set
  - TCPIP.DATA data set
  - If appropriate, sample data to recreate the problem

Chapter 9. Diagnosing OS/390 UNIX Telnet Problems

This chapter provides diagnostic information for OS/390 UNIX Telnet.

Common Problems

The following list describes common problems that you may encounter during execution of the Telnet daemon.

- 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 (check for /etc/syslog.pid).
  - If syslogd is active, ensure that the file where the output is intended to be outputted is currently allocated. Syslogd will not create the file; it expects it to exist. OS/390 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 *OS/390 IBM 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` and/or `-D all` as the OS/390 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 where you would want it to and correction is allowed, but the shell also treats these characters as part of the original command.
- The keyboard appears locked and the user can not 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 309). UNIX-type 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 4. Verify that the exception conditions identified are processed by the OS/390 UNIX Telnet server. (Check EZYTE67I messages for more information; refer to Figure 26 on page 221.)
- 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 may result in either the user system having BPX.DAEMON authority set up in their environment, and the proper authorities have not been issued to the user, or the user does not have super user authority, which may be required to issue some of these commands.

Debug Traces

Table 20 on page 220 describes options that relate to user-controlled trace information.

Table 20. 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	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 of the options/report/ptydata/netdata 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 will have in your syslogd file, the contents in both hexadecimal and ASCII, the data being sent over the sockets or between the ttys. 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 may apply, try the -D all option.

Each set of hexadecimal data will be preceded by a three-letter tag. This tag will represent the direction the data is flowing from. Figure 25 is a pictorial representation of this flow.

- Int—client to parent
- Ont—parent to client
- Ipt—child to parent
- Opt—parent to child

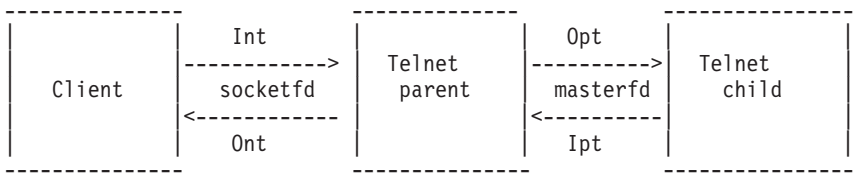


Figure 25. Trace between the Telnet Client, Parent and Child

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 26 on page 221 gives an example of the trace generated from **-t -D all**. This was generated from an AIX Telnet client. A trace explanation follows the figure.

```
1  EZYTE29I Starting new telnet session.  catfd = 134459616
   EZYTE04I catgets  EDC5000I No error occurred. rsn = 00000000
   EZYTO05I Initial EBCDIC codepage = IBM-1047, ascii codepage = ISO8859-1
2  EZYTE05I Trace 1 Debug 1d keepalive 1 kludgelinemode 2
   hostinfo 1 Registered host 0 linemode 1 multi_proc 1
3  EZYTE11I doit: host_name  rperot.raleigh.ibm.com
4  EZYTE11I doit: IP address 9.37.34.249
   EZYTE11I doit: PORT       1028
   EZYTE11I doit: host       MVS3
5  EZYTS04I STATE:send_do: send DO   TERMINAL TYPE¬(
   EZYTS04I STATE:send_do: send DO   TSPEED¬(
   EZYTS04I STATE:send_do: send DO   XDISPLOC¬(
   EZYTS04I STATE:send_do: send DO   NEW-ENVIRON¬(
   EZYTS04I STATE:send_do: send DO   OLD-ENVIRON¬(
   EZYTS04I STATE:send_do: send DO   BINARY¬(
   EZYTS09I STATE:send_will: send WILL   SUPPRESS GO AHEAD¬(
   EZYTS09I STATE:send_will: send WILL   ECHO¬(
6  EZYTU14I UTILITY: netwrite 24  chars.
7  EZYTU21I Ont: fffd18fffd20fffd23fffd27fffd24fffd00fffb .....................
   EZYTU21I Ont: 03fffb01 ....
8  EZYTU03I UTILITY:ttloop read 3 chars.
9  EZYTU47I Int: fffb18 ...
10 EZYTS05I STATE:willoption: receive WILL   TERMINAL TYPE¬(
   EZYTU03I UTILITY:ttloop read 21 chars.
   EZYTU47I Int: fffc20fffc23fffc27fffc24fffc00fffd03fffd .....................
   EZYTU47I Int: 01 .
11 EZYTS08I STATE:wontoption: receive WON'T   TSPEED¬(
   EZYTS08I STATE:wontoption: receive WON'T   XDISPLOC¬(
   EZYTS08I STATE:wontoption: receive WON'T   NEW-ENVIRON¬(
   EZYTS08I STATE:wontoption: receive WON'T   OLD-ENVIRON¬(
   EZYTS08I STATE:wontoption: receive WON'T   BINARY¬(
12 EZYTS10I STATE:dooption: receive DO   SUPPRESS GO AHEAD¬(
   EZYTS10I STATE:dooption: receive DO   ECHO¬(
13 EZYTU17I UTILITY: send suboption
   TERMINAL-TYPE
   SEND
   EZYTU14I UTILITY: netwrite 6  chars.
   EZYTU21I Ont: fffa1801fff0 .....0
   EZYTU03I UTILITY:ttloop read 4 chars.
   EZYTU47I Int: fffa1800 ....
   EZYTU03I UTILITY:ttloop read 7 chars.
   EZYTU47I Int: 7674323230fff0 ......0
```

Figure 26. OS/390 UNIX Telnet Trace Using -t -D all (Part 1 of 3)

```
EZYTS17I Defer suboption negotiation
EZYTU14I UTILITY: netwrite 0  chars.
EZYTU17I UTILITY: receive suboption
TERMINAL-TYPE
IS vt220
14 EZYTE10I terminaltypeok: call tgetent (buf, vt220)
¬(
EZYTE48I Ont: c5e9 EZ
EZYTE59I read_pw: Character ignored a
15 EZYTO04I lusername  = user79
¬(
EZYTE48I Ont: c5e9 EZ
EZYTE59I read_pw: Character ignored a
EZYTE22I herald()
16   EZYTE26E herald: stat error  EDC5129I No such file or directory. rsn = 057C006C
EZYTE16I uid = 215, gid = 5
EZYTY03I GOTPTY: ioctl TIOCSWINSIZ   EDC5000I No error occurred. rsn = 00000000
EZYTY05I GETPTY: slave fd = 10 , masterfd = 7
17 EZYTS15I STATE:dooption:deferred receive DO   ECHO¬(
EZYTO09I options(1) = 3 .
EZYTS15I STATE:dooption:deferred receive DO   SUPPRESS GO AHEAD¬(
EZYTO09I options(3) = 3 .
17 EZYTS16I STATE:willoption:deferred receive WILL   TERMINAL TYPE¬(
EZYTO09I options(24) = 12 .
EZYTS04I STATE:send_do: send DO   LINEMODE¬(
EZYTS04I STATE:send_do: send DO   NAWS¬(
EZYTS09I STATE:send_will: send WILL   STATUS¬(
EZYTS04I STATE:send_do: send DO   LFLOW¬(
EZYTU14I UTILITY: netwrite 12  chars.
EZYTU21I Ont: fffd22fffd1ffffb05fffd21 ............
EZYTU03I UTILITY:ttloop read 6 chars.
EZYTU47I Int: fffc22fffb1f ......
EZYTS08I STATE:wontoption: receive WON'T   LINEMODE¬(
EZYTS05I STATE:willoption: receive WILL   NAWS¬(
EZYTS04I STATE:send_do: send DO   TIMING MARK¬(
EZYTS04I STATE:send_do: send DO   BINARY¬(
EZYTU14I UTILITY: netwrite 6  chars.
EZYTU21I Ont: fffd06fffd00 ......
EZYTE66I PROTOCOL:lmodetype=2, linemode=0, uselinemode=0
18 EZYTY08I argv_fsum(0) =  fomtlinp
EZYTY08I argv_fsum(1) =  *4OurhrEa)R0,H/h
EZYTY08I argv_fsum(2) =
```

Figure 26. OS/390 UNIX Telnet Trace Using -t -D all (Part 2 of 3)

```
EZYTY08I argv_fsum(3) =  0
EZYTY08I argv_fsum(4) =  7
EZYTY08I argv_fsum(5) =  10
EZYTY08I argv_fsum(6) =  0
EZYTY08I argv_fsum(7) =  0
EZYTY08I argv_fsum(8) =  6
EZYTY08I argv_fsum(9) =  80
EZYTY08I argv_fsum(10) =
EZYTY08I argv_fsum(11) = vt220
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 131080
19 EZYTE67I S(nfd):socketfd..ibits=00000001 obits=00000000 ebits=00000000
       S(nfd) pty..ibits=00000000 obits=00000000 ebits=00000080
20 EZYTE68I Ept: #bytes = 4 pkcontrol(cntl) 1003
   EZYTE69I PROTOCOL: cntl = 1003
   EZYTE65I PROTOCOL: send IAC Data Mark.  DMARK¬(
```

Figure 26. OS/390 UNIX Telnet Trace Using -t -D all (Part 3 of 3)

Following are short descriptions of the numbered items in the trace:

1 EZYTE29I indicates the start of a new OS/390 UNIX Telnet client session.

2 EZYTE05I indicates what options were specified in /etc/inetd.conf for OS/390 UNIX Telnet.

3 EZYTE11I indicates the resolved host name (from the client).

4 EZYTE11I shows the IP address of the OS/390 UNIX Telnet client.

5 EZYTS04I indicates the terminal negotiation options sent to the client by the OS/390 UNIX Telnet server.

6 EZYTU14I traces netwrites (writes to the client terminal).

7 EZYTU21I traces data from parent to client; that is, OS/390 UNIX Telnet to the client terminal.

8 EZYTU03I indicates the number of bytes read from the client by OS/390 UNIX Telnet.

9 EZYTU47I traces data from the client to the parent (OS/390 UNIX Telnet server).

10 EZYTS05I shows the terminal option negotiation the client has sent/received.

11 EZYTS08I shows the terminal option negotiation the client has sent/received.

12 EZYTS10I shows the terminal option negotiation the client has sent/received.

13 EZYTU17I traces OS/390 UNIX Telnet sending terminal negotiation sub-options to the client.

14 EZYTE10I traces the call to tgetent(), which determines client terminal type.

15 EZYTO04I shows the user name with which the telnet client logged in.

16 EZYTE26E indicates no /etc/banner file was found.

17 EZYTS15I and EZYTS16I show that a state change was processed due to options/responses received from the client.

18 EZYTY08I traces the parameters passed to the spawned/forked child address space where the OMVS shell runs.

19 EZYTE67I traces the socket sets to show whether input/ibits, output/obits, or exception/ebits data has been received.

20 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 may have terminated the session abnormally. When this occurs the utmp entry for that user will remain 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 will not be available for reuse until someone under superuser erases the /etc/utmpx file.

Note: If you erase the file while someone is logged on, the next logoff will report not finding the utmpx entry for the user. This can be seen with a waitpid failure during that user cleanup.

Chapter 10. Diagnosing Telnet Problems

This chapter describes how to diagnosis Telnet problems.

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 want to use 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 240.

Telnet Server Definitions

Telnet must be defined correctly to both VTAM and TCP/IP. A VTAM APPL definition statement is needed for each Telnet LU that will be used or model application naming can be used with VTAM V4R3 or higher. A corresponding LU must be specified in the BEGINVTAM section of the PROFILE.TCPIP data set. Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for detailed information about these definitions.

Note: All default 3270 LOGMODE entries from the table of Telnet device name parameters in *OS/390 IBM 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. Also, the SESSLIM=YES parameter should be coded for each VTAM APPL definition statement (SNA or non-SNA) to ensure correct Telnet processing with applications using VTAM CLSDST/PASS macros.

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.

If a problem is re-creatable, you can use the DEBUG DETAIL statement in TELNETPARMS, refer to *OS/390 IBM Communications Server: IP Configuration Reference* for details.

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 will be 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 to ensure that a useful dump is obtained in the event of an abend.

Analysis

Refer to *OS/390 MVS Diagnosis: Procedures* or see Chapter 3. Diagnosing Abends, Loops, and Hangs, for debugging dumps produced during TCP/IP processing.

Logon Problems (Server)

Telnet login 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 TCP/IP).

Documentation

The following documentation should be available for initial diagnosis of Telnet login problems:

- TCP/IP console log
- PROFILE.TCPIP data set
- VTAM APPL definitions for Telnet LUs

More documentation that might be needed is discussed later in the analysis section.

Analysis

Table 21 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 21. Telnet Login Problems

Login Problem	Analysis Steps
No LUs available	1, 2, 6, 10
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, 11

Following are the diagnosis steps referred to in Table 21.

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.TCPIP data set.
2. Is the VTAM node containing the Telnet LU definitions active?
3. Is there a DEFAULTAPPL coded in the PROFILE.TCPIP data set?

4. Is the host application (or DEFAULTAPPL) active?
5. Is there an ALLOWAPPL statement coded that will include the requested application?
6. Have comment delimiters been added or removed as needed in the BEGINVTAM section of the PROFILE.TCPIP data set?
7. Have correct LOGMODEs (or required overrides for SNA) been coded in the PROFILE.TCPIP 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.TCPIP 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?

If the problem still occurs after following the preceding procedure and making any needed changes, obtain the following documentation:
- TCP/IP packet trace and data trace
- VTAM buffer trace of the Telnet LU
- VTAM DISPLAY of the host application

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:
- Component Trace
- TCP/IP Services Telnet display output
- VTAM VIT trace (API, PIU, MSG, PSS options)
- VTAM DISPLAY of Telnet LU
- Dump of TCP/IP address space

For information about obtaining VTAM traces, refer to *OS/390 IBM Communications Server: SNA Operation* or to *OS/390 IBM Communications Server: SNA Diagnosis V2 FFST Dumps and the VIT* for your release. Instructions on obtaining a dump can be found in *OS/390 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.

Documentation

To determine the cause of a Telnet session hang, the following documentation will usually be required:
- TCP/IP packet trace and data trace
- VTAM buffer trace of the Telnet LU

- Information about what was seen at the client screen

Analysis

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 and data trace will show the data as it arrives from the client and is sent to the platform code to be forwarded to the host application. Some processing steps during this time are also included in the trace.

The VTAM buffer trace will show 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 will provide an indication of where the problem is occurring.

The following list suggests information to check in the traces. Refer to *OS/390 IBM Communications Server: SNA Diagnosis V2 FFST Dumps and the VIT* or to *SNA Network Product Formats* for more information about VTAM buffer trace output.

Was the last activity at the client input or output? If input (data from the client), start with step 1. If output, go to Step 9.

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 data trace show data passed to Telnet? 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 data trace output? If not, the problem is in the Telnet server. Otherwise, continue with Step 8.
8. Does the packet trace show the data sent to the client? If not, the error is in TCP/IP platform. Otherwise, continue with Step 9.
9. Check the data in the packet trace output to see if unlock keyboard is set on in the data stream. If unlock is set in the output data, the problem is in the emulator or client code. Otherwise, continue with Step 10.
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. You should contact your host application IBM Software Support Center.

If the preceding problem determination shows the error to be in the TCP/IP platform or Telnet server code, a dump will be 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.

Documentation

Documentation needed to find the source of the error in an incorrect output problem would be:

- TCP/IP packet trace and data trace
- VTAM buffer trace of the Telnet LU
- Client screen output information

Analysis

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 22 lists the types of incorrect output that might be seen and the steps needed to identify the code in error.

Table 22. Incorrect Output Types for Telnet

Incorrect Output	Analysis Steps
Blank screen	1, 6, 7
Garbled or unreadable characters on the screen	2, 3, 4, 5, 6, 7
Incorrectly formatted screen	6, 7

Refer to Table 22 to find which of the following steps to use in determining the cause of the error.

1. Was the last output data seen in the packet trace displayed at the terminal? If not, problem is in the client or emulator. Contact your IBM Software Support Center for this product. If the last output was seen at the terminal, go to step 9 on page 228 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. Is the data stream sent to the client (packet trace) the same as that sent by Telnet (data trace)? If not, the problem is with the TCP/IP platform code. Otherwise, continue with Step 4 on page 228.
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 228.

   Note: 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. Is the data passed by VTAM the same as received by VTAM (check VTAM USER and VTAM BUFF entries)? If not, VTAM has corrupted the code. 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?

Note: Use display connection to determine the LOGMODE of the session.

7. Does the device type information in the BIND sent by the host application match the device type information in the specified LOGMODE entry?

   Note: This can be checked by comparing the specified LOGMODE entry (refer to *OS/390 IBM Communications Server: SNA Customization*) with the BIND in the buffer trace at logon to the selected application.

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 will result in the client being returned to the panel where the initial TELNET command was entered.

The Telnet server will end a session if there is no activity for the amount of time specified in the INACTIVITY parameter of the TELNETPARMS statement in the PROFILE.TCPIP data set. The server will also send packets to each Telnet connection at the interval specified in the TIMEMARK parameter of the TELNETPARMS statement. If no response to this TIMEMARK packet is received, the session will be ended.

Documentation

The following documentation is needed for initial investigation of problems reported as session outages:

- MVS system console log
- TCP/IP console log
- TCPIP SYSDEBUG data set

Analysis

The preceding output is needed to begin diagnosis of a session outage reported against Telnet. It will also be helpful to know what kind of processing the Telnet user was doing at the time of the interrupted session.

The following steps are suggested for initial investigation of a Telnet session outage:

1. If a time-out due to inactivity or termination due to TIMEMARK processing is suspected, check the values set in the PROFILE.TCPIP data set.
2. Check the documentation listed in “Documentation” for indications of an error.
   - If the MVS system console indicates a VTAM error, continue diagnosis with your VTAM programmer.
   - If the TCP/IP console shows a Telnet or TCP/IP error, check *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)* and follow the directions for system programmer response for the message.
   - Information in the TCPIP SYSDEBUG data set should contain a detailed description of the error.

   If messages are found that do not lead to an accurate diagnosis and resolution of the error, contact your IBM Software Support Center for host TCP/IP.
3. If only one Telnet user session was affected, continue with step 4 on page 231. Otherwise, go to step 6 on page 231.

4. If the problem can be recreated by performing the same operation or processing, the following traces should be helpful in further diagnosis of the error:
   - VTAM buffer trace output
   - TCP/IP packet trace and data trace
   - Component Trace output
   - Please contact your IBM Software Support Center for TCP/IP for information about options needed before running these traces.

   A VTAM internal trace (VIT) might also be needed.
5. If all Telnet user sessions were interrupted, check the MVS system console and LOGREC for abends. This type of outage is usually seen when an abend in VTAM or the TCP/IP VTAM interface code occurs. See “Abends (Server)” on page 226 for information about obtaining and diagnosing a dump of the failure.
6. If there are no messages or abends and all Telnet user sessions have been disconnected, the traces listed in Step 4 will be needed during a recurrence of the failure.

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 once SSL handshaking is completed. If problem determination requires seeing Telnet handshake or user data, you also need to run Component Trace to see the decrypted data field. When starting Component Trace, specify **options=**(*TELNET*) and use IPCS to format the Component Trace. For more information on Component Trace, see “Chapter 5. TCP/IP Services Traces and IPCS Support” on page 47.

The Telnet Component Trace records contain the client IP address in the Connection Identifier (CID) field. Use this field to locate records related to the client in question. Once 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 has completed and subsequent data on this connection will be 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 IP address of the client.

USER The LUname associated with the client, once it has been assigned. Prior to LUname assignment, this field may 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 connecting from 9.37.82.196 with the LUName TCPM1011, code the following IPCS command:

```
CTRACE COMP(SYSTCPIP) SUB((TCPIP)) FULL JOBLIST (TCPM1011)
     OPTIONS((TELNET,CID(X'092552C4')))
```

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 *OS/390 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 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. Refer to the “Chapter 3. Diagnosing Abends, Loops, and Hangs” on page 21, for more information.

Analysis

Refer to *OS/390 MVS Diagnosis: Procedures* or see “Chapter 3. Diagnosing Abends, Loops, and Hangs” on page 21 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 data trace 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.

Analysis

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 27 on page 236. Refer to *OS/390 IBM Communications Server: SNA Diagnosis V2 FFST Dumps and the VIT* or to *SNA Network Product Formats* for more information about VTAM buffer trace output.

Following are suggested 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 225. Otherwise, continue with step 2.
2. Was the last activity at the terminal input or output? If input, go to step 5. If output, continue with step 3.
3. Check the data in the VTAM buffer trace to see if unlock keyboard is set on in the data stream. If unlock is set on in the data stream, the problem is in the emulator, control unit, or terminal device. If not, check the Telnet client trace to ensure the output data stream matches what is seen in the buffer trace. If the data streams match, the remote host application has not unlocked the keyboard. Contact your IBM Software Support Center for 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. Refer to “Session Hangs (Server)” on page 227 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.

Note: Information about starting and examining traces is discussed in “Starting Telnet Client Traces” on page 235.

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

Analysis

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 an 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.

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 27 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
   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 27. 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 27. 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 27. 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 27. 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 (via the Telnet server) that is being sent to the terminal.

Telnet Commands and Options

Table 23 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 23. 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 24 lists the options available for Telnet commands from RFC 1060. For more information about Telnet protocols, refer to RFC 1060 and RFC 1011.

Table 24. 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

Table 24. Telnet Command Options from RFC 1060 (continued)

Option	Option (Hex)	Name
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
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 11. Diagnosing Simple Mail Transfer Protocol (SMTP) Problems

The Simple Mail Transfer Protocol (SMTP) protocol 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.

Note: For information about diagnosing problems with the other CS for OS/390 mail application, OS/390 UNIX sendmail, see “Chapter 12. Diagnosing OS/390 UNIX sendmail and Popper Problems” on page 253.

Sender SMTP

The sender SMTP performs the following functions:

- Receives notes from the SMTPNOTE CLIST via a TSO TRANSMIT command
- Resolves the host name of recipients via 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:

- Accepts mail from remote TCP/IP hosts
- Delivers mail to the local user via 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 28 shows the SMTP environment.

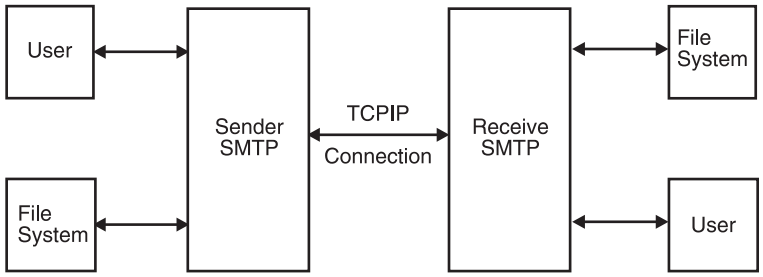


Figure 28. SMTP Environment

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)
```

Notes:

1. The NJE node name, *nodename*, must be the same as the *hostname* and the *smtpnode* in the SMTPNOTE CLIST.
2. SMTP can handle only one NJE node name.

Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for more information about configuring SMTP and about the Program Directory.

Diagnosing SMTP Problems

Problems with SMTP 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 will be needed unless the symptoms already match a known problem.

Documentation

The following documentation is needed for abends:

- Dump

  Note: 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 *OS/390 MVS Diagnosis: Procedures* or see “Chapter 3. Diagnosing Abends, Loops, and Hangs” on page 21, 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 the *OS/390 IBM Communications Server: IP User's Guide* .

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

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.

Analysis

Use the following procedure to analyze the problem:

1. If the problem is that mail was not forwarded to a local user:
   a. Was SMTPNOTE customized for your installation?
   b. Is the local user one that is coded as a restricted user in the SMTP.CONFIG data set?
   c. 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.
2. If the problem is that a mail note was not forwarded to an NJE host:
   a. Follow the preceding steps for mail that was not forwarded to a local user.
   b. Is SMTP configured as an NJE gateway?
   c. Was SMTPNJE successfully run to create the NJE host table dataset.
   d. Check if the NJE mode is in the NJE host table data set.

   Note: Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for information about SMTP configuration.
3. If the problem is that mail was not forwarded to a remote TCP/IP host:
   a. 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 247.

      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 0.
   b. 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 “RESOLVER Trace” on page 250.
   c. Is the remote TCP/IP/SMTP server running?

      Use the PING command to see if the remote TCP/IP is running.

      Note: 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, please 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.

Note: Coding the NOSPIE runtime 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 21 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.
- ADDRBLOK data set.

  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.
- SMTPNOTE data set.

Analysis

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?

Use the following procedure to analyze the problem:

1. If the problem is that the received mail item has incorrect output:
   a. Is the correct translation table being used or could it have been customized to cause the error?

      Correct the translation error.

b. 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.

c. Analyze the output from SMTP DEBUG for obvious errors.

   Note: The body of the note (mail item) is not shown in this output.

2. If the problem is that the sent mail item has incorrect output:
   a. Is the correct translation table being used, or could it have been customized to cause the error?

      Correct the translation error.

   b. 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.

   c. Analyze the output from SMTP DEBUG for obvious errors.

      Note: 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 Re-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

To force the SMTP server to reresolve the addresses, modify the ADDRBLOK data set for the piece of mail. For each recipient record (records 3 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 reresolve 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.

Note: 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 25 shows the format of Record 1 (the master control record) of an SMTP ADDRBLOK data set.

Table 25. 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”.		
57–64	Tag user ID	8
65–72	Tag node ID	8
73–76	Spool ID on the current system	4
77–80	Spool ID of the file source	4

Table 26 shows the format of Record 2 (for an unresolved From record) of an SMTP ADDRBLOK data set.

Table 26. 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

Table 26. Format of Record 2 (For an Unresolved From Record) of an SMTP ADDRBLOK Data Set (continued)

Characters	Description	Length (in Characters)
(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 27 shows the format of Record 2 (for a resolved From record) of an SMTP ADDRBLOK data set.

Table 27. 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 28 shows the format of Records 3–*n* of an SMTP ADDRBLOK data set.

Table 28. 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

Table 28. Format of Record 3 (For an Unresolved From Record) of an SMTP ADDRBLOK Data Set (continued)

Characters	Description	Length (in Characters)
(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.		

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 the job log output from the started SMTP procedure.

Figure 29 on page 251 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 29. 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 12. Diagnosing OS/390 UNIX sendmail and Popper Problems

This chapter describes how to diagnose problems with OS/390 UNIX sendmail, an electronic mail-transport agent and server, and with OS/390 UNIX popper, a mail-delivery agent.

Diagnostic Aids for sendmail

The following sections describe various tools and techniques available for diagnosing problems with OS/390 UNIX sendmail. For a comprehensive discussion of sendmail, refer to the industry-accepted publication *sendmail* by O'Reilly & Associates, Inc. (ISBN 1-56592-222-0). 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 OS/390 UNIX sendmail at the *http://www.sendmail.org* web site.

Note: For information about diagnosing problems with the other CS for OS/390 mail application, Simple Mail Transfer Protocol (SMTP), see "Chapter 11. Diagnosing Simple Mail Transfer Protocol (SMTP) Problems" on page 243.

Debugging Switches

The following table is 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 29. Debugging Switches by Category

Category	Bat Book Reference	Useful for Mail Problems	Description
-d0.1	37.5.1	X	Print version information
-d0.4	37.5.2	X	Our name and aliases
-d0.10	37.5.3		Operating System defines
-d0.15	37.5.4	X	Dump delivery agents
-d0.20	37.5.5	X	Print network address of each interface
-d0.22	37.5.6		Show uname() failure
-d0.40	37.5.7		Show scanning of interfaces
-d0.44	37.5.8		Print addresses of strings
-d0.90	37.5.9	(obsolete)	Print first 10 rule sets
-d1.1	37.5.10		Show sender information
-d1.5	37.5.11		Dump the sender address
-d2.1	37.5.12		End with finis()
-d2.9	37.5.13		Show file descriptors with *dumpfd()*
-d3.1	37.5.14		Print the load average
-d3.5	37.5.15		Print load average
-d3.15	37.5.16		Print three load averages

Table 29. Debugging Switches by Category (continued)

Category	Bat Book Reference	Useful for Mail Problems	Description
-d3.20	37.5.17		Show offset for load average
-d3.30	37.5.18		Show result of decision to queue
-d4.80	37.5.19	X	Trace enoughspace()
-d5.4	37.5.20		Tick for queued events
-d5.5	37.5.21		Events set and cleared
-d5.6	37.5.22		Show events triggered
-d6.1	37.5.23	X	Show failed mail
-d6.5	37.5.24		The current error state
-d6.20	37.5.25		Show sender of return to sender
-d7.1	37.5.26		The Queue filename
-d7.2	37.5.27		Show assigned queue filename
-d7.9	37.5.28		Dump file descriptor for the qf file
-d7.20	37.5.29		Show queue names being tried
-d8.1	37.5.30	X	Failure of MX search (low level)
-d8.2	37.5.31	X	Call to getcanonname(3)
-d8.3	37.5.32	X	Trace dropped local hostnames
-d8.5	37.5.33	X	Hostname being tried in getcanonname(3)
-d8.7	37.5.34	X	Yes/no response to -d8.5
-d8.8	37.5.35	X	MX lookup gets wrong type
-d8.20	37.5.36		Inconsistency in returned information
-d9.1	37.5.37		Canonify hostname and RFC1413 queries
-d9.3	37.5.38		Show raw RFC1413 reply
-d9.10	37.5.39		Show RFC1413 query being sent
-d10.1	37.5.40		Show recipient delivery
-d10.2	37.5.41		Dump controlling user's address
-d10.5	37.5.42		Show don't send to MeToo address
-d10.100	37.5.43		Predelivery file descriptor dump
-d11.1	37.5.44	X	Trace delivery
-d11.2	37.5.45	X	Show the uid/gid running as during delivery
-d11.20	37.5.46		Show tried D= directories
-d12.1	37.5.47	X	Show mapping of relative host
-d13.1	37.5.48	X	Show delivery
-d13.5	37.5.49		Show addresses that we should not send to
-d13.6	n/a		Trace envelope stripping, dropping, and moving
-d13.10	37.5.50		Trace sendenvelope()
-d13.20	37.5.51		Show final mode
-d13.21	n/a		Show final send queue
-d13.25	n/a		Watch owner deliveries
-d13.29	37.5.52		Show autoqueueing

Table 29. Debugging Switches by Category (continued)

Category	Bat Book Reference	Useful for Mail Problems	Description
-d13.30	37.5.53		Show envelopes being split
-d14.2	37.5.54		Show header field commas
-d15.1	37.5.55		Show network get request activity
-d15.2	37.5.56		Incoming connections
-d15.101	37.5.57		Kernel TCP debugging
-d16.1	37.5.58		Outgoing connections
-d16.101	37.5.59		Kernel TCP debugging
-d17.1	37.5.60		List MX hosts
-d17.9	37.5.61		Show randomizing MX records
-d18.1	37.5.62		Show SMTP replies
-d18.2	37.5.63		Show entry to MAIL From:
-d18.100	37.5.64		Pause on SMTP read error
-d19.1	37.5.65		Show ESMTP MAIL and RCPT parameters
-d20.1	37.5.66	X	Show resolving delivery agent: parseaddr()
-d21.1	37.5.67		Trace rewriting rules
-d21.2	37.5.68	X	Trace $& macros
-d21.3	37.5.69		Show subroutine calls
-d21.4	37.5.70		Result after rewriting by a rule
-d21.10	37.5.71		Announce failure
-d21.12	37.5.72		Announce success and show LHS
-d21.15	37.5.73		Show $digit replacement
-d21.35	37.5.74		Show token by token LHS matching
-d21.36	37.5.75		Trace class matching in the LHS
-d22.1	37.5.76	X	Trace tokenizing an address: prescan()
-d22.11	37.5.77	X	Show address before prescan
-d22.12	37.5.78		Show address after prescan
-d22.36	37.5.79		Show each token
-d22.101	37.5.80		Trace low-level state machine
-d24.4	37.5.81		Trace address allocation
-d24.5	37.5.82		Trace assembly of tokens
-d24.6	37.5.83		Show result of buildaddr()
-d25.1	37.5.84	X	Trace ″sendtolist″
-d26.1	37.5.85		Trace recipient queueing
-d26.8	37.5.86		Trace self-destructing addresses
-d26.10	37.5.87		Show full send queue in testselfdestruct
-d27.1	37.5.88	X	Trace aliasing
-d27.2	37.5.89	X	Include file, self-reference, error on home
-d27.3	37.5.90	X	Forwarding path and alias wait
-d27.4	37.5.91	X	Print not safe

Table 29. Debugging Switches by Category (continued)

Category	Bat Book Reference	Useful for Mail Problems	Description
-d27.5	37.5.92		Trace aliasing with printaddr()
-d27.8	37.5.93		Show setting up an alias map
-d27.9	37.5.94	X	Show uid/gid changes with :include: reads
-d27.14	37.5.95		Show controlling user that caused change in identity
-d27.20	37.5.96		Show how alias will be looked up in a map
-d28.1	37.5.97	X	Trace user database transactions
-d28.2	37.5.98		Show no match
-d28.4	37.5.99		Show result of lookup
-d28.8	37.5.100		Try hes_getmailhost()
-d28.16	37.5.101		MX records for forward host
-d28.20	37.5.102		Show udp lookup
-d28.80	37.5.103		Preview lookups
-d29.1	37.5.104		Special rewrite of local recipient
-d29.4	37.5.105	X	Trace fuzzy matching
-d29.5	37.5.106		Preview rule set 5
-d29.7	37.5.107		Show overaliasing fuzzy fallback
-d30.1	37.5.108		Trace processing of header
-d30.2	37.5.109		Eat from
-d30.3	37.5.110		Show a to-less header being added
-d30.35	37.5.111		Trace collect states
-d30.94	37.5.112		Trace collect states
-d31.2	37.5.113	X	Trace processing of headers
-d31.6	37.5.114		Is header known?
-d32.1	37.5.115		Show collected headers
-d32.2	37.5.116		Show ARPA mode with setsender
-d33.1	37.5.117		Watch crackaddr()
-d34.1	37.5.118		Watch header assembly for output
-d34.11	37.5.119	X	Trace header generation and skipping
-d35.9	37.5.120	X	Macro values defined
-d35.14	37.5.121		Macro identification
-d35.24	37.5.122		Macro expansion
-d36.5	37.5.123		Trace processing by stab()
-d36.9	37.5.124		Show hash bucket
-d36.90	37.5.125		Trace function applied to all symbols
-d37.1	37.5.126	X	Trace setting of options
-d37.8	37.5.127	X	Trace adding of words to a class
-d38.2	37.5.128	X	Show map opens and failures
-d38.3	37.5.129		Show passes
-d38.4	37.5.130	X	Show result of map open

Table 29. Debugging Switches by Category (continued)

Category	Bat Book Reference	Useful for Mail Problems	Description
-d38.9	37.5.131		Trace map closings and appends
-d38.10	37.5.132		Trace NIS search for end of aliases
-d38.12	37.5.133		Trace map stores
-d38.19	37.5.134	X	Trace switch map finds
-d38.20	37.5.135	X	Trace map lookups
-d38.44	37.5.136		Show nis_getcanonname() record
-d39.1	37.5.137		Display digit database mapping
-d40.1	37.5.138		Trace processing of the queue
-d40.3	37.5.139		Show envelope flags
-d40.4	37.5.140		Show qf file lines as they are read
-d40.8	37.5.141		Show reasons for failure
-d40.9	37.5.142		Show qf and lock file descriptors
-d40.32	37.5.143		Dump the send queue
-d41.1	37.5.144	X	Trace queue ordering
-d41.2	37.5.145		Cannot open qf
-d41.49	37.5.146		Show excluded (skipped)queue files
-d41.50	37.5.147		Show every file in the queue
-d42.2	37.5.148		Show connection checking
-d42.5	37.5.149		Trace caching and uncaching connections
-d43.1	37.5.150		Trace MIME conversions
-d43.3	37.5.151		See the final MIME boundary name
-d43.5	37.5.152		Watch search for boundaries
-d43.8	37.5.153		Show the calculations
-d43.35	37.5.154		Show boundary lines as emitted
-d43.36	37.5.155		Show content transfer encoding
-d43.40	37.5.156		Show parse of Content-Type: header
-d43.99	37.5.157		Print the leading/following comments
-d43.100	37.5.158		Mark collect() and putheader()
-d44.4	37.5.159		Trace safefile()
-d44.5	37.5.160	X	Trace writable()
-d45.1	37.5.161		Show envelope sender
-d45.3	37.5.162		Show saved domain
-d45.5	37.5.163		Show don't send to sender
-d46.9	37.5.164		Show xf file's descriptors
-d48.2	37.5.165	X	Trace calls to the check_rule sets
-d49.1	37.5.166		Trace checkcompat()
-d50.1	37.5.167		Show envelope being dropped
-d50.2	37.5.168		Show Booleans
-d50.10	37.5.169		Also show the send queue

Table 29. Debugging Switches by Category (continued)

Category	Bat Book Reference	Useful for Mail Problems	Description
-d51.4	37.5.170		Show queue entries being unlocked
-d51.104	37.5.171		Prevent unlink of xf file
-d52.1	37.5.172		Show isconnect from controlling TTY
-d52.100	37.5.173		Prevent disconnect from controlling TTY
-d53.99	37.5.174		Trace xclose()
-d54.1	37.5.175		Show error return and output message
-d54.8	37.5.176		Show message and flags
-d55.60	37.5.177		Show file locking
-d56.1	37.5.178		Persistent host status tracing
-d56.2	37.5.179		More persistent host status tracing
-d56.12	37.5.180		Perform a sanity check
-d56.80	37.5.181		Trace creating the path to the status file
-d56.193	37.5.182		Dump MCI record for the host
-d57.2	37.5.183		Monitor vsnprintf() overflows
-d59.1	37.5.184		XLA from contrib
-d60.1	37.5.185	X	Trace map lookups inside rewrite()
-d61.10	37.5.186		Trace gethostbyname()
-d62.1	37.5.187		Log file descriptors before and after all deliveries
-d62.8	37.5.188		Log file descriptors before each delivery
-d62.10	37.5.189		Log file descriptors after each delivery
-d80.1	37.5.190		Content-Length: header (Sun enhancement)
-d81.1	37.5.191		> option for remote mode (Sun enhancement)
-d91.100	37.5.192		Log caching and uncaching connections
-d99.100	37.5.193	X	Prevent backgrounding the daemon

Additional Diagnostic Aids

In addition to debugging switches, you can use the following OS/390 UNIX sendmail diagnostic aids:

- Look in the syslog.log for information. Following is a sample OS/390 UNIX sendmail syslog.log message:

```
Mar 4 16:17:15 sendmail Y973078550 : EZZ7514I: sendmail starting
 .
 .
Mar 4 16:17:47 sendmail Y486539289 : starting daemon (8.8.7):SMTP
```

  For descriptions of sendmail messages, see *OS/390 IBM Communications Server: IP and SNA Codes*.
- Use the -v (verbose) command-line switch to print a complete description of all the steps required to deliver a mail message. For details, see Section 4.2 in *sendmail, 2nd 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, see Section 26.4 in *sendmail, 2nd Edition*.
- Check the qf file for queueing concerns. OS/390 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 30. References in *sendmail, 2nd Edition* are listed in the second column.

Table 30. qf File Code Letters

Code	sendmail book Reference	Meaning	How Many
B	23.9.1	Body type	At most one
C	23.9.2	Controlling user	At most one per R line
D	23.9.3	Obsolete	Obsolete
E	23.9.4	Errors to	Many
F	23.9.5	Flag bits	Many
H	23.9.6	Header definition	Many
I	23.9.7	df file's inode number	Exactly one
K	23.9.8	Time last processed	Exactly one
M	23.9.9	Message (why queued)	At most one
N	23.9.10	Number times tried	At most one
P	23.9.11	Priority (current)	At most one
Q	23.9.12	Original recipient	At most one
R	23.9.13	Recipient address	Many
S	23.9.14	Sender address	Exactly one
T	23.9.15	Time created	Exactly one
V	23.9.16	Version	Exactly one
Z	23.9.17	DSN envelope ID	At most one
$	23.9.18	Restore macro value	At most one
.	23.9.19	End of qf file	At most one

Diagnostic Aids for Popper

Diagnostic aids are found in the SYSLOGD log information. Following is a sample OS/390 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 will refuse 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 RFC1725. Following are a few of the commands used to verify that popper is listening on port 110:

user *name 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 13. Diagnosing SNALINK LU0 Problems

The TCP/IP for MVS 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

Note: SNALINK LU6.2 diagnosis is discussed in "Chapter 14. Diagnosing SNALINK LU6.2 Problems" on page 273.

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 *OS/390 IBM 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 will indicate that VTAM is abending or a dump is being taken for the SNALINK LU0 application.

Note: In the case of a VTAM error caused by SNALINK LU0, refer to *OS/390 IBM Communications Server: SNA Messages* and *OS/390 IBM Communications Server: IP and SNA Codes* for initial problem determination.

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' will occur.

Note: 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 21.

An abend or unexpected termination of the SNALINK LU0 application will 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) will not be 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 APPL STATUS output
- SNALINK LU0 DEBUG trace output
- VTAM buffer trace of the SNALINK LU0 applications
- VTAM internal trace

Note: For information on VTAM traces, refer to *OS/390 IBM Communications Server: SNA Diagnosis V2 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.

Analysis

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. Use the following steps to help 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 will need to determine which TCP/IP application or component is failing. If there is no data traffic, continue with step 2.

   Note: SNALINK LU0 traffic can also be checked by doing multiple VTAM displays of the SNALINK LU0 application. The SEND and RECEIVE data count should increase for an active session.
2. Issue NETSTAT DEVLINKS to determine the status of the SNALINK LU0 TCP/IP. 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 “connected” or “sending,” continue with step 3.
3. At this point, we need to 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.

   Note: Information on starting and examining the trace data is discussed in “Starting SNALINK LU0 DEBUG Trace” on page 270.

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 an 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 console 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 *OS/390 IBM 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

Note: For information on VTAM traces, refer to *OS/390 IBM Communications Server: SNA Diagnosis V2 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 IP packet trace and the SNALINK LU0 DEBUG trace are useful in diagnosing SNALINK LU0 problems.

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. The following command, when passed to the SNALINK LU0 interface, starts the SNALINK LU0 address space packet trace function:

```
MODIFY addr_sp_name,PKTTRACE,ON,LINKNAME=link_name
```

where addr_sp_name is the name of the local SNALINK LU0 address space and link_name is the VTAM LU name associated with a SNALINK LU0 DEVICE statement defined in the TCPIP profile configuration data set. Refer to the information on the PKTTRACE statement in the *OS/390 IBM Communications Server: IP Configuration Reference*.

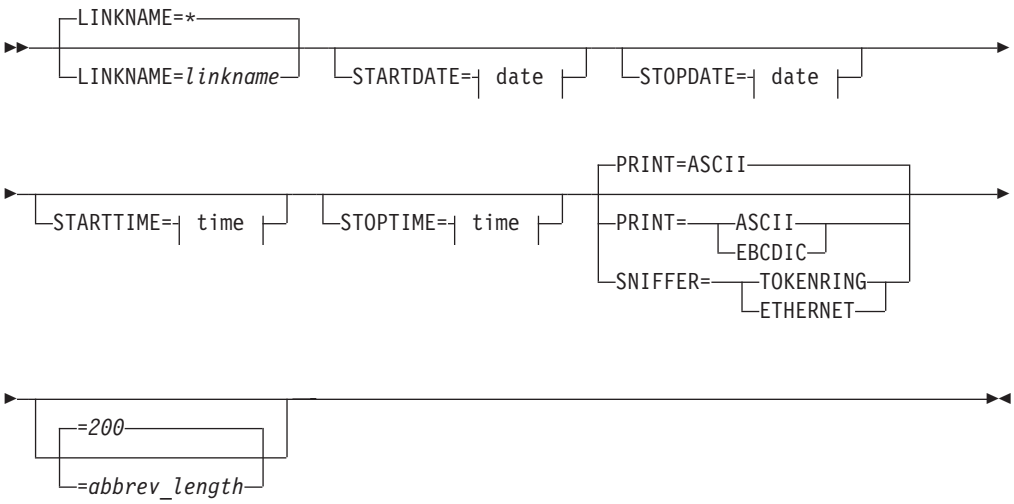
Formatting a Trace Report Using the TRCFMT Utility

Trace output can be generated by the TRCFMT utility, which uses the GTF data set as input.

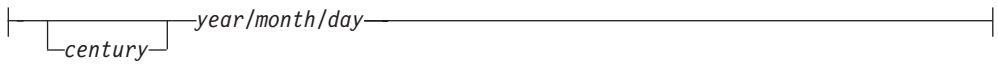
The TRCFMT utility has several options that allow the user to select the packets to be formatted and to specify how the trace output is to be formatted. The IP packets

that are formatted must meet all the conditions specified by the options passed to the utility. If no options are specified, all packet information stored in the GTF trace data set is formatted.

The syntax of the option for the TRCFMT utility is as follows:



date:



time:

LINKNAME Specifies the name of a link. The actual name specified depends on which device traced the packets. For devices in the TCPIP address space, specify the link name defined on the LINK statement. For SNALINK LU0, specify the LU name defined on the DEVICE statement in the hlq.PROFILE.TCPIP data set. rather than the associated link name. For SNALINK LU6.2 and X.25 NPSI, specify the link name on the LINK statement in the appropriate configuration data set. If the LINKNAME option is omitted or an "*" is specified, the options will apply to all links.

STARTDATE Specifies the start date for the trace report. Only packets that were captured on or after this date will appear in the report. The start date refers to the local date of the host where the packets were traced. If this option is omitted, formatting will start with the earliest packet traced.

The subfields of the value associated with this option are described below. Each of the subfields has a fixed length and must be supplied with leading zeros.

century
Specifies the century as a 2-digit number

year Specifies the year as a 2-digit number

month
Specifies the month as a 2-digit number

day Specifies the day as a 2-digit number

STOPDATE Specifies the stop date for the trace report. Only packets that were captured before or on this date will appear in the report. The stop date refers to the local date of the host where the packets were traced. If this option is omitted, formatting will stop with the last packet traced.

The subfields of the value associated with this option are described above in the description of the STARTDATE option.

STARTTIME Specifies the start time for the trace report. Only packets that were captured on or after this time will appear in the report. The start time refers to the local time of the host where the packets were traced. If this option is omitted, formatting will start with the earliest time, that is 00:00:00.000000.

The subfields of the value associated with this option are described below. Each of the subfields has a fixed length and must be supplied with leading zeros.

hour Specifies the hour as a 2-digit number

min Specifies the minutes as a 2-digit number

sec Specifies the seconds as a 2-digit number

microsec
Specifies the microseconds as a 2-digit number

STOPTIME Specifies the stop time for the trace report. Only packets that were captured before or at this time will appear in the report. The stop time refers to the local time of the host where the packets were traced. If this parameter is omitted, formatting will stop with the latest time, that is 23:59.59.999999.

The subfields of the value associated with this option are described above in the description of the STARTTIME option.

PRINT Specifies that the trace records are to be formatted as a text report for display on a terminal or printer. This option is the default.

ASCII Specifies that the uninterpreted fields in the IP packet are assumed to contain ASCII data. The data in these fields is therefore translated to displayable EBCDIC characters according to the following table before it is presented in the character equivalent dump. The ASCII keyword is the default.

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1_	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
2_	.	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_	P	Q	R	S	T	U	V	W	X	Y	Z	.	\	.	.	_
6_	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_	p	q	r	s	t	u	v	w	x	y	z	{	|	}	.	.

```
8_  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
9_  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
A_  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
B_  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
C_  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
D_  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
E_  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
F_  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
```

EBCDIC

If this keyword is specified, the uninterpreted fields in the IP packet are assumed to contain EBCDIC data. The data in these fields is translated, according to the following table, only to remove nondisplayable EBCDIC characters before it is presented in the character-equivalent dump.

```
     _0  _1  _2  _3  _4  _5  _6  _7  _8  _9  _A  _B  _C  _D  _E  _F
0_   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
1_   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
2_   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
3_   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
4_   .   .   .   .   .   .   .   .   .   .   ¢   .   <   (   +   |
5_   &   .   .   .   .   .   .   .   .   .   !   $   *   )   ;   ¬
6_   -   /   .   .   .   .   .   .   .   .   ¦   ,   %   _   >   ?
7_   .   .   .   .   .   .   .   .   .   ’   :   #   @   ‘   =   "
8_   .   a   b   c   d   e   f   g   h   i   .   .   .   .   .   .
9_   .   j   k   l   m   n   o   p   q   r   .   .   .   .   .   .
A_   .   .   s   t   u   v   w   x   y   z   .   .   .   .   .   .
B_   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
C_   {   A   B   C   D   E   F   G   H   I   .   .   .   .   .   .
D_   }   J   K   L   M   N   O   P   Q   R   .   .   .   .   .   .
E_   \   .   S   T   U   V   W   X   Y   Z   .   .   .   .   .   .
F_   0   1   2   3   4   5   6   7   8   9   .   .   .   .   .   .
```

SNIFFER Formats the trace records so that they are suitable for downloading to a DatagLANce Network Analyzer or a Sniffer Network Analyzer for analysis.

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 will be displayed. Packets from all devices can be displayed using this option.

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 will be displayed. Packets from all devices can be displayed using this option.

Specifies that a truncated portion of the IP packet is to be formatted. If a length is specified, it must be an integer in the range from 0 to 65535 inclusive and it nominates the maximum number of bytes that are formatted from each packet. If no length is supplied, the first 200 bytes of each packet are formatted.

As shown in the syntax diagram, the date and time selection criteria can be specified separately. The date and time options are tested independent of each other. Thus, if a date specification is given without a time specification, only the date when the packet was traced is examined to see if it meets the criteria.

Similarly, if a time specification is given without a date specification, only the time when the packet was traced is examined to see if it meets criteria. This will, for example, allow the same time slot to be examined over more than one day.

The TRCFMT program uses three optional ddnames to specify the input, output, and error data sets during processing. If data sets other than the default are desired, the ddnames must be assigned to the required data sets when invoking the TRCFMT utility.

The ddnames are:

FMTIN
: Identifies the input GTF trace data set. If this ddname is omitted, the default data set is SYS1.TRACE.

FMTOUT
: Identifies the destination for the trace report for the PRINT option or the destination of the analyzer records for the SNIFFER option. If this ddname is omitted, the default data set is TCPIP.TRACE.OUTPUT.

  The data set associated with this ddname must have the attributes of variable-length records and logical record length of 137 bytes.

FMTERR
: Identifies the destination for error messages produced when creating a DatagLANce Network Analyzer or a Sniffer Network Analyzer output file in FMTOUT. Reporting of inconsistencies in an IP packet is an integral part of the trace report generated by the PRINT option, but these cannot be sent to FMTOUT when it is to contain data in a format to be read by a DatagLANce Network Analyzer or a Sniffer Network Analyzer. If this ddname is omitted, the default data set is TCPIP.TRACE.ERROR. This ddname is only required for the SNIFFER option.

  The data set associated with this ddname must have the attributes of variable-length records and logical record length of 137 bytes.

The TRCFMT utility can be invoked in either the TSO environment or as a batch job using JCL statements.

TRCFMT in the TSO Environment: Figure 30 shows the steps required to use TRCFMT in the TSO environment.

```
System:   READY
   User:   allocate dd(fmtin) ds('sys1.trace')
System:   READY
   User:   allocate dd(fmtout) ds(output.trc)
System:   READY
   User:   trcfmt linkname=tr1 sniffer=ethernet starttime=08:30 stoptime=17:30
System:   EZA2070W OPENFILE: DDname FMTERR not defined by user, using default
System:   EZA2072I OPENDD:   FMTERR DDname has been defined to TCPIP.TRACE.ERROR

System:   READY
```

Figure 30. Invoking TRCFMT from TSO

In the figure, TRCFMT reads input from the GTF trace data set SYS1.TRACE, the output is written to the data set userid.OUTPUT.TRC, and error messages are written to the default data set TCPIP.TRACE.ERROR.

```
  TRCFMT processing records using the following selection criteria:
      Linkname=*                                ABBREV=NONE
      Start date=*                    Start time=*
      Stop  date=*                    Stop  time=*
      Print = ASCII


  TCPIP Packet Trace formatting routine, Version 3.00 for MVS

 1  PKT  0000001 DATE=94/04/15 TIME= 9:06:09.422725
       TO   LINK=L622             DEV=SNA_LU6.2
  IP   SRC=1.3.1.3            DST=1.3.1.2            VIA=1.3.1.2
       VER=4  HDLEN=5  TOS=X'00' TOTLEN=276   ID=4     FLAGS=B'000'
       FRAGOFF=0    TTL=60   PROTOCOL=ICMP        CHECKSUM=X'79DB'
  ICMP ECHO_REQ          CHECKSUM=X'F9CF' ID=902    SEQ=14144
  DATA LEN=248
       9F558F7D AF95FCA8 3D147846 127B3A56 ED45A121 *.U.} .... =.xF .{;V .E.!*
       C472BE46 C92D57F6 10D711F2 6DAAC1F8 96AD894B *.r.F .-W. .... m... ...K*
       E293D49E A2BE13FB 34F82362 BC5F387C 0D67B681 *.... .... 4.#b ._8| .g..*
       6E1733E2 A94FA9E5 246E7212 4D4C194D 463B87C4 *n.3. .O.. $nr. ML.M F;..*
       0A2C1E51 2BD9E5F0 A26DFCD5 2D494831 BF7D7EBD *.,.Q +... .m.. -IH1 .}~.*
       40EE5540 56E57393 4305CA6A E75256D8 C6AE4FA6 *@.U@ V.s. C..j .RV. ..O.*
       F8B10D2C 17793F3D 82E44F70 3D5206B6 DE3FD102 *..., .y?= ..Op =R.. .?..*
       F452F421 754AF90B F0803244 5CC33EF4 3306D523 *OR.! uJ.. ..<D \.>. 3..#*
       87DE6EBA 6030ABF6 BAD669DE 8C077951 D0F88513 *..n. '0.. ..i. ..yQ ....*
       35DEC1C6 C1DB46AB E42666A3 F2DC11EB 97B36039 *5... ..F. .&f .... ..'9*
753385DD 7977E93E D00B1E6F 71144771 41389A58 *u3.. yw.> ...o q.Gq A8.X*

7FD6A671 E26FAEA9 846A2EAF E4247160 7DDD7E5B *...q .o.. .j.. .$q' }.~.*

89971613 A63160BA                          *.... .1'.               *
 2  PKT  0000002 DATE=94/04/15 TIME= 9:06:09.448764
       FROM LINK=L622             DEV=SNA_LU6.2
  IP   SRC=1.3.1.2            DST=1.3.1.3
       VER=4  HDLEN=5  TOS=X'00' TOTLEN=276   ID=4     FLAGS=B'000'
       FRAGOFF=0    TTL=60   PROTOCOL=ICMP        CHECKSUM=X'79DB'
  ICMP ECHO_REPLY        CHECKSUM=X'01D0' ID=902    SEQ=14144
  DATA LEN=248
       9F558F7D AF95FCA8 3D147846 127B3A56 ED45A121 *.U.} .... =.xF .{;V .E.!*
       C472BE46 C92D57F6 10D711F2 6DAAC1F8 96AD894B *.r.F .-W. .... m... ...K*
       E293D49E A2BE13FB 34F82362 BC5F387C 0D67B681 *.... .... 4.#b ._8| .g..*
       6E1733E2 A94FA9E5 246E7212 4D4C194D 463B87C4 *n.3. .O.. $nr. ML.M F;..*
       0A2C1E51 2BD9E5F0 A26DFCD5 2D494831 BF7D7EBD *.,.Q +... .m.. -IH1 .}~.*
       40EE5540 56E57393 4305CA6A E75256D8 C6AE4FA6 *@.U@ V.s. C..j .RV. ..O.*
       F8B10D2C 17793F3D 82E44F70 3D5206B6 DE3FD102 *..., .y?= ..Op =R.. .?..*
       F452F421 754AF90B F0803244 5CC33EF4 3306D523 *OR.! uJ.. ..<D \.>. 3..#*
       87DE6EBA 6030ABF6 BAD669DE 8C077951 D0F88513 *..n. '0.. ..i. ..yQ ....*
       35DEC1C6 C1DB46AB E42666A3 F2DC11EB 97B36039 *5... ..F. .&f .... ..'9*
       753385DD 7977E93E D00B1E6F 71144771 41389A58 *u3.. yw.> ...o q.Gq A8.X*
       7FD6A671 E26FAEA9 846A2EAF E4247160 7DDD7E5B *...q .o.. .j.. .$q' }.~.*
       89971613 A63160BA                          *.... .1'.
```

Figure 31. IP Packet Trace Output for SNALINK LU6.2

The trace output in Figure 31 was done on the local SNALINK LU6.2 host. Packets are traced as FROM and TO the local host. This trace output enables the identification of IP packet flow and shows the data content of the packets.

The packets that contain data display the data in hexadecimal digits and, in this case, both their EBCDIC and ASCII characters.

The message flow for the trace in the Figure is:

1 The local host sends an Echo Request packet (packet 1) that is 256 bytes in length, including the ICMP header, to the remote host.

2 The remote host returns the local host Echo Request packet (packet 1) as an identical Echo Reply packet (packet 2).

TRCFMT As a Batch Job: Figure 32 shows a set of JCL statements to invoke TRCFMT with a batch job.
In this example, TRCFMT reads its input from the data set SYS1.TRACE. Its output

```
//JOB#1   JOB 'JOB1','TRCFMT BATCH',MSGLEVEL=(1,1),MSGCLASS=X,REGION=4096K,
//    CLASS=A
//*-------------------------------------------------------------------------
//*
//*  IKJEFT01 - RUN A TSO COMMAND IN BATCH
//*
//*-------------------------------------------------------------------------
//IKJEDF01 EXEC PGM=IKJEFT01
//FMTIN    DD DSN=SYS1.TRACE,DISP=SHR
//FMTOUT   DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
TRCFMT STARTTIME=09:00 PRINT=EBCDIC
/*
//
```

Figure 32. Invoking TRCFMT in a Batch Job

and error messages are written to system-managed spool files.

Generating a File for the DatagLANce or Sniffer Network Analyzer: The TRCFMT utility supports the generation of a file in a format suitable for downloading to a DatagLANce Network Analyzer or a Sniffer Network Analyzer. This feature is selected with the SNIFFER option, which requires specification of either the ETHERNET or TOKENRING suboption to indicate the format of the file to be generated for the analyzer. Refer to “Formatting a Trace Report Using the TRCFMT Utility” on page 264 for information about these suboptions. Once the file is generated, it is downloaded as a binary file to the analyzer and loaded using the standard features of the analyzer. If you are using the Ethernet analyzer application, the DOS file type must be ENC. If you are using the token-ring analyzer application, the DOS file type must be TRC.

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 *OS/390 IBM Communications Server: IP Configuration Reference*.

DEBUG Trace Example

Figure 33 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 current entry, followed by the address of the last entry in the trace. The rest of the table contains entries.

Use the information following the trace to interpret the entry types and their meaning.

```
0000E660 TO 0000E84F (X'000001F0' bytes)--All bytes contain X'00'
0000E850   00000000  00000000  E3D9C3E3  C2D34040   | ........TRCTBL  |
0000E860   0000F888  000126E8  A8448AE1  C45E1D01   | ..8h...Yy...D;..|
0000E870   E3C3D7C9  C2F74040  03000000  031E0000   | TCPIB7  ........|
0000E880   00059000  00000068  A8448AE1  C46B2201   | ........y...D,..|
0000E890   E3C3D7C9  C2F74040  10220000  00018860   | TCPIB7  ......h-|
0000E8A0   00059000  0000006A  A8448AE1  C4A03801   | ........y...D...|
0000E8B0   E3C3D7C9  C2F74040  22000000  00018860   | TCPIB7  ......h-|
0000E8C0   00000000  00000000  A8448AE1  C6C28103   | ........y...FBa.|
0000E8D0   E3C3D7C9  C2F74040  23100000  000188D0   | TCPIB7  ......h}|
0000E8E0   00071000  0000002C  A8448AE1  C6D43E03   | ........y...FM..|
0000E8F0   00000000  00000000  0E040068  00000000   | ................|
0000E900   00000000  00000000                       | ........
   .
   .
   .
   .
0000F800                       A8448AE2  ECE40E00            y..S.U..|
0000F810   E3C3D7C9  C2F74040  10220000  00018860   | TCPIB7  ......h-|
0000F820   00059000  00000038  A8448AE2  ED641A01   | ........y..S....|
0000F830   E3C3D7C9  C2F74040  0F098802  01341234   | TCPIB7  ..h.....|
0000F840   07836B88  00000062  A8448AE2  F7B1C602   | .c,h....y..S7.F.|
0000F850   00000000  00000000  0E050000  00000000   | ................|
0000F860   00000000  00000000  A8448AE2  F7B21602   | ........y..S7...|
0000F870   E3C3D7C9  C2F74040  03000000  032A0000   | TCPIB7  ........|
0000F880   00061000  00000060  A8448A43  C202C303   | .......-y...B.C.|
0000F890   00000000  00000000  0E050000  00000000   | ................|
0000F8A0   00000000  00000000  A8448A43  C202F603   | ........y...B.6.|
0000F8B0   E3C3D7E3  C4F44040  03000000  BCD50000   | TCPTD4  .....N..|
0000F8C0   00019000  00000036  A8448A43  C20CEC03   | ........y...B...|
0000F8D0   E3C3D7E3  C4F44040  10220000  00018EA0   | TCPTD4  ........|
0000F8E0   00019000  00000038  A8448A43  C2474300   | ........y...B...|
0000F8F0   E3C3D7E3  C4F44040  22000000  00018EA0   | TCPTD4  ........|
0000F900   00000000  00000000                       | ........
   .
   .
   .
   .
```

Figure 33. Example of a SNALINK LU0 DEBUG Trace

The layout of a SNALINK trace table entry is shown in Table 31.

Table 31. Format of a SNALINK Trace Table Entry

Bytes	Definition
00–07	TOD time stamp
08–0F	LU name, if any

Table 31. Format of a SNALINK Trace Table Entry (continued)

Bytes	Definition
10	Entry Type **Value** 0F DLC Interrupt 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
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
18–1B	VTAM Send/Receive/DLC buffer address
1C–1F	VTAM Send/Receive/DLC buffer length

Chapter 14. Diagnosing SNALINK LU6.2 Problems

The SNALINK LU6.2 interface uses the LU type 6.2 protocol to establish a point-to-point connection across an 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 34.

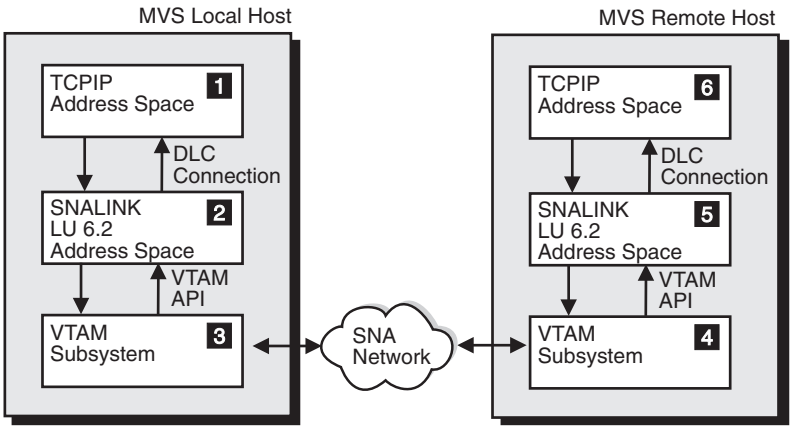


Figure 34. Components of an SNALINK LU6.2 Connection on MVS

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 via a DLC connection.
6. The data is unencapsulated and processed by the TCP/IP address space.

Setting Up a SNALINK LU6.2 Network

Completing the following steps will establish the system described in Figure 34. Each step should be completed in the order given.

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 *OS/390 IBM 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 285 for details about how to verify SNALINK LU6.2 connections.

The example in Figure 35 shows the messages that are expected when the SNALINK LU6.2 address space is started and a network connection is established.

```
16.43.55 STC02790  $HASP373 L62LINK  STARTED
a 16.43.55 STC02790  IEF403I L62LINK - STARTED - TIME=16.43.55
b 16.43.56 STC02790  EZA5927I LU62CFG : No Errors Detected -
                                        Initialization will continue
c 16.43.57 STC02790  EZA5932I Initialization complete - Applid:
                            X3938L61  TCP/IP: V2R10
d 16.44.00 STC02790  EZA5935I Send conversation allocated for 1.3.1.2
d 16.44.01 STC02790  EZA5936I Receive conversation allocated for 1.3.1.2
e 16.44.25 STC02790  EZA5933I Link L62      opened
```

Figure 35. 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.

a The L62LINK address space has been started.

b The SNALINK LU6.2 configuration data set for the L62LINK address space has been successfully parsed.

c The L62LINK address space displays its local VTAM application LU and the TCP/IP address space name to which it will connect.

d The L62LINK address space establishes a network connection through the VTAM API.

e The L62LINK 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.

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 and are helpful in identifying 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, refer to “Network Connection Establishment Problems” on page 280 for a detailed list of network connection problems and their solutions.

2. **Have you completed all the required definitions?**

   See “Setting Up a SNALINK LU6.2 Network” on page 273 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 286 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 286 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 will not start, see “Problems Starting the SNALINK LU6.2 Address Space” 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 293 to determine the reason for the configuration errors. Text in the message documentation will specify 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 285 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 278 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 *OS/390 IBM Communications Server: SNA Messages* and *OS/390 IBM 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 an SNALINK LU6.2 address space starting problem, note any error messages or abend codes that are displayed on the MVS system console.

Analysis

The following is a list of some of the common SNALINK LU6.2 address space startup problems. Each error symptom is listed with possible causes and resolutions.

1. **The message "Errors Detected - Address Space will Terminate" has been displayed on the MVS system console with no other error messages.**

   Cause: This error message indicates that an error has occurred with the SNALINK LU6.2 configuration data set.

   Resolution: 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 *OS/390 IBM Communications Server: IP Configuration Reference* for details about the SNALINK LU6.2 configuration data set statement syntax.

2. **The message "Error in open of LU62CFG - no data will be read" has been displayed on the MVS system console.**

   Cause: The SNALINK LU6.2 address space cannot access a SNALIN LU6.2 configuration data set. The LU62CFG ddname might have been omitted from the SNALINK LU6.2 cataloged procedure.

   Resolution: 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 *OS/390 IBM Communications Server: IP Configuration Reference* for an example of a SNALINK LU6.2 cataloged procedure.

3. **The message "Address Space Already Active - this Address Space will Terminate" has been displayed on the MVS system console.**

   Cause: An address space with the same name as the SNALINK LU6.2 address space is already active.

   Resolution: 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.

4. **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.**

   Cause: 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.

   Resolution: This problem might be resolved by one of the following solutions:

   a. 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 286 for a detailed description of the VTAM operations that display the status of VTAM application LUs and activate them.

   b. 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 *OS/390 IBM Communications Server: IP Configuration Reference* for details about the SNALINK LU6.2 VTAM statement syntax and the SNALINK LU6.2 VTAM application LU definition.

5. **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.**

   Cause: VTAM security is not allowing the SNALINK LU6.2 address space to access the VTAM application LU.

   Resolution: 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 *OS/390 IBM Communications Server: IP Configuration Reference* for details about the SNALINK LU6.2 VTAM statement syntax and the SNALINK LU6.2 VTAM application LU definition.

6. **The SNALINK LU6.2 address space abends with a system abend code of 300 after the "SNALINK LU6.2 address space STARTED" message.**

   Cause: The abend code of 300 indicates that there is insufficient storage for the SNALINK LU6.2 address space.

   Resolution: 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 *OS/390 IBM Communications Server: SNA Messages* and *OS/390 IBM Communications Server: IP and SNA Codes* for detailed SNALU6.2 abend code descriptions.

7. **The SNALINK LU6.2 address space abends with an abend code of S0F8 after the "Initialization Complete..." message.**

   Cause: 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.

   Resolution: The SRBEXIT option in the VTAM application LU definition has been set to "Yes." Correct the VTAM application LU definition.

   Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for details about the SNALINK LU6.2 VTAM application LU definition.

If the SNALINK LU6.2 startup problem has not been found after using these analysis steps, 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 293 and "Finding Error Message Documentation" on page 293 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 is 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.

Documentation

To check the status of the DLC connection, perform the following diagnosis steps:

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 285 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

The following list contains some of the common DLC connection problems between the SNALINK LU6.2 address space and the TCP/IP address space. Each error symptom is listed with possible causes and resolutions.

1. **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.”**

   Cause: 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.

   Resolution: Check if the SNALINK LU6.2 address space is active and start it, if necessary.

2. **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:

   Cause: 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.

   Resolution: 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 to see if a valid hlq.TCPIP.DATA data set has been specified as the SYSTCPD ddname in the SNALINK LU6.2 cataloged procedure and correct it, if necessary.

   Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for an example of a SNALINK LU6.2 catalogued procedure.

   If a valid hlq.TCPIP.DATA data set has been used, check the TCP/IP address space specified in the TCPIPJOBNAME statement within it.

   Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for a detailed description of the TCPIPJOBNAME statement in the hlq.PROFILE.TCPIP.

   Cause: The SNALINK LU6.2 address space might be rejecting the connect attempt from the TCP/IP address space because of an incorrectly-defined SNALINK LU6.2 link name.

   Resolution: 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 *OS/390 IBM Communications Server: IP Configuration Reference* for details about the SNALINK LU6.2 LINK statement syntax and the TCPIP LINK statement syntax.

3. **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."**

   Cause: 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.

   Resolution: 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 *OS/390 IBM 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. There are three ways to initiate the establishment of a network connection.

1. 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.
2. 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.
3. Connections can be established using the SNALINK LU6.2 RESTART MODIFY subcommand.

Documentation

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 285 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 will attempt to establish the LU type 6.2 connection between the SNALINK LU6.2 devices. During connection establishment, any problems will cause 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

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.

1. **The SNALINK LU6.2 address space issued error message: “Unable to allocate send conversation.”**

   Cause: 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.

   Resolution: The APPC option in the VTAM application LU definition must be set to YES to enable LU type 6.2 conversations.

   Cause: 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.

   Resolution: 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 286 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 *OS/390 IBM Communications Server: SNA Messages* and *OS/390 IBM Communications Server: IP and SNA Codes*. These messages might indicate a network configuration or hardware error.

2. **VTAM error message output to the MVS system console: “REQUIRED LOGMODE NAME UNDEFINED.”**

   Cause: 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.

   Resolution: 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.

Network Connection Loss Problems

SNA network connection loss can be either expected or unexpected. This section deals with unexpected connection-loss 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 time-out 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 time-out 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 285 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 293 for details on finding the documentation for these messages. Text in the message documentation specifies the action required to fix the problem.

The following is an example of an 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: 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 will issue a negative response with sense 800A0000 (PIU too long), 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 080A0000. The definitions used in the NCP and SNALINK must be such that MAXRU is at least 29 bytes less than MAXDATA. Refer to *OS/390 IBM 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.

Documentation

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 285 for details. When listing the connection status, the number of packets sent and received over the connection since establishment is displayed in the report. The following steps help you determine the source of the 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.

Note: IP packet trace, as described in “Using IP Packet Trace” on page 291, can also be used to trace and validate the IP data packets as they enter and leave the SNALINK LU6.2 address space.

Analysis

The following list contains some of the common SNALINK LU6.2 data-loss problems. Each error symptom is listed with possible causes and resolutions.

1. **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:

   Cause: The DLC link between the TCP/IP address space and the SNALINK LU6.2 address space might not be active.

   Resolution: See “DLC Connection Problems” on page 278 to diagnose the DLC link problem.

   Cause: The SNALINK LU6.2 address space might be discarding packets.

   Resolution: 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 293 for details on finding the documentation for these messages. Text in the message documentation specifies the action required to fix the problem.

2. **Data packets are actually not lost but the protocol (PING) times out.**

   Cause: 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.

   Resolution: 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 will be 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 time-out value or the connection time-out value should be increased.

3. **Data packets are lost between the SNALINK LU6.2 devices.**

   Cause: The network is failing.

   Resolution: Check for VTAM error messages on the MVS system console. See “VTAM Buffer Traces” on page 292 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.

Note: 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. Once 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 291.

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 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.

Note: The TCP/IP address space must be started before the NETSTAT command can query the connection status.

The command `MODIFY DEVLINKS` displays the devices and links that have been defined to the main TCP/IP address spaces and the status of these devices (whether or not they are active).

Figure 36 is a sample of output from the NETSTAT DEVLINKS command.

```
Device OS2_DEV           Type: SNA LU 6.2  Status: Inactive
Queue size: 0            Jobname: SNA62OS2
   Link OS2_LINK         Type: DLC        Net number: 1

Device SNALU62A          Type: SNA LU 6.2  Status: Will retry connect
Queue size: 0            Jobname: L62LINK
   Link L621             Type: DLC        Net number: 1

Device SNALU62B          Type: SNA LU 6.2  Status: Connected
Queue size: 0            Jobname: L62LINK
   Link L622             Type: DLC        Net number: 2

Device SNALU62C          Type: SNA LU 6.2  Status: Issued Connect
Queue size: 0            Jobname: L62LINK
   Link L623             Type: DLC        Net number: 3
```

Figure 36. NETSTAT DEVLINKS Output Example

The example shows four SNALINK LU6.2 devices and associated links known to TCP/IP. The Jobname field contains the name of the SNALINK LU6.2 address space associated with the SNALINK LU6.2 device.

The most significant field for diagnosing DLC connection problems is the Status field. Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for detailed interpretation of the device status and its importance in the SNALINK LU6.2 DLC connection.

For more details about the usage, parameters, and output of the NETSTAT command, refer to the *OS/390 IBM Communications Server: IP User's Guide*.

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
```

where 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 37 shows a sample output from the subcommand.

```
EZA5971I LIST Accepted; Range = Single Connection
EZA5967I   192.9.2.4 (Connected on 94.109 at 10:30:56)         045
EZA5968I     Connected via:  RESTART         Trace Level: OFF
EZA5969I     SEND:-  Status: Allocated       Packets Out: 0
EZA5970I     RECV:-  Status: Allocated       Packets In:  0
EZA5974I LIST Completed
```

Figure 37. 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 293. Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* 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
```

where 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 *OS/390 IBM 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
```

where majnode_name is the major node name to deactivate.

See “Activating an LU” for the major node activation subcommand.

Refer to *OS/390 IBM 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
```

where node_name is the major or minor node name for which you want to display the status.

Displaying the status of a major node will list 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 a LU type 6.2 conversation.

Figure 38 on page 288 shows a sample of the output from an LU in connectable state.

```
IST097I DISPLAY ACCEPTED
IST075I NAME = MINORLU, TYPE = APPL 148
IST486I STATUS= CONCT, DESIRED STATE= CONCT
IST977I MDLTAB=***NA*** ASLTAB=***NA***
IST861I MODETAB=***NA*** USSTAB=***NA*** LOGTAB=***NA***
IST934I DLOGMOD=***NA***
IST597I CAPABILITY-PLU INHIBITED,SLU INHIBITED,SESSION LIMIT NONE
IST654I I/O TRACE = OFF, BUFFER TRACE = OFF
IST271I JOBNAME = ***NA***, STEPNAME = ***NA***
IST171I ACTIVE SESSIONS = 0000000000, SESSION REQUESTS = 0000000000
IST172I NO SESSIONS EXIST
IST314I END
```

Figure 38. 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 39 shows a sample of the output from a DISPLAY command for an LU in active state.

```
IST097I DISPLAY ACCEPTED
IST075I NAME = MINORLU, TYPE = APPL 148
IST486I STATUS= ACTIV, DESIRED STATE= ACTIV
IST977I MDLTAB=***NA*** ASLTAB=***NA***
IST861I MODETAB=LU62LMTB USSTAB=***NA*** LOGTAB=***NA***
IST934I DLOGMOD=LU62LOGM
IST597I CAPABILITY-PLU ENABLED  ,SLU ENABLED  ,SESSION LIMIT NONE
IST654I I/O TRACE = OFF, BUFFER TRACE = OFF
IST271I JOBNAME = SNALNK62, STEPNAME = SNALNK62
IST171I ACTIVE SESSIONS = 0000000003, SESSION REQUESTS = 0000000000
IST206I SESSIONS:
IST634I NAME     STATUS         SID          SEND RECV VR TP NETID
IST635I REMOTELU ACTIV-S    CC4B1D5168E1815A 0001 0000  0  2 IBMNETXA
IST635I REMOTELU ACTIV-S    CC4B1D5168E18159 0001 0001  0  2 IBMNETXA
IST635I REMOTELU ACTIV-P    CC4F225168686526 0000 0001  0  0 IBMNETXA
IST314I END
```

Figure 39. DISPLAY Subcommand Output Example for Active LU

This example shows that the SNALINK LU6.2 address space (SNALNK62) has been started successfully and has its local LU (MINORLU) in use with three sessions active to a remote LU (REMOTELU).

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 *OS/390 IBM Communications Server: SNA Operation* for more information about the DISPLAY command.

Traces

The following traces can be used to obtain information about the data flows and actions of the SNALINK LU6.2 network connection. 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 will start internal tracing:

```
MODIFY procname, TRACE DETAIL ALL
```

where procname is the member name of the cataloged procedure used to start the local SNALINK LU6.2 address space.

Refer to *OS/390 IBM Communications Server: IP Configuration Reference* for detailed descriptions of the TRACE statement parameters and the TRACE subcommand parameters.

```
SYSTCPD : Line 38: TCPIP address space name set to TCPCS
  LU62CFG : Starting Pass 1 of 2
  LU62CFG : Starting Pass 2 of 2
  LU62CFG : Line 15: LINK statement - Blank Idle Timeout, 0 assumed
  LU62CFG : Line 17: LINK statement - Blank Idle Timeout, 0 assumed
1 LU62CFG : No Errors Detected - Initialization will Continue
  Initialization complete - Applid: X3938L61  TCP/IP: TCPCS
6 CNOS for independent partner; SESSLIM=0002, WINNER=0001,
           LOSER=0001
    OLU= X3938L61, DLU= X383BL61, IP address= 1.2.1.2
  OPRCNOS  err. R15 00000000 R0 0000000B RTNCD 00000000 FDBK2 0000000B
  OPRCNOS  err.  RCPRI= 0008,  RCSEC= 0000
  OPRCNOS  Sense code received:  087D0001
           Sense code specified: 00000000
  Unable to complete CNOS on LU X383BL61 for 1.2.1.2
2 Conversations for 1.2.1.2 terminated
  Unable to allocate send conversation for 1.2.1.2
7 CNOS for independent partner; SESSLIM=0002, WINNER=0001,
          LOSER=0001
    OLU= X3938L61, DLU= X3839L63, IP address= 1.3.1.2
2 Send conversation allocated for 1.3.1.2
  VTAM conversation allocated; Convid= 010000C3, SID= F04B1A51695651C1
    OLU= X3938L61, DLU= X3839L63, IP address= 1.3.1.2
  Connection 1.3.1.2 will timeout in 0 seconds
  Receive conversation allocated for 1.3.1.2
  VTAM conversation allocated; Convid= 010000C4, SID= 004B1B516957467F
    OLU= X3839L63, DLU= X3938L61, IP address= 1.3.1.2
  Connection 1.3.1.2 will timeout in 0 seconds
1 Link L622     opened
4 IP datagram added to the VTAM send queue, length= 276,
        queue count= 1
     LU= X3839L63, Linkname= L622    , IP address = 1.3.1.2
3 VTAM sent logical record; Convid= 010000C3, SID=
        F04B1A51695651C1, length= 280
     OLU= X3938L61, DLU= X3839L63, IP address= 1.3.1.2
  Number of IP packets sent on 1.3.1.2 = 1
6    450001140002000003C0179DD0103010301030102080032900386374O
          C521A047E0A474CDE4014469
    70F7ECA6577AF7E7C98A28B3D28B152730E78CF9C39EB1651D187E1EF935F43
          ADCC89DB647CA288E
    50DBBAF3BDC9B32C6F6659B1A9846B26DED06BDAA37C9DFE96A7AC79A8BF074
          3966346240EB7D349
    664C02D818C37E7105A06530A619F261F0265602CB68EE2B3AA417020CE6B8F
          5B5F99E80904EC91E
    7C1B12A72E715AA0102C187559A012244761A009552307E1DB276B869CFAB10
          9477CFCC591029435
    641F0EBCB62D6B7CD7A577BA547AF9D7276F6E29CB3FE6842F8FA3CBC5BFE7F
          C591A36CD2583F676
    EBC69ACAD273A6D391F3E5D640D3F49A97076C9C151695961421531B3AB6AE6
          03C649A2E00000000
3 VTAM received logical record; Convid= 010000C4, SID= 004B1B516957467F, length= 280

    OLU= X3839L63, DLU= X3938L61, IP address= 1.3.1.2
  Number of IP packets received on 1.3.1.2 = 1
```

Figure 40. SNALINK LU6.2 Internal Trace Output (Part 1 of 2)

```
6   4500011400020000 3C0179DD010301020103010300003A9003863740C
          521A047E0A474CDE4014469
    70F7ECA6577AF7E7C98A28B3D28B152730E78CF9C39EB1651D187E1EF935F43A
          DCC89DB647CA288E
    50DBBAF3BDC9B32C6F6659B1A9846B26DED06BDAA37C9DFE96A7AC79A8BF0743
          966346240EB7D349
    664C02D818C37E7105A06530A619F261F0265602CB68EE2B3AA417020CE6B8F5
          B5F99E80904EC91E
    7C1B12A72E715AA0102C187559A012244761A009552307E1DB276B869CFAB109
          477CFCC591029435
    641F0EBCB62D6B7CD7A577BA547AF9D7276F6E29CB3FE6842F8FA3CBC5BFE7FC
          591A36CD2583F676
    EBC69ACAD273A6D391F3E5D640D3F49A97076C9C151695961421531B3AB6AE60
          3C649A2E00000000
5   IP datagram packed into message, length= 276
    LU= X3839L63, Linkname= L622    , IP address = 1.3.1.2
  Received operator shutdown request
2   Link L622     closed
```

Figure 40. 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 *OS/390 eNetwork Communications Server: SNA Programmers LU 6.2 Guide* for more information about CNOS processing.

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. The following

command, when passed to the SNALINK LU6.2 interface, starts the SNALINK LU6.2 address space packet trace function:

```
MODIFY procname,PKTTRACE ON LINKNAME=link_name
```

where procname is the member name of the cataloged procedure used to start the local SNALINK LU6.2 address space and link_name is the local TCP/IP host SNALINK LU6.2 link name.

See “Chapter 5. TCP/IP Services Traces and IPCS Support” on page 47 for details about how to use the IP packet trace facility. Figure 31 on page 269 shows an IP packet trace for a local SNALINK LU6.2 host sending PING to a remote SNALINK LU6.2 host. It was formatted using the TRCFMT command. For details on this command, see “Formatting a Trace Report Using the TRCFMT Utility” on page 264.

Figure 41 shows an example of a CTRACE formatted packet trace record.

```
SYSM      PACKET    00000001  13:29:20.210227  Packet Trace
TO LINK   =  FDDIB0    DEV = LCS_FDDI             FULL
 TOD CLOCK = XB1B69613  5352AF02  TIME ZONE = XFFFFBCF1
 PKT 11744       LOST RECORDS = 0      HDR SEQUENCE NUM = 0
IP SRC = 152.85.63.3              IP DST = 152.85.156.24
 HDLEN = 5    TOS = X00  TOTLEN = 167    ID = 55401  FLAGS = (none)
 FRAGOFF = 0      TTL = 64   PROTOCOL = TCP    CHECKSUM = X9621  FFFF
TCP SRC PORT = 721                TCP DST PORT = 515
 SEQ NUM = 1481366540  ACK NUM =   23384773  FLAGS = ACK PSH
 HDLEN = 5    WINDOW = 32767  CHECKSUM = XDCAF  FFFF  URGENT PTR = 0
 HEADER LENGTH = X0028
0000 450000A7 D8690000 40069621 98553F03 *...xQ... .o.q...|E....i..@..!
0010 98559C18 02D10203 584BDC0C 0164D2C5 *q....J........KE|.U......XK..
0020 50187FFF DCAF0000                   *&."....        |P.......
  DATA LENGTH = X007F
0000 48545641 5359534D 2E434841 2E545641 *.......(........|HTVASYSM.CHA
0010 2E474F56 0A505650 53503032 31370A66 *..|..&.&.&......|.GOV.PVPSP02
0020 64664138 32365456 41535953 4D2E4348 *............(...|dfA826TVASYS
0030 412E5456 412E474F 560A5564 66413832 *.......|........|A.TVA.GOV.Ud
0040 36545641 5359534D 2E434841 2E545641 *.......(........|6TVASYSM.CHA
0050 2E474F56 0A4E5359 534D2F56 50532F49 *..|..+...(..&...|.GOV.NSYSM/V
0060 444D5955 2E49444D 59552E4A 4F423033 *.(.....(....|...|DMYU.IDMYU.J
0070 3339392E 44303030 30303041 2E3F0A   *.............. |399.D000000A
```

Figure 41. 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 *OS/390 IBM Communications Server: SNA Diagnosis V2 FFST Dumps and the VIT*.

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 *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)* and *OS/390 IBM Communications Server: IP and SNA Codes* for detailed SNALU6.2 abend code descriptions.
- Sense codes in SNALINK LU6.2 error messages are generated by VTAM. Refer to *OS/390 IBM Communications Server: SNA Diagnosis V2 FFST Dumps and the VIT* and *OS/390 IBM 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 *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)* and *OS/390 IBM 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 *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)* and *OS/390 IBM 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 *OS/390 IBM Communications Server: SNA Diagnosis V2 FFST Dumps and the VIT* and *OS/390 IBM Communications Server: IP and SNA Codes* for descriptions of the VTAM error messages written to the MVS system console.

Chapter 15. Diagnosing Dynamic Domain Name Server (DDNS) Problems

This chapter describes how to diagnose problems involving the BIND-based dynamic domain name server (DNS). Problem diagnosis involving connection optimization is also described.

Note: For additional information on diagnosing problems with a BIND-based name server, refer to *DNS and BIND, 2nd Edition, Paul Albitz and Cricket Liu (Sebastopol, CA: O'Reilly & Associates, 1997), ISBN: 1-56592-236-0*.

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, please have available the output from syslogd and documentation from debug level 11.

Diagnosing Name Server Problems

The following methods are available for identifying name server problems:

- "Checking Messages Sent to the Operators Console"
- "Checking the Syslog Messages"
- "Using the onslookup and NSLOOKUP Commands" on page 296
- "Using the Debug Option with the Name Server" on page 296
- "Debugging with a Resolver Directive" on page 297
- "Using Name Server Signals" on page 297
- "Using the NSUPDATE Command" on page 298
- "Using Component Trace" on page 298

These methods are discussed in the following sections.

Checking Messages Sent to the Operators 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 (always EZZ6475N)
- Name server termination

For explanations of console messages, refer to *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)*.

Checking the Syslog Messages

Error messages can also be sent to a log file. You specify the name and location of the 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 *OS/390 IBM Communications Server: IP Configuration Guide*. For information about syslog messages, refer to *OS/390 IBM Communications Server: IP Messages Volume 3 (EZY-EZZ-SNM)*.

Using the onslookup and NSLOOKUP Commands

The onslookup and NSLOOKUP commands are helpful in diagnosing name-resolution problems in the OS/390 UNIX and TSO environments, respectively. Both commands query name servers with query packets similar to those of name servers.

The amount of information a name server provides depends on the debugging level. The lower the debugging level, the less information is provided. Level 1 provides basic information about timeouts and response packets. To turn on debugging at level 1, enter the following commands from the OS/390 UNIX shell:

onslookup

set debug

To turn the debugging off, enter the **set nodebug** command.

You can set the debugging option to level 2 by entering the following commands:

onslookup

set d2

In addition to level 1 information, level 2 displays the query packets that were sent. To turn d2 off, enter **set nod2**. Turning off d2 does not turn off level 1 debug. To turn off both d2 and debug, enter **set nodebug**.

For more information about the onslookup and NSLOOKUP commands, refer to the *OS/390 IBM Communications Server: IP User's Guide*.

Note: The onslookup command messages do not give a message ID for debugging and are not documented in the IBM Communications Server for OS/390 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. Valid levels for this option are in a range from one to 11, where 11 supplies the most information. Debugging information is sent to the file /tmp/named.run.

If named is started from the OS/390 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 OS/390 UNIX shell.

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. If the debug level is six or greater, the debug information exchanged during the last initiated zone transfer is written to /tmp/xfer.trace.

For details on the named command, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.

Debugging with a Resolver Directive

Programs that query name servers are called *resolvers*. To debug name-resolution problems, you can specify the debug option in the file /etc/resolve.conf (using the options *debug* directive) or in the TCP/IP configuration file. The resolver trace is sent directly into the output stream for the command using the resolver (for example, nslookup).

Using Name Server Signals

You can use OS/390 UNIX signals to send messages to the named daemon. The following signals are available:

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 **kill -USR2 $(cat /etc/named.pid)**

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.

Note: Signals do not affect zone transfers in progress. If debug is on, debugging for zone transfers occurs when the command named_xfer is invoked.

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 to use 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 OS/390 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 OS/390 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 OS/390 UNIX shell, and then check the file /tmp.named.run.
- To disable debugging, enter the **kill -USR2 $(cat /etc/named.pid)** command from the OS/390 UNIX shell.

- To turn query logging on, enter the **kill -WINCH $(cat /etc/named.pid)** command from the OS/390 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.

Using the NSUPDATE Command

The NSUPDATE command creates and executes Domain Name System (DNS) update operations on a host record. The -v option is used for debugging. It turns on verbose mode and displays all requests to and responses from the name server. To turn on debugging, enter the following commands from TSO:

NSUPDATE

set v

For details on the NSUPDATE command, refer to the *OS/390 IBM Communications Server: IP User's Guide* .

Using Component Trace

You can use the component-trace function to trace data at the TCP/IP layer. This information can be helpful in resolving name-resolution problems. For detailed information on the component-trace function, see "Chapter 5. TCP/IP Services Traces and IPCS Support" on page 47.

Return Codes

Following are the return codes, origination of the return codes, and explanations for the most common problems that you might encounter:

Return Code	Origin	Explanation
0	N/A	Successful.
-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.

Return Code	Origin	Explanation
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.

Diagnosing Problems with Connection Optimization

Connection optimization is a technique that uses DNS for balancing IP connections and workload in a sysplex domain. You may 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 *OS/390 IBM 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 *OS/390 IBM 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 may 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 *OS/390 IBM 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 is 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 *OS/390 IBM 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 16. Diagnosing REXEC, REXECD, and RSH Problems

This chapter contains diagnosis information about the classic (non-OS/390 UNIX) Remote Execution Protocol (REXEC), the Remote Execution Protocol Daemon (REXECD), and the remote shell client (RSH). Refer to "General Information about REXEC and RSH" for information about REXEC and RSH and to "General Information about REXECD" on page 305 for information about REXECD.

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.

Note: Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for information about defining the remote execution server.

Figure 42 shows the principle behind REXECD.

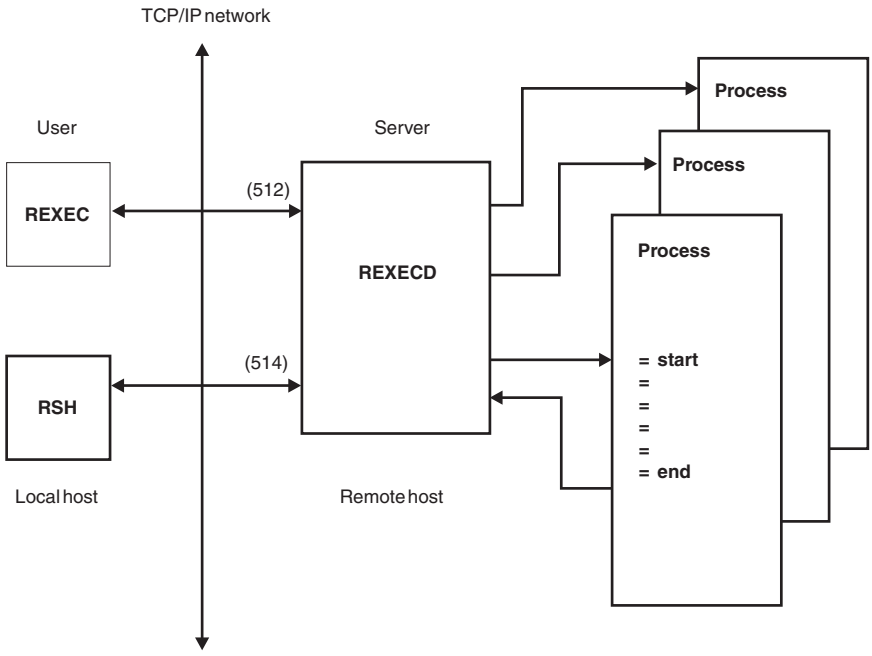


Figure 42. Remote Execution Protocol Principle

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, the following command needs to be specified:

```
REXEC -d
```

Note: Refer to the *OS/390 IBM Communications Server: IP User's Guide* for more information about REXEC commands.

REXEC Trace Example and Explanation

Figure 43 on page 303 shows an example of an REXEC trace. Short descriptions of the numbered items in the trace follow the figure.

Note: REXEC trace output is sent to the TSO console from which the command was submitted.

```
rexec -d -l walton -p terrica red time

 1
EZA4772I parms are -d -l walton -p terrica red time
   EZA4756I Variables have the following assignments:
   EZA4757I fhost   : red
   EZA4758I userid  : walton
   EZA4759I passwd  : terrica
 2
EZA4760I command : time
   EZA4801I MVS TCP/IP REXEC CS/390 V2R10
 3
EZA4780I calling GetHostResol with red
   EZA4791I Connecting to red 9.67.112.25, port REXEC (512) (8900)
   EZA4783I Passive Conn - OK(8520) on local port 1064
   EZA4785I passive open complete on port 0
   EZA4786I Active Conn - OK(8520) on local port 1065
   EZA4788I active  open complete on port 1
   EZA4774I rexec invoked;
 4
EZA4775I sending: 1064 walton terrica time
 5
EZA4776I D2 len 25
   EZA4738I getnextnote until DD
   EZA4739I Connection state changed (8681)
   EZA4740I Trying to open (8676)
   EZA4739I Connection state changed (8681)
   EZA4740I Open (8673)
   EZA4739I Data delivered (8682)
 6
EZA4743I Bytes in 76
 7
TIME-02:30:32 PM. CPU-00:00:00 SERVICE-1308 SESSION-00:00:00 FEBRUARY 8,
   1998
   EZA4739I Data delivered (8682)
   EZA4743I Bytes in 1
   EZA4739I Connection state changed (8681)
   EZA4740I Sending only (8675)
   EZA4739I Connection state changed (8681)
   EZA4740I Connection closing (8670)
   EZA4739I Connection state changed (8681)
   EZA4740I Sending only (8675)
   EZA4739I Connection state changed (8681)
   EZA4740I Connection closing (8670)
   EZA4751I Returning from recv_notices.
   EZA4778I returning from REXEC_UTIL
   EZA4789I rexec complete
 ***
```

Figure 43. Example of an REXEC Trace

Following are short descriptions of numbered items in the trace:

1 The REXEC parameters that are entered at the console or received from batch

2 The command that is to be sent to the remote host

3 The client is attempting to resolve the name to an IP address by calling GetHostResol.

4 The port number, user ID, password, and command sent to the REXECD

5 Length of data being sent

6 Length of data passed back to the REXEC client

7 Actual command output sent to the client

RSH Trace Example and Explanation

Figure 44 shows an example of an RSH trace. Short descriptions of numbered items in the trace follow the figure.

Note: RSH trace output is sent to the RSH console.

```
  rsh -d -l walton/terrica tiffany time
1
EZA5022I parms are RSH -d -l walton/terrica tiffany time
   EZA5006I Variables have the following assignments:
   EZA5007I fhost   : tiffiany
   EZA5049I locuser : adrian
   EZA5008I userid  : walton/terrica
2
EZA5010I command : time
3
EZA5030I calling GetHostResol with tiffany
   EZA5041I Connecting to tiffany 9.67.112.25, port RSH (514) (8902)
   EZA5033I Passive Conn - OK on local port 1023
   EZA5035I passive open complete on port 0
   EZA5036I Active Conn - OK on local port 1023
   EZA5038I active  open complete on port 1
   EZA5046I rsh invoked;
4
EZA5025I sending: 1023 adrian walton/terrica time
5
EZA5026I D2 len 34
   EZA4988I getnextnote until DD
   EZA4989I Connection state changed (8681)
   EZA4990I Trying to open (8676)
   EZA4989I Connection state changed (8681)
   EZA4990I Open (8673)
   EZA4989I Data delivered (8682)
6
EZA4993I Bytes in 73
7
TIME-07:22:19 AM. CPU-00:00:00 SERVICE-1199 SESSION-00:00:00 MARCH 4,1998
   EZA4989I Data delivered (8682)
   EZA4993I Bytes in 1
   EZA4989I Connection state changed (8681)
   EZA4990I Sending only (8675)
   EZA4989I Connection state changed (8681)
   EZA4990I Connection closing (8670)
   EZA4989I Connection state changed (8681)
   EZA4990I Nonexistent (8672)
   EZA4989I Connection state changed (8681)
   EZA4990I Sending only (8675)
   EZA4989I Connection state changed (8681)
   EZA4990I Connection closing (8670)
   EZA5001I Returning from recv_notices.
   EZA5047I returning from RSH_UTIL
   EZA5048I rsh complete
```

Figure 44. Example of an RSH Trace

Following are short descriptions of numbered items in the trace.

1 The RSH parameters that are entered at the console or received from batch

2 The command that is sent to the remote host

3 The client is attempting to resolve the name to an IP address by calling GetHostResol.

4 The port number, user ID, password, and command sent to the REXECD

5 Length of data being sent

6 Length of data passed back to the RSH client

7 Actual command output sent to the client

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.

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 *OS/390 IBM Communications Server: IP Configuration Reference* for more information about the options.

Notes:

1. REXECD trace output is included in the job output log.
2. If more than one trace option is selected, the options must be enclosed in parentheses.

Example of an REXECD Trace of a Client Using the SEND Command

Figure 45 on page 306 shows an example of an REXECD trace of a client using a SEND command. Short descriptions of numbered items in the trace follow the figure.

```
   MVS TCP/IP REXEC CS/390 V2R10
 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 45. 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.

2 The 80 indicates the job is currently active.

3 The 20 indicates the job is on the output queue.

Note: Refer to the *OS/390 IBM 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 46 on page 307 shows 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 46. 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.

3 The 80 indicates that job JOB00044 is currently active.

4 The 20 indicates that job JOB00044 is on the output queue.

Note: Refer to *OS/390 IBM 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 17. Diagnosing OS/390 UNIX REXEC, REXECD, and RSHD Problems

This chapter contains diagnosis information about the OS/390 UNIX remote execution protocol (REXEC), the remote execution protocol daemon (REXECD), and the remote shell daemon (RSHD).

Setting Up the inetd Configuration File

The `inetd` program is a generic listener program used by such servers as OS/390 UNIX TELNETD and OS/390 UNIX REXECD. Other servers such as OS/390 UNIX FTPD have their own listener program and do not used inetd.

The inetd.conf file is an example of the user's configuration file. It is stored in the /etc directory. Upon startup, the OS/390 UNIX TELNETD server, rshell, rlogin, and rexec are initiated. If it does not include OS/390 UNIX TCP/IP applications, add the information shown in Figure 47:

```
#======================================================================
# 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
```

Figure 47. Adding Applications to /etc/inetd.conf

To establish a relationship between the servers defined in the /etc/inetd.conf file and specific port numbers in the OS/390 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.

The traces for both the OS/390 UNIX REXECD server and the OS/390 UNIX RSHD server are enabled by options in the inetd configuration file (/etc/inetd.conf). See Figure 48.

```
#======================================================================
# service | socket | protocol | wait/ | user | server  | server program
# name    | type   |          | nowait|      | program |   arguments
#======================================================================
#
shell      stream    tcp       nowait OMVSKERN /usr/sbin/orshd rshd -d  1
exec       stream    tcp       nowait OMVSKERN /usr/sbin/orexecd rexecd -d  2
```

Figure 48. Setting Traces in /etc/inetd.conf

The traces are turned on for both servers by passing a -d argument to the server programs. **1** is the RSHD server and **2** is the REXECD server. All commands executed after the debug flags have been turned on in the inetd configuration file and the inetd server has reread the file will produce 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 *OS/390 IBM Communications Server: IP Configuration Reference* for more information about syslogd.

For more information about inetd, refer to *OS/390 UNIX System Services Planning* or *Accessing OpenEdition MVS from the Internet*.

Diagnosing OS/390 UNIX REXEC

The following kinds of information can help you diagnose an OS/390 UNIX REXEC problem:

- A message beginning with EZYRC
- A code
- An OS/390 UNIX REXEC debug trace
- A REXECD debug trace from the foreign host

Activating the OS/390 UNIX REXEC Debug Trace

To activate the OS/390 UNIX REXEC debug trace, specify the -d option.

OS/390 UNIX REXEC Trace Example and Explanation

Enter the following command with either a dotted decimal address or a host name.

```
orexec -d -l user21 -p xxx 9.67.113.61 ls -l
```

The following is an example of the trace output:

```
EZYRC01I  Calling function orexec with the following:
EZYRC02I  Host: 9.67.113.61, user user 21, cmd ls -l, port 512
EZYRC191  Data socket = 4, Control socket = 6.
```

EZYRC01I shows that the OS/390 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 OS/390 UNIX REXECD

The following kinds of information can help you diagnose OS/390 UNIX REXECD problem:

- A message beginning with EZYRD
- A code
- An OS/390 UNIX REXECD debug trace

- A trace from the OS/390 UNIX REXECD client

Activating the OS/390 UNIX REXECD Debug Trace

To activate the OS/390 UNIX REXECD debug trace, specify the -d option in the /etc/inetd.conf file.

OS/390 UNIX REXECD Trace Example and Explanation

These examples are in the file specified in syslogd.conf.

Note: Syslogd must be running to collect these traces and the file must have 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 the *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)*.

Diagnosing OS/390 UNIX RSHD

The following kinds of information can help you diagnose an OS/390 UNIX RSHD problem:

- A message beginning with EZY4S01I
- A code
- A OS/390 UNIX RSHD debug trace
- A trace from the RSH client

Activating the OS/390 UNIX RSHD Debug Trace

To activate the OS/390 UNIX RSHD debug trace, specify the -d option in the /etc/inetd.conf file.

OS/390 UNIX RSHD Trace Example and Explanation

These examples are from the file specified in syslogd.conf.

Note: Syslogd must be running to collect these traces and the file must exist and have 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 the *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)*.

Note: If you enable the L or d option, the entry in syslogd will include both user ID and password in clear text. Either do not specify these flags, or make sure your syslogd log files are properly protected

If the -U option is specified in /etc/inetd.conf, the OS/390 UNIX RSHD server will 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 OS/390 UNIX RSHD server may encounter an error so early in the processing of a command that the server has not established a proper EBCDIC-to-ASCII translation yet. In such a situation, the client end user may see garbage data returned to his or her terminal. A packet trace will reveal 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 OS/390 UNIX name resolution has not been configured correctly, so the OS/390 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 18. 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 both to the NDB server, which returns them to the NDB client.

Note: Refer to the *OS/390 IBM Communications Server: IP User's Guide* for more information about NDB usage.

The components of the Network Database System are shown in Figure 49.

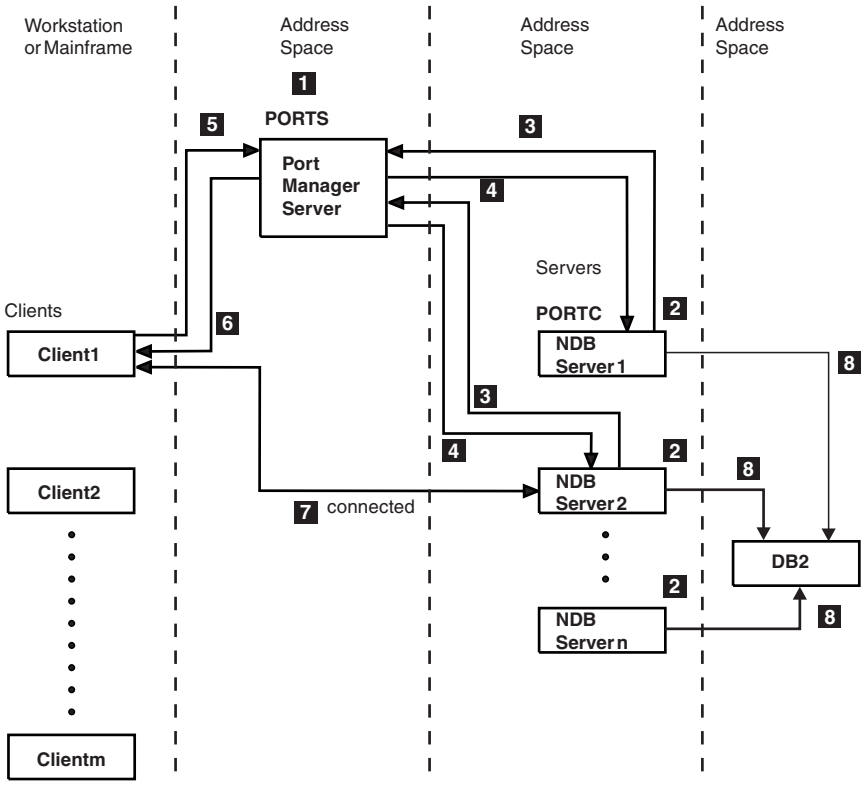


Figure 49. Components of the network database system

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, through the NDB port client (PORTC), issues a request to PORTS for a program number.

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 one to 20 NDB servers. Each PORTC address space can access a different DB2 subsystem. A total of 100 NDB servers, across from five to 100 PORTC address spaces, is supported.

Note: Refer to the *OS/390 IBM 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

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 one to 20 NDB servers within

this address space. Refer to the *OS/390 IBM 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 *OS/390 IBM Communications Server: IP Configuration Reference* for more information.

Diagnosing NDB Problems

Most of the information you need can be gathered through return codes.

Follow these steps to gather information you need:

1. Check the return code and error message. Refer to the *OS/390 IBM Communications Server: IP User's Guide* and *OS/390 IBM 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 *OS/390 IBM Communications Server: IP User's Guide* . 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 *OS/390 IBM Communications Server: IP Configuration Reference* for information about binding the DBRM DBUTIL2 (NDBSETUP) to created 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 will execute 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 V2R3, or higher, and 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 done.

NDB Trace Examples and Explanations

Figure 50 on page 317 is 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 51 on page 319.

Notes:

1. 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.
2. NDB trace output is included in the job log output from the started NDB procedure.

```
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 50. NDB Port Manager Trace with Two NDB Servers Started and One Client Invoked

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 51 on page 319 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 50 on page 317.

Notes:

1. 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.
2. NDB trace output is included in the job log output from the started NDB procedure.

```
 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 51. 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 51. 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 51. 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 51. 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 51. 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 51. 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 51. NDB Port Client Trace with Two NDB Servers Started and One Client Invoked (Part 7 of 7)

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 will be 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 1 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 has been started up and show the values of the parameters passed to it.

13 The following 10 trace messages indicate that NDB server 2 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 2 has been started up and show 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 1 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 1.

20 The following three trace messages indicate that the NDB server is ready to start processing the user's request. The host userid and password supplied by the user with the NDB client have passed the security check and the NDBC control block has been 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 will be 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 1 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 19. Diagnosing X Window System and OSF/Motif Problems

An environment variable, XWTRACE, controls the generation of traces of the socket level communication between Xlib and the X Window System Server.

- 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.

Note: There are no special post-install activities for GDDMXD in CS for OS/390. (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 52 shows a typical stream of socket level activity that is generated when an X application running on OS/390 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 52. 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 53 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 53. 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 53. 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 both
- If exiting the invoked function, the trace statement indicates that the function is returning

Chapter 20. Diagnosing Simple Network Management Protocol (SNMP) Problems

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).

Overview

This section provides explanations for SNMP-related concepts and terms.

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 CS for OS/390, 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 runs as part of OMPROUTE, not as a separate application. The subagent program for SLA-related MIB variables runs as a separate application. For a list of all the MIB objects supported by the agent and subagents shipped as part of CS for OS/390, refer to the *OS/390 IBM Communications Server: IP User's Guide*.

In addition, user-written subagent programs can also exist. All subagent programs, whether provided by CS for OS/390 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. CS for OS/390 supports two management applications, the OS/390 UNIX SNMP command (**osnmp**) and the NetView SNMP command. The **osnmp** command is a management application used from the OS/390 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. CS for OS/390 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. CS for OS/390 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
- SLA subagent that manages the sla MIB

These subagents communicate with the SNMP agent using the DPI 2.0 protocol.

Trap Forwarder Daemon

The Trap Forwarder daemon on CS for OS/390, 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.

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 *OS/390 IBM Communications Server: IP Configuration Reference*. In the text that follows, upper case 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 *OS/390 IBM 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.

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. May 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 CS for OS/390, 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 32 on page 336.

Table 32. 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

ATM DEVICE and LINK statements
Used when SNMP is to monitor ATM network information

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.

SLA Subagent

The SLA subagent is controlled by start options specified when the subagent is started. The following statements 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.

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.

MESSAGECASE
Controls whether messages sent to the MVS operator console are displayed in mixed case or uppercase.

TCPIPJOBNAME
Determines the TCP/IP instance in which SNMP attempts to establish its relationship through the SETIBMOPT socket call. For more information about TCPIPJOBNAME, see "Appendix B. Search Paths" on page 521 or refer to the *OS/390 IBM 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:

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 *OS/390 IBM Communications Server: IP Configuration Guide* and *OS/390 IBM Communications Server: IP Configuration Reference*.

Abends

An abend during SNMP processing should result in messages and error-related information sent to the system console. A dump of the error will be 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 *OS/390 MVS Diagnosis: Procedures* or "Chapter 3. Diagnosing Abends, Loops, and Hangs" on page 21, for information about debugging dumps produced during SNMP processing.

SNMP Connection Problems

This section describes how to diagnosis and correct SNMP connection problems.

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:

1. 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).
2. Is the SNMPIUCV subtask started?
   - If not, start the subtask by issuing the command:

     ```
     START TASK=SNMPIUCV
     ```

     from a NetView operator console.
3. 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 2 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 *OS/390 MVS Diagnosis: Tools and Service Aids* manual for your release of OS/390. Obtaining SNMP traces is discussed in "SNMP Traces" on page 353.

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:

1. Did any socket-related errors occur?

   Check the SNMP query engine job output for socket(), bind(), accept(), or other socket error messages.

2. 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?
   - 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 *OS/390 MVS Diagnosis: Tools and Service Aids* manual for your release of OS/390. Obtaining SNMP traces is discussed in "SNMP Traces" on page 353.

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 *OS/390 IBM 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: Check the following for problems connecting the SNMP client or agent address space to the TCP/IP address space:

1. 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 OS/390 UNIX MVS.
2. 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 will occur when one or more of the ports needed by the SNMP agent is already in use. Refer to the *OS/390 IBM Communications Server: IP Configuration Guide* for more information about syslogd.
3. 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 OMPROUTE address space (for OMPROUTE subagent problems)
- Dump of SLA subagent address space (for SLA subagent problems)
- Dump of TCP/IP address space
- The syslogd traces from the agent (using trace level 255). Refer to the *OS/390 IBM Communications Server: IP Configuration Guide* for more information about reading the syslogd.

Information on obtaining a dump can be found in the *OS/390 MVS Diagnosis: Tools and Service Aids* manual for your release of MVS. Obtaining SNMP traces is discussed "SNMP Traces" on page 353.

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 jobname of the TCP/IP stack that it wants to associate with.

Analysis: Check the following for problems connecting the SNMP agent to the correct TCP/IP stack.

1. Message EZZ6205I indicates that when _iptcpn() was called, it did not return the correct TCPIPjobname for that agent. Check _iptcpn()'s search path, as indicated in "Appendix B. Search Paths" on page 521.
2. Message EZZ6272I indicates that the setibmopt call failed. This means that _iptcpn() returned a name that OS/390 UNIX did not recognize as a PFS. Check the BPXPRMxx member (in SYS1.PARMLIB) used to configure OS/390 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
- 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)
- 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).

Analysis: Check the following for problems connecting an SNMP subagent program to the SNMP agent:

1. Is the subagent in question the TCP/IP subagent? If so,
   - Is the SACONFIG statement configured correctly?
   - Is SACONFIG disabled?
2. 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>?
3. 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?

4. 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.
5. Is the subagent using the correct IP address to send the connection request to the SNMP agent? The subagent uses the primary interface IP address of this stack when sending the connection request to the SNMP agent. The primary interface IPS 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 primary interface address of the stack. This IP address is the one that will be used by the SNMP agent, along with the community name to verify the subagents authority to connect to the SNMP agent.
6. Is the port number correct?
7. Is the community name (or password) correct?

   Note: Note that 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*.
8. 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.
9. 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.

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. 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 start-up. It can also be done after TCP/IP has been started by using the `VARY TCPIP` command, which is documented in *OS/390 IBM Communications Server: IP Configuration Reference*.
- 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 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.

The following is a list of things to look for in the SNMP agent trace:

1. 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 *hlq*.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 SLA subagent, specify the community name by way of the -c parameter.
   - 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.
2. 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 will display the port on which the agent is accepting incoming UDP requests.
     – To display the dpiPath name, issue an osnmp get request for dpiPathNameForUnixStream.
3. One of the following- incoming subagent connection
   - 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*.

4. 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 subagent problems)
- Dump of OMPROUTE address space (for OMPROUTE 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 *OS/390 MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP agent traces is discussed in "Starting SNMP Agent Traces" on page 353.

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

  Note: A common mistake is forgetting to add a dot-zero (.0) to an object that is not part of a table. For example, a GET for ifNumber returns with "no such object" while performing a GET on ifNumber 0. If unsure how to proceed, try a GETNEXT on ifNumber. This will return the point-zero (.0) version of the OID along with its value.
- 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 *OS/390 IBM 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) may also be needed.
- Include all the configuration files described earlier under "Definitions" on page 335.

Analysis: Check the following 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 may have more than one occurrence of the variable value. To get the value of the variable, you will 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?
   - Check the SNMP agent job output for messages indicating a problem in create_DPI_port.
   - If the DPI port was not successful, no SNMP subagents will be able to register MIB variables. The SNMP agent will have no knowledge of these unregistered variables and will report them as "noSuchName" for SNMPv1 requests or "noSuchObject" for SNMPv2 and SNMPv3 requests.
4. Has the subagent successfully connected to the SNMP agent?
   - For subagents shipped as part of CS for OS/390, check the MVS operator console for a message indicating that the subagent has completed its initialization.
   - Issue an osnmp walk command on the SNMP agent subagent status table. For example, the following commands display the subagents that are connected to the CS for OS/390 SNMP agent and the status of their connections:

     ```
     osnmp -v walk saDescription
     ```

     or

     ```
     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?
   - 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.

- Alternatively, SNMP agent configuration can be determined from SNMP agent traces if they were set to level 255 at agent initialization.
- 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 4 manager traces.
- If the manager is the NetView SNMP command, obtain the SNMP query engine level 2 output.

The SNMP query engine level 2 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 CS for OS/390 subagent)

The SNMP agent level 15 trace will show PDUs between the manager and agent, as well as between the agent and any existing subagents. Things to look for in the trace:

1. Is the ASN.1 number received from the manager in the SNMP GetRequest-PDU correct?
2. Has a DPI packet registering the requested variable been received?
   - If not, if you know which subagent program owns the variable, check the subagent program for errors.
   - If the DPI register has been received, make a note of the FD number for further trace information.
3. Were any errors reported for this FD number after the DPI register request was received?
4. 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 TCP/IP address space (for CS for OS/390 subagent variables)
- Dump of OMPROUTE address space (for OMPROUTE 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 *OS/390 MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP traces is discussed later in "SNMP Traces" on page 353.

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: Check the following:

1. Is the variable contained in the *hlq*.MIBDESC.DATA data set or MIBS.DATA file?
   - The SNMP query engine will use 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.
   - **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 *OS/390 IBM Communications Server: IP User's Guide* 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 4 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 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 *OS/390 IBM Communications Server: IP Configuration Guide*.
- SNMP query engine level 4 trace output or osnmp command trace level 4.

- 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.

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 Software Support Center be contacted before this documentation is obtained:
- Dump of SNMP subagent address space
- Dump of SNMP agent address space
- Dump of SNMP query engine address space
- Dump of OMPROUTE address space (for OMPROUTE subagent problems)
- Dump of SLA subagent address space (for SLA subagent problems)

Information on obtaining a dump can be found in *OS/390 MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP traces is discussed in "SNMP Traces" on page 353.

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: Check the following:
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 SLA subagent? If so, is it being cached? The SLA subagent caches MIB objects for 5 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 3 will show what was returned to the client.
- For the TCP/IP subagent, ITRACE level 4 trace output to show 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 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 *OS/390 IBM Communications Server: IP Configuration Guide*.

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 CS for OS/390 subagent variables)
- 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 *OS/390 MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP traces is discussed in "SNMP Traces" on page 353.

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

Check the following 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, may 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. If the manager is the osnmp command, was the command destination specified using the `-h` parameter? Did the `-h` value have an entry in the OSNMP.CONF file? If so, and the destination agent only supports SNMPv1, the request will be

discarded. Try reissuing the command using an IP address as the value of the -h parameter. If an entry with the value specified on the -h parameter does not exist in the OSNMP.CONF file, an SNMPv1 request will be sent. An SNMPv2 or SNMPv3 agent can process SNMPv1 requests, but an SNMPv1 agent cannot process SNMPv2 or SNMPv3 requests.

If the problem still occurs after checking the preceding items and making any needed changes, obtain SNMP agent level 7 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?

Note: 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 *OS/390 IBM 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 OMPROUTE address space (for OMPROUTE subagent problems)
- Dump of SLA subagent address space (for SLA subagent problems)

Information on obtaining a dump can be found in *OS/390 MVS Diagnosis: Tools and Service Aids*. Obtaining SNMP traces is discussed in "SNMP Traces" on page 353.

Report Received from SNMP Agent

With SNMPv3, certain error conditions detected on a request will be 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 will be received.

Figure 56 on page 357 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 OS/390 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.

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 2 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 *OS/390 MVS Diagnosis: Tools and Service Aids* manual for your release of MVS. Obtaining SNMP traces is discussed “SNMP Traces” on page 353

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 3
- Traces from the sending agent (the originator of the trap)
- Trace from the receiving client (the target of the forwarded trap)

Analysis

Check the following:

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.
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

Check the following:

1. What is the version of the SNMP trap?

   In the case of SNMPv1 traps the address from which the trap is originated is encoded within the trap packet. A manager that needs the originating address should look into the SNMP packet to get the address. 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 was 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 *OS/390 IBM Communications Server: IP User's Guide*.
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 an specify a trace level of 0 to 4. A trace level of 0 provides no tracing, while a level 4 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 1 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 *OS/390 IBM 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 jobname the agent is running; this is the jobname to specify on the MODIFY command.

Note: While most agent tracing can be modified dynamically, tracing of SNMPv3 requests in and responses out can be enabled only at agent initialization.

For example, assume the procname is OSNMPD and you want to change the trace level to 3 (tracing SNMP requests and responses). You would enter

```
MODIFY OSNMPD,trace,level=3
```

For more information on using the MVS MODIFY command, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.

Starting TCP/IP Subagent Traces

To start the TCP/IP subagent traces, code the ITRACE statement in the PROFILE.TCPIP file or in a file to be used by the *OS/390 IBM 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 *OS/390 IBM Communications Server: IP Configuration Guide*.

To stop TCP/IP subagent traces, code the ITRACE statement in the PROFILE.TCPIP or the obey file to be used by the VARY TCPIP command:

```
ITRACE OFF SUBAGENT
```

For more information on the VARY command, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.

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 OS/390 Shell" on page 438 and "Starting OMPROUTE Tracing and Debugging Using the MODIFY Command" on page 439.

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 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 *OS/390 IBM 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 jobname the trapfwd is running; this is the jobname 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 *OS/390 IBM Communications Server: IP Configuration Reference*.

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 54 on page 357 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 54. 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 55 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 55. 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 56 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 will receive a report indicating usmStatsWrongDigests.

```
IDSTMVS.S@AU1104.SOURCE.S@AGV123(1624): rc=-65 (SNMP_RC_USM_WRONG_DIGEST)
   from snmp_process_message()
```

Figure 56. SNMP Messages and Agent Trace for Nonmatching Key

Figure 57 on page 358 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 OS/390 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 57. SNMP Messages and Agent Trace When Data Not in Defined View

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 58 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
Apr  4 16:28:34 MVS097 snmpagent[67108869]: IDSTMVS.SO064350.SOURCE.S@AGV123(1623): rc=-14 (SNMP_RC_NOT_AUTHENTICATED) from
```

Figure 58. SNMP Subagent Trace

SNMP Query Engine Trace

This section discusses the output produced by the SNMP query engine trace.

Figure 59 on page 360 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 will show this type of trace entry. On the other hand, lines 10 through 17 have a “trc lvl” of 2. This means that trace level 2 or higher will produce this trace information. 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 59. SNMP Query Engine Traces (Part 1 of 10)

```
    54  EZA6364I packet type:        TRAP
    55  EZA6394I  filter id:         1001
    56  EZA6395I  agent address:     9.67.114.37
    57  EZA6399I  generic trap:      4     ( 0X4 )
    58  EZA6400I  specific trap:     0     ( 0X0 )
    59  EZA6401I  time stamp:        9600
    60  EZA6402I  enterprise len:    22
    61  EZA6403I  enterprise:        1.3.6.1.4.1.2.2.1.2.4
4   62  EZA6292I Received following NVquery packet:
    63  EZA6305I dumping packet of 15 bytes:
    64  00 0d 01 01 01 02 07 00 00 03 ea 00 00 03 e9
2   65  EZA6359I major version:  1
    66  EZA6360I minor version:  1
    67  EZA6361I release:        1
    68  EZA6363I native set:     EBCDIC
    69  EZA6364I packet type:    TRAP UN-REQUEST
    70  EZA6367I  sequence id:       1002
    71  EZA6394I  filter id:         1001
2   72  EZA6359I major version:  1
    73  EZA6360I minor version:  1
    74  EZA6361I release:        1
    75  EZA6363I native set:     EBCDIC
    76  EZA6364I packet type:    RESPONSE
    77  EZA6367I  sequence id:       1002
    78  EZA6388I  major error:       0
    79  EZA6389I  minor error:       0
    80  EZA6390I  error index:       0
    81  EZA6391I  error text len:    9
    82  EZA6392I  error text:        no error
4   83  EZA6301I Received following SNMP_trap packet:
    84  EZA6305I dumping packet of 43 bytes:
    85           30 29 02 01 00 04 04 4d 56 53 4c a4 1e 06 0a 2b
    86           06 01 04 01 02 02 01 02 04 40 04 09 43 72 25 02
    87           01 04 02 01 00 43 02 38 40 30 00
3   88  EZA6424I Decoded SNMP PDU :
    89  {
    90     version version-1,
    91     community '4d56534c'H,
    92     data {
    93        trap {
    94           enterprise 1.3.6.1.4.1.2.2.1.2.4,
    95           agent-addr {
    96              internet '09437225'H
    97           },
    98           generic-trap authenticationFailure,
    99           specific-trap 0,
   100           time-stamp 14400,
```

Figure 59. 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 59. 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 59. SNMP Query Engine Traces (Part 4 of 10)

```
      210  EZA6377I  backoff exponent:  1
   211  EZA6380I   name length:      22
   212  EZA6381I   name:            1.3.6.1.2.1.2.2.1.2.0
3  213  EZA6424I Decoded SNMP PDU :
   214  {
   215     version version-1,
   216     community '534e4d50'H,
   217     data {
   218        get-next-request {
   219           request-id 2,
   220           error-status noError,
   221           error-index 0,
   222           variable-bindings {
   223              {
   224                 name 1.3.6.1.2.1.2.2.1.2.0,
   225                 value {
   226                    simple {
   227                       empty {}
   228                    }
   229                 }
   230              }
   231           }
   232        }
   233     }
   234  }
   235  EZA6308I sending SNMP request to 9.67.114.37
4  236  EZA6295I Received following SNMP_response packet:
   237  EZA6305I dumping packet of 47 bytes:
   238           30 2d 02 01 00 04 04 53 4e 4d 50 a2 22 02 01 02
   239           02 01 00 02 01 00 30 17 30 15 06 0a 2b 06 01 02
   240           01 02 02 01 02 01 04 07 49 42 4d 20 4c 43 53
3  241  EZA6424I Decoded SNMP PDU :
   242  {
   243     version version-1,
   244     community '534e4d50'H,
   245     data {
   246        get-response {
   247           request-id 2,
   248           error-status noError,
   249           error-index 0,
   250           variable-bindings {
   251              {
   252                 name 1.3.6.1.2.1.2.2.1.2.1,
   253                 value {
   254                    simple {
   255                       string '49424d204c4353'H
   256                    }
   257                 }
   258              }
   259           }
   260        }
   261     }
   262  }
2  263  EZA6359I major version:  1
   264  EZA6360I minor version:  1
   265  EZA6361I release:        1
```

Figure 59. 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 59. 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 59. SNMP Query Engine Traces (Part 7 of 10)

```
   347                   }
   348                }
   349             }
   350          }
   351       }
   352    }
   353 }
2  354 EZA6359I major version:  1
   355 EZA6360I minor version:  1
   356 EZA6361I release:        1
   357 EZA6363I native set:     EBCDIC
   358 EZA6364I packet type:    RESPONSE
   359 EZA6367I  sequence id:       1005
   360 EZA6388I  major error:       0
   361 EZA6389I  minor error:       0
   362 EZA6390I  error index:       0
   363 EZA6391I  error text len:    9
   364 EZA6392I  error text:        no error
   365 EZA6380I   name length:      22
   366 EZA6381I   name:             1.3.6.1.4.1.2.2.1.4.1
   367 EZA6382I   value type:       number
   368 EZA6384I   value length:     4
   369 EZA6386I   value:            12345
4  370 EZA6292I Received following NVquery packet:
   371 EZA6305I dumping packet of 29 bytes:
   372          00 1b 01 01 01 02 08 00 00 03 ee 00 10 f1 4b f3
   373          4b f6 4b f1 4b f2 4b f1 4b f1 4b f1 00
2  374 EZA6359I major version:  1
   375 EZA6360I minor version:  1
   376 EZA6361I release:        1
   377 EZA6363I native set:     EBCDIC
   378 EZA6364I packet type:    VAR_NAME
   379 EZA6367I  sequence id:       1006
   380 EZA6405I  object id len:     16
   381 EZA6406I  object id:         1.3.6.1.2.1.1.1
2  382 EZA6359I major version:  1
   383 EZA6360I minor version:  1
   384 EZA6361I release:        1
   385 EZA6363I native set:     EBCDIC
   386 EZA6364I packet type:    RESPONSE
   387 EZA6367I  sequence id:       1006
   388 EZA6388I  major error:       0
   389 EZA6389I  minor error:       0
   390 EZA6390I  error index:       0
   391 EZA6391I  error text len:    9
   392 EZA6392I  error text:        no error
   393 EZA6380I   name length:      16
   394 EZA6381I   name:             1.3.6.1.2.1.1.1
   395 EZA6382I   value type:       display string
   396 EZA6384I   value length:     9
   397 EZA6385I   value:            sysDescr
4  398 EZA6292I Received following NVquery packet:
   399 EZA6305I dumping packet of 23 bytes:
   400          00 15 01 01 01 02 0a 00 00 03 ef 00 05 d4 e5 e2
```

Figure 59. 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 59. 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 59. SNMP Query Engine Traces (Part 10 of 10)

The following is an explanation of the traces in Figure 59 on page 360.

- 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 will allow. (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 will be 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 60 on page 372 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 60. 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 60. 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 60. 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 60. 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 60. 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 61 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 61. TRAPFWD Trace

Chapter 21. Diagnosing Policy Agent Problems

Overview

The OS/390 UNIX Policy Agent provides administrative control for service policies. The following terms must be defined to understand service policies:

Quality of Service (QoS)
The overall service that a user or application receives from a network, in terms of throughput, delay, etc.

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.

The Policy Agent reads service policies defined in a local file, or read by way of the Lightweight Directory Access Protocol (LDAP) from an LDAP server. These service policies are then installed in one or more TCP/IP stacks. The 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. The 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. Refer to the *OS/390 IBM Communications Server: IP Configuration Guide* for more information on configuring and starting the Policy Agent, as well as defining service policies.

Service Policy Scopes

Service policies can be defined with different scopes. The supported scopes are:

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. This type of policy is in support of the TRM daemon.

The TCP/IP stack maps TCP, UDP, and RAW traffic to service policies based on the selection criteria defined in the policy. Search criteria can include, but is 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. For UDP and RAW, however, the mappings are cached such that subsequent packets sent to the same destination will 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 22. Diagnosing RSVP Agent Problems" on page 395 for more information on the operation of RSVP. TR policy only applies to TCP, and is also mapped at connection time.

You can see the effect of defined service policies in either of the following ways.

- The SLA Subagent can be used to display service policy and mapped application information, as well as to manage and display SLA performance monitoring.
- Using the UNIX pasearch and TSO NETSTAT commands.
  - The pasearch command shows defined service policies.
  - The NETSTAT SLAP or netstat -j command shows performance metrics for active policy rules.
  - The NETSTAT ALL or onetstat -A command has additional information for each active connection that shows the service policy rule name if the connection maps to a service policy.

Refer to the *OS/390 IBM Communications Server: IP User's Guide* for more information on the NETSTAT/onetstat commands, the pasearch command, and the SLA Subagent.

Gathering Diagnostic Information

The 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. By default, only error and warning messages are written. To gather more diagnostic information, you can specify a LogLevel value and/or start the Policy Agent with the -d startup option. When the -d startup option is used, the maximum amount of information is logged. Use the debug levels as follows:

- Use debug level 1 for most debugging.
- Use debug level 2 to verify Pagent processing of LDAP objects, or if a problem is suspected in how LDAP objects are defined.

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 may not be sufficient to capture all of the information needed. In this case, use the environment variable to increase the number and/or size of the log files. Refer to the *OS/390 IBM 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.

Certain other information may be useful in diagnosing Policy Agent problems:

- Output from the pasearch command
- Output from the NETSTAT SLAP or onetstat -j commands

- Output from the NETSTAT ALL or onetstat -A commands for active connections mapped to service policies
- SNMP output from walks of the SLA subagent MIB tables
- TCP/IP CTRACE output, using the INTERNET and IOCTL CTRACE options
- RSVP Agent log output if RSVP scoped policies are defined

Diagnosing Policy Agent Problems

Problems with the Policy Agent will generally fall into one of the following categories:

- Initialization
- Policy definition
- LDAP

Initialization Problems

If the Policy Agent does not complete initialization, run it with the -d startup option and check the log file for error conditions. If the Policy Agent fails to initialize, message EZZ8434I is issued to the console. Check the log file for the specific error encountered. Common problems could include:

- Policy Agent started from a userid without superuser authority.

  The Policy Agent must be started from a superuser. The symptoms for this are the 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.

  The Policy Agent must be authorized to a security product profile. Refer to the *OS/390 IBM Communications Server: IP Configuration Guide* for details on setting up the proper authorization. The symptoms for this are the EZZ8434I message, along with messages in the log file indicating that the user is not authorized to start PAGENT and showing an exit code value of 18.

- Unable to read configuration file.

  The symptoms for this are the 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.

  – Is the correct configuration file specified? Refer to *OS/390 IBM Communications Server: IP Configuration Guide* for the search order used to locate the main configuration file.
  – Does the file exist?
  – Are the permission bits correctly set for an HFS file?
  – The main configuration file may also include other configuration files for specific TCP/IP stacks. Check these files as well if necessary.

- Timing windows when using the TcpImage FLUSH parameter.

  When FLUSH is specified on the TcpImage configuration statement, the Policy Agent deletes all policies from its own internal database and from the TCP/IP stack whenever it needs to reprocess policy definitions. This reprocessing occurs at the following times:

  – At each refresh interval specified on the TcpImage statement, if the configuration file is an MVS data set.
  – At each refresh interval specified on the TcpImage statement, if the configuration file is an HFS file that has been changed since the last refresh interval.

Depending on the number of policy definitions and other factors, there may be a delay between the time the previous policies are deleted and the new policies are re-installed. During this time, no policies are installed in the stack. If this is unacceptable, specify the NOFLUSH parameter on the TcpImage statement. However, be aware that the entire set of policies is never deleted when NOFLUSH is specified. If you use NOFLUSH and policy definitions are frequently changing, unused policies may accumulate in the stack. In this situation, consider periodically flushing old policies by starting the Policy Agent with a configuration file that only contains a TcpImage statement with FLUSH specified.

- 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 non-overlapping 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. Since the minimum time resolution used by the Policy Agent is one minute, there will be a period of one minute when neither policy is active.

Policy Definition Problems

If you don't see the expected results when defining policies, check the following. Use the pasearch command to display policies (active or inactive) known by the Policy Agent. This command can be used to check if policies are active or inactive, and whether or not they contain the specifications that were expected.

Notes:

1. Misspelled attributes are treated by the Policy Agent as unknown attributes, and are not flagged in error, so it's the same as if the misspelled attribute were not specified at all.
2. Policy rules with complex conditions (using CNF/DNF logic) are processed by the 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).
3. 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.

Another way to check if policies are being installed and used correctly is to use the NETSTAT commands. Use the NETSTAT SLAP or onetstat -j command to display active policy statistics for policies installed in the stack, as opposed to the policies in the Policy Agent. The NETSTAT ALL or onetstat -A command shows which policy rule (if any) is mapped to active connections.

Some of the problems you may encounter include the following.

- Problems with converting version 1 policies to version 2 policies.

  There are some semantic differences between version 1 and version 2 policy definitions. Only the version 2 semantics are currently supported, so version 1 policy semantics are converted to version 2 semantics when the policies are processed by the Policy Agent. Be aware of the following processing:

  – 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. 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.

  - Similarly to the above, when Direction Both is specified in a version 1 policy, two policies are installed in the stack - one for the outbound direction with source/destination attributes intact, and one for the inbound direction with the attributes swapped. When converting version 1 rules with Direction Both specified, be sure to 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.
  - 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. 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.

- Policy groups or rules are discarded when defined on an LDAP server.

  Policy groups and policy rules defined on an LDAP server may 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. Be sure to specify the correct reference Distinguished Names on LDAP objects that reference other objects.

- Policies with complex conditions (using CNF or DNF) are not mapping correctly.

  Complex policy conditions using CNF or DNF can be difficult to properly configure. Since some conditions are logically ANDed, the result may be invalid. For example, two or more distinct interfaces cannot be ANDed and still be true. Or two non-overlapping port ranges also cannot be ANDed. The Policy Agent tries to detect these types of errors and discard the policy rules with an error message, but there are cases that can't be detected (for example logical ANDs between CNF/DNF levels, or when negated conditions are used). In these cases, a policy rule may well be installed that can never be true. Similar problems could occur when ORing conditions. For example, a very broad condition may map much more traffic than was intended, simply because it's 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 the 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 will help to understand how the working set was derived.

- Wrong policy being mapped to traffic.

  There may be times when 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.

- Policies not installed in the TCP/IP stack. The symptom for this is an unexpected or missing set of policies displayed by the NETSTAT SLAP or onetstat -j command.

  Is the stack in question configured via a TcpImage statement in the Policy Agent configuration file? Check that day of week and/or time of day specifications are correct. Verify the specified time zone. For time zones other than local time, the specified time periods may not be currently active. Has the stack been started or restarted after the Policy Agent was started? If so, check that the temporary file used by the stack to inform the Policy Agent of restarts has not been deleted. Refer to the *OS/390 IBM Communications Server: IP Configuration Guide* for more details.

- Policies not mapping to the expected traffic. The symptom for this is a blank policy rule name displayed for an active connection using the NETSTAT ALL or onetstat -A command.

  If you think data traffic should be mapped to certain policies, but isn't, then check to make sure you've 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 will be mapped as inbound data, while the data flow will be 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 isn't 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.

- Policies defined in an MVS data set are not being installed

  When an MVS data set is used to define policies, be careful that sequence numbers are not part of the file, as these will cause parsing errors. In ISPF, use the NUMBER OFF and/or UNNUM commands to remove sequence numbers.

LDAP Problems

Service 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. The Policy Agent acts as an LDAP client to communicate with and retrieve policies from an LDAP server. The Policy Agent uses an LDAP DLL to perform its LDAP client functions.

If you're having problems receiving policies from an LDAP server, run the Policy Agent with the -d startup option, and check the following:

- Unable to connect to the LDAP server.

  The symptom for this is message EZZ8440I issued to the console. If the 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 (1 minute, then at 5 minute intervals, with the maximum time between connect attempts being 30 minutes). Note that if a backup LDAP server is configured, the EZZ8440I message is only issued if neither the primary or backup server can be connected.

  If you receive message EZZ8440I, 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 userid and password, and SSL parameters. Also check that the specified protocol version matches the protocol version being run on the server.

- No objects, or incorrect objects, retrieved from the LDAP server.

  The symptoms for this are missing or incorrect policies displayed by the pasearch command, or the NETSTAT SLAP or onetstat -j commands. 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 sample file /usr/lpp/tcpip/samples/pagent_oc.conf for the version 2 schema object classes.

- Wrong set of objects retrieved from the LDAP server. The symptoms for this are missing or incorrect policies displayed by the pasearch command, or the NETSTAT SLAP or onetstat -j commands.

  Check that the search and selection criteria specified on the ReadFromDirectory statement in the configuration file are correct. For version 1 policies, verify the correct Base and SelectedTag attributes are used. For version 2 policies, check the SearchPolicyBaseDN, SearchPolicyGroupKeyword, and SearchPolicyRuleKeyword attributes.

- LDAP DLL not found

  The Policy Agent must have access to the LDAP DLL at run time. The symptom for this is that the Policy Agent terminates unexpectedly with a CEEDUMP. The reason for termination in the CEEDUMP indicates that the LDAP DLL (GLDCLDAP) was not found. The Policy Agent accesses the LDAP DLL using the LIBPATH environment variable. 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.

- Version 1 policies not shared among multiple TCP/IP stacks

  The 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 will scope 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 multi-value 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.

Example Log File

Figure 62 on page 386 demonstrates some of the Policy Agent processing. This log file was created using the -d startup option. The following configuration files were used to produce this output:

- Main configuration file:

```
TcpImage                                TCPCS FLUSH 120
TcpImage                                TCPCS2 /u/user10/pagent.conf

LogLevel                                511

ReadFromDirectory
{
  LDAP_Server                           9.37.83.93
```

```
  LDAP_Port                          9000
  LDAP_DistinguishedName             cn=root, o=IBM, c=US
  LDAP_Password                      secret
  LDAP_SchemaVersion                 2
  LDAP_ProtocolVersion               2
  SearchPolicyBaseDN                 o=ibm, c=us
}

PolicyPerfMonitorForSDR              Enable
{
   SamplingInterval                  60
   LossRatioAndWeightFr              20 25
   TimeoutRatioAndWeightFr           50 50
   LossMaxWeightFr                   95
   TimeoutMaxWeightFr                100
}

SetSubnetPrioTosMask
{
  SubnetAddr                         9.37.65.139
  SubnetTosMask                      11100000
  PriorityTosMapping                 1 11100000
  PriorityTosMapping                 1 11000000
  PriorityTosMapping                 2 10100000
  PriorityTosMapping                 2 10000000
  PriorityTosMapping                 3 01100000
  PriorityTosMapping                 3 01000000
  PriorityTosMapping                 4 00100000
  PriorityTosMapping                 4 00000000
}

PolicyRule                           DiffServRule
{
   ProtocolNumberRange               6:42
   OutboundInterface                 9.37.65.139
   SourceAddressRange                9.67.100.0 9.67.100.255
   SourcePortRange                   20:21
   DayOfWeekMask                     0111110
   TimeOfDayRange                    08:00-12:00
   PolicyActionReference             DiffServAction1
   PolicyRulePriority                8
   ApplicationName                   FTP*
   ApplicationData                   /u/user10
   DayOfMonthMask                    11111111111111100000000000000000
}

PolicyAction                         DiffServAction1
{
   PolicyScope                       DataTraffic
   MaxRate                           10000
   MinRate                           2000
   OutboundInterface                 9.67.116.90
   OutboundInterface                 9.67.116.91
   OutboundInterface                 9.67.116.92
   OutboundInterface                 9.67.116.93
   OutboundInterface                 9.67.116.94
   OutboundInterface                 9.67.116.95
   OutboundInterface                 9.67.116.96
   OutboundInterface                 9.67.116.97
   OutboundInterface                 9.67.116.98
   OutboundInterface                 9.67.116.99
   OutboundInterface                 9.67.117.90
   OutboundInterface                 9.67.117.91
   OutboundInterface                 9.67.117.92
   OutboundInterface                 0.0.0.0
   OutgoingTOS                       10000000
   MaxDelay                          50
```

```
   MaxConnections                          100
   DiffServInProfileRate                   128
   DiffServInProfileTokenBucket            32
   DiffServOutProfileTransmittedTOSByte    00100000
   DiffServExcessTrafficTreatment          BestEffort
}

PolicyRule                                 TRRule
{
   DestinationPortRange                    10000:10100
   DayOfWeekMask                           0111111
   PolicyActionReference                   TRAction
}

PolicyAction                               TRAction
{
   PolicyScope                             TR
   TypeActions                             Statistics Log
   TotalConnections                        500
   Percentage                              80
   TimeInterval                            120
   LoggingLevel                            7
}

PolicyRule                                 BothRule
{
   DestinationAddressRange                 9.67.116.0 9.67.116.255
   SourcePortRange                         8000:9000
   ProtocolNumberRange                     6
   PolicyActionReference                   DiffServAction2
   PolicyActionReference                   RSVPAction
}

PolicyAction                               DiffServAction2
{
   PolicyScope                             Both
   MaxRate                                 1000000
   MinRate                                 10000
   OutgoingTOS                             11000000
}

PolicyAction                               RSVPAction
{
   PolicyScope                             RSVP
   Permission                              Allowed
   OutgoingTOS                             10100000
   FlowServiceType                         ControlledLoad
   MaxRatePerFlow                          440   # 55000 bytes/second
   MaxTokenBucketPerFlow                   48    # 6000 bytes
   MaxFlows                                10
}
```

- Policy Agent Processing Log:

```
 (01)
03/22 16:33:56 INFO   :.log_init[31]: Log File Size = 300, Number of Log Files = 3
03/22 16:33:56 INFO   :.main[31]: OS/390 Service Policy Agent
 (02)
03/22 16:33:56 INFO   :.main[31]: Specified configuration file: /u/user10/diagnose.conf
03/22 16:33:56 LOG    :.main[31]: EZZ8431I PAGENT STARTING

03/22 16:33:56 INFO   :.main[31]: Using log level 511
03/22 16:33:56 INFO   :..reg_process[30]: registering process with the system
03/22 16:33:56 INFO   :..reg_process[30]: attempt OS/390 registration
03/22 16:33:56 INFO   :..reg_process[30]: return from registration rc=0
03/22 16:33:56 INFO   :....mailslot_create[30]: creating mailslot for timer
03/22 16:33:56 INFO   :...mailbox_register[30]: mailbox allocated for timer
03/22 16:33:56 INFO   :...mailslot_create[30]: creating mailslot for terminate
03/22 16:33:56 INFO   :..mailbox_register[30]: mailbox allocated for terminate
03/22 16:33:56 INFO   :...mailslot_create[30]: creating mailslot for (broken) pipe
03/22 16:33:56 INFO   :..mailbox_register[30]: mailbox allocated for pipe
03/22 16:33:56 INFO   :...mailslot_create[30]: creating mailslot for process abend
03/22 16:33:56 INFO   :..mailbox_register[30]: mailbox allocated for abend
03/22 16:33:56 INFO   :...mailslot_create[30]: creating mailslot for process abort
03/22 16:33:56 INFO   :..mailbox_register[30]: mailbox allocated for abort
03/22 16:33:56 INFO   :...mailslot_create[30]: creating mailslot for reconfig
03/22 16:33:56 INFO   :..mailbox_register[30]: mailbox allocated for config
03/22 16:33:56 INFO   :...mailslot_create[30]: creating mailslot for process quit
03/22 16:33:56 INFO   :..mailbox_register[30]: mailbox allocated for quit
03/22 16:33:56 INFO   :..profile_initialize[30]: Pagent threads created installation(/flush) and initialization.
 (03)
03/22 16:33:56 INFO   :.check_main_config_file[29]: Main config file thread active
03/22 16:33:56 INFO   :...pinit_create_tmpFile[29]: Creating temporary working file /tmp/TCPCS.Pagent.tmp
03/22 16:33:56 INFO   :...pagent_mvs_command_handler[31]: Command handler thread active
03/22 16:33:56 INFO   :..pinit_init_tcpimages[29]: processed TcpImage statement: TcpImage
             TCPCS FLUSH 120

03/22 16:33:56 INFO   :...pinit_create_tmpFile[29]: Creating temporary working file /tmp/TCPCS2.Pagent.tmp
03/22 16:33:56 INFO   :..pinit_init_tcpimages[29]: processed TcpImage statement: TcpImage
             TCPCS2 /u/user10/pagent.conf

03/22 16:33:56 INFO   :..pinit_init_tcpimages[29]: Main config file refresh interval = 120 seconds
03/22 16:33:56 INFO   :.check_main_config_file[29]: Start up a config event update thread
03/22 16:33:56 INFO   :.check_main_config_file[29]: Finish starting policy profile installation(/flush) and
initialization.
 (04)
03/22 16:33:56 INFO   :.check_config_files[28]: Config processing thread active for image 'TCPCS2', index 1
03/22 16:33:56 INFO   :...settcpimage[28]: Associate with TCP/IP image name = TCPCS2
03/22 16:33:56 INFO   :...mailslot_create[30]: creating mailslot for dump
03/22 16:33:56 INFO   :.listen_thread[27]: PAPI server thread active
03/22 16:33:56 INFO   :.check_config_files[26]: Config processing thread active for image 'TCPCS', index 0
 (05)
03/22 16:33:56 INFO   :.config_files_update_event[25]: Config files update event thread active
03/22 16:33:56 INFO   :..mailbox_register[30]: mailbox allocated for dump
03/22 16:33:56 INFO   :.config_files_update_event[25]: IPC message queue id obtained, qid =10
 (06)
03/22 16:33:57 LOG    :.main[30]: EZZ8432I PAGENT INITIALIZATION COMPLETE

 (07)
03/22 16:33:57 SYSERR :..S390KernelInit[28]: socket(INET, DGRAM, 0) failed - errno EDC5112I Resource temporarily
unavailable.
03/22 16:33:57 INFO   :...settcpimage[26]: Associate with TCP/IP image name = TCPCS
03/22 16:33:57 SYSERR :..S390KernelInit[26]: socket(INET, DGRAM, 0) failed - errno EDC5112I Resource temporarily
unavailable.
 (08)
03/22 16:34:24 TRACE  :.config_files_update_event[25]: File Event notification is (__rfim_event)= 1, (__rfim_type)= 9
03/22 16:34:24 INFO   :.config_files_update_event[25]: Image TCPCS has been recycled
```

Figure 62. Policy Agent (Part 1 of 6)

```
(09)
03/22 16:34:24 INFO   :.config_files_update_event[25]: Restarting config processing thread for image 'TCPCS'
03/22 16:34:24 INFO   :.check_config_files[26]: Thread cleanup completed
03/22 16:34:25 INFO   :.check_config_files[26]: Config processing thread active for image 'TCPCS', index 0
03/22 16:34:25 INFO   :...settcpimage[26]: Associate with TCP/IP image name = TCPCS
(10)
03/22 16:34:25 INFO   :..FlushAllPolicies[26]: Start first by flushing all policies
03/22 16:34:25 INFO   :...profile_delete_ALL_ServiceClass[26]: Image name: TCPCS
03/22 16:34:25 OBJERR :...profile_delete_ALL_ServiceClass[26]: Failed reason code =70
03/22 16:34:25 WARNING:...profile_delete_ALL_ServiceClass[26]: ... reason is: P_RC_POLICY_NOT_FOUND
03/22 16:34:25 INFO   :...profile_delete_ALL_ServiceClass[26]: Finished deleting ALL Service Class in image 0
03/22 16:34:25 INFO   :...profile_delete_ALL_PolicyRule[26]: Image name:  TCPCS
03/22 16:34:25 OBJERR :...profile_delete_ALL_PolicyRule[26]: Failed  reason code =70
03/22 16:34:25 WARNING:...profile_delete_ALL_PolicyRule[26]: ... reason is: P_RC_POLICY_NOT_FOUND
03/22 16:34:25 INFO   :...profile_delete_ALL_PolicyRule[26]: Finished deleting ALL Policy Rules
(11)
03/22 16:34:25 INFO   :..pinit_fetch_policy_profile[26]: Processing policy config data file: /u/user10/diagnose.conf
for image TCPCS
03/22 16:34:25 INFO   :...processing_Stmt_UseLDAPRules[26]: (Re)starting LDAP processing thread for image 'TCPCS'
(12)
03/22 16:34:25 INFO   :...processing_Stmt_PolicyPerfMonit[26]: processing:  PolicyPerfMonitorForSDR
        Enable

03/22 16:34:25 TRACE  :...processing_Stmt_PolicyPerfMonit[26]: Turn ON policy performance monitor
(13)
03/22 16:34:26 INFO   :.ReadLdapRules[24]: LDAP processing thread active for image 'TCPCS', index 0
03/22 16:34:26 INFO   :....parse_policy_perf_monitor()[26]: Performance sampling interval 60 (sec)
03/22 16:34:26 INFO   :....parse_policy_perf_monitor()[26]: Loss Ratio specified is 20 (in 1/1000 th)
03/22 16:34:26 INFO   :....parse_policy_perf_monitor()[26]: Loss Weight Fraction specified is 25 (percent)
03/22 16:34:26 INFO   :....parse_policy_perf_monitor()[26]: Timeout ratio specified is 50 (in 1/1000 th)
03/22 16:34:26 INFO   :....parse_policy_perf_monitor()[26]: Timeout Weight Fraction specified is 50 (percent)
03/22 16:34:26 INFO   :....parse_policy_perf_monitor()[26]: Maximum Retrans Weight Fraction specified is 95(percent)
03/22 16:34:26 INFO   :....parse_policy_perf_monitor()[26]: Maximum Timeout Weight Fraction specified is 100(percent)
(14)
03/22 16:34:26 INFO   :...processing_Stmt_SetSubnetPrioTo[26]: processing:  SetSubnetPrioTosMask

(15)
03/22 16:34:26 INFO   :...processing_Stmt_ServicePolicyRu[26]: processing:  PolicyRule
        DiffServRule

03/22 16:34:26 INFO   :...processing_Stmt_ServiceCategori[26]: processing:  PolicyAction
        DiffServAction1

03/22 16:34:26 INFO   :...processing_Stmt_ServicePolicyRu[26]: processing:  PolicyRule
        TRRule

03/22 16:34:26 INFO   :...processing_Stmt_ServiceCategori[26]: processing:  PolicyAction
        TRAction

03/22 16:34:26 INFO   :...processing_Stmt_ServicePolicyRu[26]: processing:  PolicyRule
        BothRule

03/22 16:34:26 INFO   :...processing_Stmt_ServiceCategori[26]: processing:  PolicyAction
        DiffServAction2

03/22 16:34:26 INFO   :...processing_Stmt_ServiceCategori[26]: processing:  PolicyAction
        RSVPAction
```

Figure 62. Policy Agent (Part 2 of 6)

```
03/22 16:34:26 INFO   :...UpdateSCProfileData[26]: Updating SC profile for caller id: 1
03/22 16:34:26 INFO   :....computePolicyRuleWeight[26]: Weight computed for PR DiffServRule is: 108
03/22 16:34:26 INFO   :....computePolicyRuleWeight[26]: Weight computed for PR TRRule is: 2
03/22 16:34:26 INFO   :....computePolicyRuleWeight[26]: Weight computed for PR BothRule is: 4
03/22 16:34:26 INFO   :...UpdatePRProfileData[26]: Updating PR profile for caller id: 1
03/22 16:34:26 INFO   :....computePolicyRuleWeight[26]: Weight computed for PR DiffServRule is: 108
03/22 16:34:26 INFO   :....computePolicyRuleWeight[26]: Weight computed for PR BothRule is: 4
03/22 16:34:26 INFO   :....computePolicyRuleWeight[26]: Weight computed for PR TRRule is: 2
 (16)
03/22 16:34:26 INFO   :...profile_install_A_ServiceClass[26]: Service Class: DiffServAction1
03/22 16:34:26 INFO   :...profile_install_A_ServiceClass[26]: Finished installing Service Class: DiffServAction1
03/22 16:34:26 INFO   :...profile_install_A_ServiceClass[26]: Service Class: TRAction
03/22 16:34:26 INFO   :...profile_install_A_ServiceClass[26]: Finished installing Service Class: TRAction
03/22 16:34:26 INFO   :...profile_install_A_ServiceClass[26]: Service Class: DiffServAction2
03/22 16:34:26 INFO   :...profile_install_A_ServiceClass[26]: Finished installing Service Class: DiffServAction2
03/22 16:34:26 INFO   :...profile_install_A_ServiceClass[26]: Service Class: RSVPAction
03/22 16:34:26 INFO   :...profile_install_A_ServiceClass[26]: Finished installing Service Class: RSVPAction
03/22 16:34:26 TRACE  :....process_time_condition[26]: PR=DiffServRule inactive, next check in 446 minutes
03/22 16:34:26 TRACE  :....process_time_condition[26]: PR=BothRule active, next check in 446 minutes
03/22 16:34:26 INFO   :...profile_install_A_PolicyRule[26]: Finished installing policy rule: BothRule
03/22 16:34:26 TRACE  :....process_time_condition[26]: PR=TRRule active, next check in 446 minutes
03/22 16:34:26 INFO   :...profile_install_A_PolicyRule[26]: Finished installing policy rule: TRRule
03/22 16:34:26 INFO   :...profile_install_SubnetPrioTosMa[26]: Image name: TCPCS
03/22 16:34:26 INFO   :...profile_install_SubnetPrioTosMa[26]: Finished installing PrioTosMasks
 (17)
03/22 16:34:26 INFO   :..pinit_fetch_policy_profile[26]: Finish processing above policy config file
03/22 16:34:26 INFO   :.check_config_files[26]: Started a thread to monitor policy performance for image 0
03/22 16:34:26 INFO   :.policy_perf_monitor[23]: Performance monitor thread active for image 'TCPCS', index 0
03/22 16:34:26 INFO   :..settcpimage[23]: Associate with TCP/IP image name = TCPCS
03/22 16:34:26 INFO   :..settcpimage[24]: Associate with TCP/IP image name = TCPCS
 (18)
03/22 16:34:26 INFO   :.ReadLdapRules[24]: Contacting LDAP server 9.37.83.93 on port 9000
03/22 16:34:27 OBJERR :.ReadLdapRules[24]: Can not bind to directory server: No such object
03/22 16:34:27 LOG    :.ReadLdapRules[24]: EZZ8440I PAGENT CANNOT CONNECT TO LDAP SERVER

 (19)
03/22 16:34:27 INFO   :.ReadLdapRules[24]: Wait for retrying LDAP Connection = 60 seconds
03/22 16:35:27 INFO   :..settcpimage[24]: Associate with TCP/IP image name = TCPCS
03/22 16:35:27 INFO   :.ReadLdapRules[24]: Contacting LDAP server 9.37.83.93 on port 9000
03/22 16:35:30 OBJERR :.ReadLdapRules[24]: Can not bind to directory server: No such object
03/22 16:35:30 LOG    :.ReadLdapRules[24]: EZZ8440I PAGENT CANNOT CONNECT TO LDAP SERVER

03/22 16:35:30 INFO   :.ReadLdapRules[24]: Wait for retrying LDAP Connection = 300 seconds
 (20)
03/22 16:35:57 INFO   :.check_main_config_file[29]: Cancelling config processing thread for image 'TCPCS'
03/22 16:35:57 INFO   :.check_main_config_file[29]: Cancelling config processing thread for image 'TCPCS2'
03/22 16:35:57 INFO   :.check_config_files[26]: Thread cleanup completed
03/22 16:35:57 INFO   :...pinit_create_tmpFile[29]: Creating temporary working file /tmp/TCPCS.Pagent.tmp
03/22 16:35:57 INFO   :..pinit_init_tcpimages[29]: processed TcpImage statement: TcpImage
             TCPCS FLUSH 120

03/22 16:35:57 INFO   :...pinit_create_tmpFile[29]: Creating temporary working file /tmp/TCPCS2.Pagent.tmp
03/22 16:35:57 INFO   :..pinit_init_tcpimages[29]: processed TcpImage statement: TcpImage
             TCPCS2 /u/user10/pagent.conf

03/22 16:35:57 INFO   :..pinit_init_tcpimages[29]: Main config file refresh interval = 120 seconds
03/22 16:35:57 INFO   :.check_main_config_file[29]: Finish starting policy profile installation(/flush) and
initialization.
```

Figure 62. Policy Agent (Part 3 of 6)

```
(21)
03/22 16:35:57 INFO   :.check_config_files[26]: Config processing thread active for image 'TCPCS2', index 1
03/22 16:35:57 INFO   :...settcpimage[26]: Associate with TCP/IP image name = TCPCS2
03/22 16:35:57 INFO   :.check_config_files[22]: Config processing thread active for image 'TCPCS', index 0
03/22 16:35:57 INFO   :.check_config_files[28]: Thread cleanup completed
03/22 16:35:57 SYSERR :..S390KernelInit[26]: socket(INET, DGRAM, 0) failed - errno EDC5112I Resource temporarily
unavailable.
03/22 16:35:57 INFO   :...settcpimage[22]: Associate with TCP/IP image name = TCPCS
03/22 16:35:57 INFO   :..FlushAllPolicies[22]: Start first by flushing all policies
03/22 16:35:57 INFO   :...profile_delete_ALL_ServiceClass[22]: Image name: TCPCS
03/22 16:35:57 INFO   :...profile_delete_ALL_ServiceClass[22]: Finished deleting ALL Service Class in image 0
03/22 16:35:57 INFO   :...profile_delete_ALL_PolicyRule[22]: Image name:  TCPCS
03/22 16:35:57 INFO   :...profile_delete_ALL_PolicyRule[22]: Finished deleting ALL Policy Rules
03/22 16:35:57 INFO   :..pinit_fetch_policy_profile[22]: Processing policy config data file: /u/user10/diagnose.conf
for image TCPCS
03/22 16:35:57 INFO   :...processing_Stmt_UseLDAPRules[22]: (Re)starting LDAP processing thread for image 'TCPCS'
03/22 16:35:57 INFO   :...processing_Stmt_PolicyPerfMonit[22]: processing:  PolicyPerfMonitorForSDR
        Enable

03/22 16:35:57 TRACE  :...processing_Stmt_PolicyPerfMonit[22]: Turn ON policy performance monitor
03/22 16:35:57 INFO   :.ReadLdapRules[24]: Thread cleanup completed
03/22 16:35:57 INFO   :....parse_policy_perf_monitor()[22]: Performance sampling interval 60 (sec)
03/22 16:35:57 INFO   :....parse_policy_perf_monitor()[22]: Loss Ratio specified is 20 (in 1/1000 th)
03/22 16:35:57 INFO   :....parse_policy_perf_monitor()[22]: Loss Weight Fraction specified is 25 (percent)
03/22 16:35:57 INFO   :....parse_policy_perf_monitor()[22]: Timeout ratio specified is 50 (in 1/1000 th)
03/22 16:35:57 INFO   :....parse_policy_perf_monitor()[22]: Timeout Weight Fraction specified is 50 (percent)
03/22 16:35:57 INFO   :....parse_policy_perf_monitor()[22]: Maximum Retrans Weight Fraction specified is 95(percent)
03/22 16:35:57 INFO   :....parse_policy_perf_monitor()[22]: Maximum Timeout Weight Fraction specified is 100(percent)
03/22 16:35:57 INFO   :.ReadLdapRules[28]: LDAP processing thread active for image 'TCPCS', index 0
03/22 16:35:57 INFO   :...processing_Stmt_SetSubnetPrioTo[22]: processing:  SetSubnetPrioTosMask

03/22 16:35:57 INFO   :...processing_Stmt_ServicePolicyRu[22]: processing:  PolicyRule
        DiffServRule

03/22 16:35:57 INFO   :...processing_Stmt_ServiceCategori[22]: processing:  PolicyAction
        DiffServAction1

03/22 16:35:57 INFO   :...processing_Stmt_ServicePolicyRu[22]: processing:  PolicyRule
        TRRule

03/22 16:35:57 INFO   :...processing_Stmt_ServiceCategori[22]: processing:  PolicyAction
        TRAction

03/22 16:35:57 INFO   :...processing_Stmt_ServicePolicyRu[22]: processing:  PolicyRule
        BothRule

03/22 16:35:57 INFO   :...processing_Stmt_ServiceCategori[22]: processing:  PolicyAction
        DiffServAction2

03/22 16:35:57 INFO   :...processing_Stmt_ServiceCategori[22]: processing:  PolicyAction
        RSVPAction

03/22 16:35:57 INFO   :...UpdateSCProfileData[22]: Updating SC profile for caller id: 1
03/22 16:35:57 INFO   :....computePolicyRuleWeight[22]: Weight computed for PR DiffServRule is: 108
03/22 16:35:57 INFO   :....computePolicyRuleWeight[22]: Weight computed for PR TRRule is: 2
03/22 16:35:57 INFO   :....computePolicyRuleWeight[22]: Weight computed for PR BothRule is: 4
03/22 16:35:57 INFO   :...UpdatePRProfileData[22]: Updating PR profile for caller id: 1
03/22 16:35:57 INFO   :....computePolicyRuleWeight[22]: Weight computed for PR DiffServRule is: 108
03/22 16:35:57 INFO   :....computePolicyRuleWeight[22]: Weight computed for PR BothRule is: 4
03/22 16:35:57 INFO   :....computePolicyRuleWeight[22]: Weight computed for PR TRRule is: 2
```

Figure 62. Policy Agent (Part 4 of 6)

```
03/22 16:35:57 INFO   :...profile_install_A_ServiceClass[22]: Service Class: DiffServAction1
03/22 16:35:57 INFO   :...profile_install_A_ServiceClass[22]: Finished installing Service Class: DiffServAction1
03/22 16:35:57 INFO   :...profile_install_A_ServiceClass[22]: Service Class: TRAction
03/22 16:35:57 INFO   :...profile_install_A_ServiceClass[22]: Finished installing Service Class: TRAction
03/22 16:35:57 INFO   :...profile_install_A_ServiceClass[22]: Service Class: DiffServAction2
03/22 16:35:57 INFO   :...profile_install_A_ServiceClass[22]: Finished installing Service Class: DiffServAction2
03/22 16:35:57 INFO   :...profile_install_A_ServiceClass[22]: Service Class: RSVPAction
03/22 16:35:57 INFO   :...profile_install_A_ServiceClass[22]: Finished installing Service Class: RSVPAction
03/22 16:35:57 TRACE  :....process_time_condition[22]: PR=DiffServRule inactive, next check in 444 minutes
03/22 16:35:57 TRACE  :....process_time_condition[22]: PR=BothRule active, next check in 444 minutes
03/22 16:35:57 INFO   :...profile_install_A_PolicyRule[22]: Finished installing policy rule: BothRule
03/22 16:35:57 TRACE  :....process_time_condition[22]: PR=TRRule active, next check in 444 minutes
03/22 16:35:57 INFO   :...profile_install_A_PolicyRule[22]: Finished installing policy rule: TRRule
03/22 16:35:57 INFO   :...profile_install_SubnetPrioTosMa[22]: Image name: TCPCS
03/22 16:35:57 INFO   :...profile_install_SubnetPrioTosMa[22]: Finished installing PrioTosMasks
03/22 16:35:57 INFO   :..pinit_fetch_policy_profile[22]: Finish processing above policy config file
03/22 16:35:57 INFO   :.check_config_files[22]: Cancelling performance monitor thread
03/22 16:35:57 INFO   :.policy_perf_monitor[23]: Thread cleanup completed
03/22 16:35:57 INFO   :.check_config_files[22]: Started a thread to monitor policy performance for image 0
03/22 16:35:57 INFO   :.policy_perf_monitor[24]: Performance monitor thread active for image 'TCPCS', index 0
03/22 16:35:57 INFO   :..settcpimage[24]: Associate with TCP/IP image name = TCPCS
03/22 16:35:57 INFO   :..settcpimage[28]: Associate with TCP/IP image name = TCPCS
 (22)
03/22 16:35:57 INFO   :.ReadLdapRules[28]: Contacting LDAP server 9.37.83.93 on port 9000
03/22 16:35:58 INFO   :.ReadLdapRules[28]: Processing version 2 schema
03/22 16:35:58 INFO   :...search_ldap[28]: Searching for policies with base o=ibm, c=us, filter (&(objectClass=*)),
scope subtree
 (23)
03/22 16:35:59 INFO   :..sla_ldap_get_v2_policies[28]: 13 objects returned from LDAP search
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object o=IBM, c = US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object g=policy, o=IBM, c = US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object pg=groupA, g=policy, o=IBM, c= US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object pg=groupA-1, g=policy, o=IBM, c= US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object pa=inter1, pg=groupA-1, g=policy,
o=ibm, c=US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object pa=inter2, pg=groupA-1, g=policy,
o=ibm, c=US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object pr=rule1, pg=groupA-1, g=policy,
o=IBM, c= US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object pc=period1, pg=groupA-1, g=policy,
o=IBM, c= US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object pc=cond1, pg=groupA-1, g=policy,
o=IBM, c= US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object pc=cond1a, pg=groupA-1, g=policy,
o=IBM, c= US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object pr=rule2, pg=groupA-1, g=policy,
o=IBM, c= US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object pc=period2, pg=groupA-1, g=policy,
o=IBM, c= US
03/22 16:35:59 TRACE  :...process_ldap_group[28]: Search group contains object pc=cond2, pg=groupA-1, g=policy,
o=IBM, c= US
 (24)
03/22 16:35:59 INFO   :..sla_ldap_get_v2_policies[28]: Processing policy rule pr=rule1,pg=groupA-1,g=policy,o=ibm,c=US
03/22 16:35:59 INFO   :..sla_ldap_get_v2_policies[28]: Processing policy rule pr=rule2,pg=groupA-1,g=policy,o=ibm,c=US
 (25)
03/22 16:35:59 OBJERR :.....validate_numeric_att[28]: Attribute ProtocolNumberRange 'from' value TCP contains
non-numeric characters
```

Figure 62. Policy Agent (Part 5 of 6)

```
03/22 16:35:59 OBJERR :...process_policy_condition[28]: Attribute parsing error in policy condition entry for
attribute protocolnumberrange value TCP ...
03/22 16:35:59 OBJERR :..sla_ldap_get_v2_policies[28]: Error parsing policy condition pc=cond2,pg=groupA-1,g=policy,
o=ibm,c=us
03/22 16:35:59 OBJERR :..sla_ldap_get_v2_policies[28]: Rule pr=rule2,pg=groupA-1,g=policy,o=ibm,c=US discarded
due to reference errors
03/22 16:35:59 INFO   :..sla_ldap_get_v2_policies[28]: Processing policy action pa=inter1,pg=groupA-1,g=policy,o=ibm,c=US
03/22 16:35:59 INFO   :..sla_ldap_get_v2_policies[28]: Processing policy action pa=inter2,pg=groupA-1,g=policy,o=ibm,c=US
03/22 16:35:59 INFO   :..UpdateSCProfileData[28]: Updating SC profile for caller id: 2
03/22 16:35:59 INFO   :...computePolicyRuleWeight[28]: Weight computed for PR rule1 is: 102
03/22 16:35:59 INFO   :..UpdatePRProfileData[28]: Updating PR profile for caller id: 2
03/22 16:35:59 INFO   :...computePolicyRuleWeight[28]: Weight computed for PR rule1 is: 102
03/22 16:35:59 INFO   :...computePolicyRuleWeight[28]: Weight computed for PR DiffServRule is: 108
03/22 16:35:59 INFO   :...computePolicyRuleWeight[28]: Weight computed for PR BothRule is: 4
03/22 16:35:59 INFO   :...computePolicyRuleWeight[28]: Weight computed for PR TRRule is: 2
 (26)
03/22 16:35:59 LOG    :.ReadLdapRules[28]: EZZ8438I PAGENT POLICY DEFINITIONS CONTAIN ERRORS

03/22 16:35:59 INFO   :....settcpimage[28]: Associate with TCP/IP image name = TCPCS
 (27)
03/22 16:35:59 INFO   :...profile_install_A_ServiceClass[28]: Service Class: inter1
03/22 16:35:59 INFO   :...profile_install_A_ServiceClass[28]: Finished installing Service Class: inter1
03/22 16:35:59 INFO   :....settcpimage[28]: Associate with TCP/IP image name = TCPCS
03/22 16:35:59 INFO   :...profile_install_A_ServiceClass[28]: Service Class: inter2
03/22 16:35:59 INFO   :...profile_install_A_ServiceClass[28]: Finished installing Service Class: inter2
03/22 16:36:00 TRACE  :....process_time_condition[28]: PR=rule1 inactive, next check in 144 minutes
03/22 16:36:00 INFO   :...search_ldap[28]: Searching for policies with base o=ibm, c=us, filter (&(objectClass=
setsubnetpriotosmask)), scope subtree
 (28)
03/22 16:36:34 INFO   :.process_rqsts[23]: PAPI client thread active for connection 7
03/22 16:36:34 INFO   :..paapi_search[23]: Found Match to Policy Entry Name = BothRule
03/22 16:36:34 INFO   :..paapi_search[23]: Found Match to Policy Entry Name = DiffServAction2
03/22 16:36:34 INFO   :..paapi_search[23]: Found Match to Policy Entry Name = RSVPAction
03/22 16:36:34 INFO   :..paapi_search[23]: Found Match to Policy Entry Name = TRRule
03/22 16:36:34 INFO   :..paapi_search[23]: Found Match to Policy Entry Name = TRAction
 (29)
03/22 16:36:59 INFO   :.process_rqsts[23]: PAPI client thread active for connection 7
03/22 16:36:59 INFO   :..paapi_search[23]: Found Match to Policy Entry Name = DiffServRule
03/22 16:36:59 INFO   :..paapi_search[23]: Found Match to Policy Entry Name = rule1
 (30)
03/22 16:37:35 INFO   :...pagent_mvs_command_handler[31]: Received a CIBSTOP
03/22 16:37:35 EVENT  :..mailslot_sitter[30]: process received signal SIGTERM
03/22 16:37:35 INFO   :...check_signals[30]: received TERM signal
03/22 16:37:35 INFO   :......dreg_process[30]: deregistering process with the system
03/22 16:37:35 INFO   :......dreg_process[30]: attempt to dereg (ifaeddrg_byaddr)
03/22 16:37:35 INFO   :......dreg_process[30]: rc from ifaeddrg_byaddr  rc =0
03/22 16:37:35 INFO   :......pAPIterminate[30]: Cancelling PAPI server thread
03/22 16:37:35 INFO   :.....terminator[30]: Terminating global thread, relative id 0
03/22 16:37:36 INFO   :.....terminator[30]: Terminating global thread, relative id 1
03/22 16:37:36 INFO   :.....terminator[30]: Terminating global thread, relative id 2
03/22 16:37:37 INFO   :.....terminator[30]: Terminating global thread, relative id 3
03/22 16:37:38 INFO   :.....terminator[30]: Terminating image thread, relative id 0
03/22 16:37:38 INFO   :.check_config_files[22]: Thread cleanup completed
03/22 16:37:38 INFO   :.....terminator[30]: Terminating image thread, relative id 0
03/22 16:37:38 INFO   :.check_config_files[26]: Thread cleanup completed
03/22 16:37:39 INFO   :.....terminator[30]: Terminating image thread, relative id 1
03/22 16:37:39 INFO   :.ReadLdapRules[28]: Thread cleanup completed
03/22 16:37:39 INFO   :.....terminator[30]: Terminating image thread, relative id 2
03/22 16:37:39 INFO   :.policy_perf_monitor[24]: Thread cleanup completed
03/22 16:37:40 INFO   :.....terminator[30]: process terminated with exit code 0

03/22 16:37:40 INFO   :.....terminator[30]: EZZ8433I PAGENT SHUTDOWN COMPLETE
```

Figure 62. Policy Agent (Part 6 of 6)

Following are short descriptions of the numbered items in the trace.

01 The Policy Agent is started.

02 The main configuration file being used is reported.

03 A thread is started to process the main config file. Note: the number in brackets on each line (preceding the colon) denotes the thread ID. However, these IDs get dynamically reused as threads are cancelled and restarted.

04 The threads that process each TCP/IP stack configuration file are started.

05 A thread is started to monitor Policy Agent files for dynamic updates. Since the -i option was not specified the only files monitored are temporary files used to detect stack recycles.

06 Policy Agent initialization is complete. Note that messages with an ″EZZ″ prefix are also issued to the system console.

07 The threads that process each TCP/IP stack configuration file try to associate with their respective stacks. For this example, no stacks are started yet, so the failure to connect to each stack is reported.

08 TCPCS is started. The Policy Agent detects this and begins processing for the active stack.

09 The thread that processes this stack is restarted.

10 Since the FLUSH parameter was specified for this stack, the Policy Agent first attempts to delete all policies. However, in this case, no policies were previously defined so the deletion attempt fails, and the error is reported.

11 Configuration file processing begins for the TCPCS configuration file.

12 The PolicyPerfMonitorForSDR statement is processed.

13 The thread that communicates with the LDAP server for this stack is started.

14 The SetSubnetPrioTosMask statement is processed.

15 The service policy definition statements (policy rules and actions) are processed.

16 The policy actions and active rules are installed into the TCP/IP stack.

17 Configuration file processing ends for the TCPCS configuration file.

18 Since the ReadFromDirectory statement was specified for TCPCS, an attempt is made to connect to the LDAP server. However, the server password was specified incorrectly, and the error is reported.

19 The Policy Agent tries to reconnect to the LDAP server using a sliding time interval that starts at one minute, goes to 5 minutes, and repeats at 5 minute intervals up to 30 minutes. The configuration file is corrected to specify the right password. However, since the refresh interval for this stack is 2 minutes, a second attempt to contact the server is made before the changed configuration file is reread.

20 When the refresh interval expires, the changed configuration file is processed. Because no assumptions can be made about the contents of the configuration file relative to its previous contents, the Policy Agent cancels all threads associated with all TCP/IP stacks, and re-reads the configuration file.

21 The threads that process each TCP/IP stack configuration file are restarted. Note that since policies previously existed in stack TCPCS, the flush of those policies is now successful.

22 Another connect attempt is made for the LDAP server. Since the server password was corrected, this connect attempt is successful.

23 The number of objects returned from the LDAP server initial search is reported. Note that some of these objects may reference other objects, which are then retrieved from the LDAP server as needed. This initial search just retrieves objects based on the parameters configured on the ReadFromDirectory statement.

24 The policy actions and rules retrieved from the LDAP server are processed.

25 One of the rules defined on the LDAP server contains an invalid attribute value. This error is reported and the object is discarded.

26 A message is issued to the console to indicate that configuration errors were detected.

27 The policy actions are installed into the TCP/IP stack. For this example there are no active rules read from the LDAP server.

28 A pasearch command is processed. This particular command used the default parameters, meaning all active rules, and all associated actions, are returned for the search.

29 Another pasearch command is processed, this one specifying the -r and -l parameters, meaning only inactive rules are returned for the search.

30 A STOP (P) command is entered, and the Policy Agent shuts itself down.

Chapter 22. Diagnosing RSVP Agent Problems

Overview

The OS/390 UNIX RSVP Agent provides end-to-end resource reservation services on behalf of applications. 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, etc.

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.

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 OS/390 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 *OS/390 IBM Communications Server: IP Programmer's Reference* for more information on RAPI.

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 will receive 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 will 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 2 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 may specify different values than the sender). For Guaranteed, the flowspec takes the form of a Tspec followed by an Rspec.

Service Policies and RSVP Processing

Service 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 will use Controlled Load.

RSVP processing proceeds as follows. When an application uses RAPI to indicate it's 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 in 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 may choose to wait for the reservation to be in place, or may begin sending data before this happens (although such data will of course be treated by the network as though no reservation were in place). Any router or host that is incapable of supporting the requested reservation may send an error to the receiver, which is then free to perhaps try a lesser reservation.

The OS/390 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 *OS/390 IBM Communications Server: IP Configuration Guide* for more details on using LogLevel, as well as the location of the log file.

Certain other information can be useful in diagnosing RSVP Agent problems:

- Output from the TSO NETSTAT SLAP or onetstat -j commands
- Output from the pasearch command for RSVP scoped policies
- 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

If the RSVP Agent doesn't complete initialization, run it with LogLevel set to 511 and check the log file for error conditions. Common problems could include:

- RSVP Agent not authorized to security product

  The RSVP Agent must be authorized to a security product profile. Refer to the *OS/390 IBM 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 *OS/390 IBM 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's 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 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

If a Qos-aware application using RAPI is experiencing problems, check the following:

- 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.

Service Policy Problems

Service 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 following:

- RSVP policies not being applied to data flows

  If the limits imposed by defined RSVP-scoped service 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 as such a policy will never map 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:

Service Policy Unit	Tspec/Rspec Unit
MaxRatePerFlow: kilobits/second	r/R: bytes/second
MaxTokenBucketPerFlow: kilobits	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 21. Diagnosing Policy Agent Problems” on page 377 for more information about Policy Agent.

Example Log File

Figure 63 on page 400 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
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
```

Figure 63. RSVP Agent Processing Log (Part 1 of 6)

```
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

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
```

Figure 63. RSVP Agent Processing Log (Part 2 of 6)

```
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

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
```

Figure 63. RSVP Agent Processing Log (Part 3 of 6)

```
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 63. 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
```

Figure 63. RSVP Agent Processing Log (Part 5 of 6)

```
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
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 63. 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 aren't enabled for multicasting, so when the RSVP Agent tries to enable multicasting it gets reported as a warning. 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 that the application will send 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 23. Diagnosing Traffic Regulator Management Daemon (TRMD) Problems

Gathering Diagnostic Information

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 can be directed either to a set of log files or to the syslog daemon (syslogd). Refer to the *OS/390 IBM Communications Server: IP Configuration Guide* for more details on using the -d startup option, as well as the location of the log file.

Certain other information may be useful in diagnosing TRMD problems; TCP/IP CTRACE output, using the INTERNET and IOCTL CTRACE options.

Diagnosing TRMD Problems

The types of TRMD problems that could happen include the following.

- Initialization
- Gathering log data
- Unexpected results defining policies

Initialization Problems

Problems with initialization of the TRMD include the following.

- The TCP/IP stack is not up. Message EZZ8498I is received.

  Verify that the TCP/IP stack is up.
- The Policy Agent is not up. Message EZZ8483I is received.

  Make sure the Policy Agent is up.

Gathering Log Data Problems

Problems gathering log data include the following.

- Is syslogd started?

  The syslogd must be started prior to starting TRMD.
- If the syslogd has been started:

  Make sure the TRMD syslogd output file (defined in /etc/syslog.conf) has been created and exists before starting syslogd.

Unexpected Results When Defining Policies

If you don't see the expected results when defining policies, check the following:

- The wrong policy being mapped to traffic.

  There may be times when 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.

  1. When the policy rule priority is not specified, the weight depends on the number of attributes specified in the policy conditions.

2. When policy rule priority is specified, the weight is the specified priority plus 100. This 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. Specifying the priority in policy rules better controls situations where multiple rules map to the same set of traffic packets.

- Policies not installed in the TCP/IP stack.

  Is the stack in question configured via a TcpImage statement in the Policy Agent configuration file? Has the stack been started or restarted after the Policy Agent was started? If so, check that the temporary file used by the stack to inform the Policy Agent of restarts has not been deleted. Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for more details.

- Policies not mapping to the expected traffic.

  If you think data traffic should be mapped to certain policies, but it is not, then check to make sure you have the destination port or port range defined correctly on the PolicyRule statement for the policy. Also, you should check the possibility that a wrong policy is applied.

Documentation for the IBM Software Support Center

When contacting the IBM Software Support Center for problem resolution, some or all of the following information may be required:

- Gather TRMD debugging data by starting TRMD with command 'trmd -d 3'.
- Start CTRACE in the stack to gather related information.

Example Log File

Figure 64 on page 411 demonstrates some of the TRMD processing. The example log file was created using the `-d 2` startup options.

- The following TR policy configuration file was used to produce the example output:

```
TcpImage  TCPCS  FLUSH  10
PolicyAction   FloodControl
{
   PolicyScope       TR
   TypeActions       log limit statistics
   TotalConnections  50
   Percentage        5
   TimeInterval      1
}
PolicyRule   telnet
{
   DestinationPortRange   1 23           # telnet
   ServiceReference   FloodControl
}
```

- TRMD Processing Log

```
trmd -d 3

Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: EZZ8495I TRMD STARTED
 (1)
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_debug:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_debug:  papiLogFunc = 9840618 papiUserValue = 0
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_debug:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: Exit SIOCGIDAPPECB, policy ecb 874effc, log ecb 874eff8
 (2)
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: Entering paInitPapi
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: APIInitialize:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: open_socket:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: open_socket:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: APIInitialize:  ApiHandle = 9845190,  connfd = 7
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: APIInitialize:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: RegisterWithPolicyAPI:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: RegisterWithPolicyAPI:  Writing to socket = 7
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadBuffer:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadBuffer:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: RegisterWithPolicyAPI:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: Initialization Successful, api_handle = 9845190
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: Exiting paInitPapi
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: Entering paSearchPapi
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  Parameters
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  TcpImage         = TCPCS
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  PolicyObjectType = 0
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  PolicyActionType = 0
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  PolicyStatus     = 0
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  PolicyFilterType = 2
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  PolicyFilterName =
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  PolicyScopeName  = TR
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  Add to srh_p ApiSearchList, Addr = 98451B8
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  select from socket = 7
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  Reading from socket = 7
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  recvbuf_p = 14560, amount_read = 0,
   amount_to_read = 12
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  Reading from socket = 7
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  recvbuf_p = 98451F0, amount_read = 0,
   amount_to_read = 1864
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  ReadUnknownBuffer successful
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  Add Policy Entry to Queue, Entry = 98451F0
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  select from socket = 7
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  Reading from socket = 7
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  recvbuf_p = 14560, amount_read = 0,
   amount_to_read = 12
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  Reading from socket = 7
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  recvbuf_p = 9845940, amount_read = 0,
   amount_to_read = 688
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  ReadUnknownBuffer successful
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  Add Policy Entry to Queue, Entry = 9845940
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  select from socket = 7
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  Reading from socket = 7
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  recvbuf_p = 14560, amount_read = 0,
   amount_to_read = 12
```

Figure 64. Example of TRMD Processing Log (Part 1 of 5)

```
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: ReadUnknownBuffer:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  ReadUnknownBuffer successful
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  Exit count = 2 searchrequesthandle = 98451B8
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_search ok, count = 2
 (3)
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_get_policy_entry:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_get_policy_entry:  Parameters
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_get_policy_entry:  searchrequesthandle = 98451B8
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_get_policy_entry:  Exit p_entry = 98451F0
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_get_policy_entry:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: Process Entry = 1, Address = 98451F0
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_parse_policy_rule:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_parse_policy_rule:  Parameters
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_parse_policy_rule:  p_entry = 98451F0
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_parse_policy_rule:  Exit rule_p = 9845240
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_parse_policy_rule:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: Policy Rule = 1, Address = 9845240
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: Parsing TR rule successful
 (4)
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_get_policy_entry:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_get_policy_entry:  Parameters
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_get_policy_entry:  searchrequesthandle = 98451B8
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_get_policy_entry:  Exit p_entry = 9845940
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_get_policy_entry:  Exiting
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: Get TR action successful
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_parse_qos_action:  Entering
Mar  9 10:45:08 MVSVIC34 TRMD[83886096]: papi_parse_qos_action:  Parameters
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_parse_qos_action:  p_entry = 9845940
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_parse_qos_action:  Exit action_p = 9845994
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_parse_qos_action:  Exiting
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: QOS Action = 0, Address = 9845994
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Parsing TR action successful
 (5)
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Port 1 - 23, 1 minute collection interval
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 1
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 2
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 3
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 4
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 5
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 6
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 7
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 8
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 9
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 10
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 11
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 12
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 13
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 14
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 15
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 16
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 17
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 18
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 19
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 20
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 21
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 22
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Creating new stat block: 23
```

Figure 64. Example of TRMD Processing Log (Part 2 of 5)

```
 (6)
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Log on port 1
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_mem_free:  Entering
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_mem_free:  Parameters
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_mem_free:  free_entry_p = 9845940
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_mem_free:  Exiting
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_mem_free:  Entering
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_mem_free:  Parameters
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_mem_free:  free_entry_p = 0
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_mem_free:  free_entry_p parameter is NULL
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Exiting paSearchPapi
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Entering paTermPapi
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: UnRegisterFromPolicyAPI:  Entering
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: ReadBuffer:  Entering
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: ReadBuffer:  Exiting
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: UnRegisterFromPolicyAPI:  Result = 0
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: UnRegisterFromPolicyAPI:  Exiting
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: APITerminate:  Entering
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: APITerminate:  Dequeue srh_p ApiSearchList  srh_p = 98451B8
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_search_handle_free:  Entering
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_search_handle_free:  Parameters
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_search_handle_free:  handle = 9845190
  searchrequesthandle = 98451B8
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_search_handle_free:  FFlist_deq  PAAPI_PapiSearchList = 98451D4
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_search_handle_free:  FiFoE_p is NULL
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_search_handle_free:  Dequeue srh_p from ApiSearchList and
 free srh_p = 98451B8
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_search_handle_free:  Exit rc = 0
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: papi_search_handle_free:  Exiting
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: APITerminate:  Exiting
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Policy API Terminated, exiting paTermPapi
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: Exiting paInitPapi
Mar  9 10:45:09 MVSVIC34 TRMD[83886096]: EZZ8500I TRMD INITIALIZATION COMPLETE
 (7)
Mar  9 10:45:39 MVSVIC34 TRMD[83886096]: Woke Up: Log Timer Pop
 (8)
Mar  9 10:45:39 MVSVIC34 TRMD[83886096]:
 LOG: +0000 c9c4d396 87c481a3 00000002 00170000  IDLogDat........
 LOG: +0010 00000000 b3b76a5c f096d201 0925522a  .......*0oK.....
 LOG: +0020 00000002 00000030 00000032 05000000  ................
 LOG: +0030 00000000 00000000 00000000 00000000  ................
 (9) ** Dump hex log data retrieved from the TCPIP stack,
Mar  9 10:45:39 MVSVIC34 TRMD[83886096]: numlogs = 1
Mar  9 10:45:39 MVSVIC34 TRMD[83886096]: Current log type: 2
Mar  9 10:45:39 MVSVIC34 TRMD[83886096]: EZZ8499I connection refused:3/9/2000 15:45:17.80,
port=23,host=9.37.82.42,host_current=2,available=48,total=50,percentage=5
 (10)
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]: Woke Up: Log Timer Pop
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]: numlogs = 0
 (11)
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00010000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00020000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
```

Figure 64. Example of TRMD Processing Log (Part 3 of 5)

```
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00030000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00040000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00050000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00060000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00070000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00080000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00090000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 000a0000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 000b0000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 000c0000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 000d0000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 000e0000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
```

Figure 64. Example of TRMD Processing Log (Part 4 of 5)

```
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 000f0000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00100000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00110000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00120000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00130000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00140000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00150000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 00000004 00000000 00160000 00000000  ................
 STAT: +0010 00000000 00000000 00000000 00000000  ................
 STAT: +0020 00000000 00000000 00000000 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]:
 STAT: +0000 c9c4e2a3 81a389a2 00170700 00000002  IDStatis........
 STAT: +0010 0925522a 00000002 00000002 00000000  ................
 STAT: +0020 00000002 00000000 00000001 00000000  ................
 STAT: +0030 00000000 00000000 00000000 00000000  ................
 (12)
Mar  9 10:46:09 MVSVIC34 TRMD[83886096]: EZZ8485I TRMD ext statistics:
port=23,peak=2,host=9.37.82.42,hostpeak=2,requests=2,terminations=0,
current=2,period=0,warnings=1,qos_exceptions=0,suggested_limit=0,
suggested_percentage=0
 (13)
Mar  9 10:46:37 MVSVIC34 TRMD[83886096]: Exiting paInitPapi
Mar  9 10:46:37 MVSVIC34 TRMD[83886096]: EZZ8501I TRMD ENDED
 (14)
```

Figure 64. Example of TRMD Processing Log (Part 5 of 5)

Following are short descriptions of the numbered items in the trace.

01 The TRMD is started.

02 IOCTL is issued to pass some information between trdm and the TCP/IP stack.

03 Policy search completed successfully with two items returned. In this case,

one is the policy rule and another is the policy action. Also note that most data above is written by the routines of Policy Agent.

04 The policy rule is successfully retrieved and parsed.

05 The policy action is successfully retrieved and parsed.

06 TRMD internal data structures created.

07 TRMD started successfully.

08 The 30-second log timer was popped. IOCTL issued to get log data.

09 Dump hex log data retrieved from the TCPIP stack.

10 Log data.

11 Another 30-second interval. No log data retrieved this time.

12 Retrieve statistics data. Note that only port 23 has meaningful data.

13 Display statistics data.

14 TRMD is terminated.

Chapter 24. Diagnosing OROUTED Problems

The route daemon is a server that implements the routing information protocol (RIP) described in RFC 1058 (RIP Version 1) and in RFC 1723 (RIP Version 2). It provides an alternative to the static TCP/IP gateway definitions. When configured properly the MVS host running with OROUTED becomes an active RIP router in a TCP/IP network. The OROUTED server dynamically creates and maintains the network routing tables using RIP. RIP allows gateways and routers to periodically broadcast their routing tables to adjacent nodes. This enables the OROUTED server to update the host routing table. For example, the OROUTED server can determine if a new route has been created, if a route is temporarily unavailable, or if a more efficient route exists.

Before OROUTED was implemented for TCP/IP, 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 changes in the network. By implementing the Routing Information Protocol (RIP) between a host and TCP/IP, the OROUTED server dynamically updates the internal routing tables when changes to the network occur.

The OROUTED server reacts to network topology changes on behalf of TCP/IP by maintaining the host routing tables, processing and generating RIP datagrams, and performing error recovery procedures.

Figure 65 on page 418 shows the OROUTED environment.

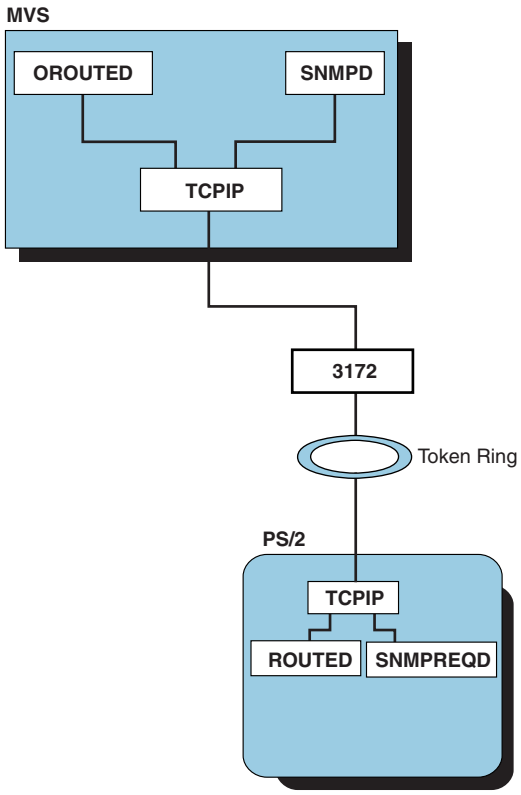


Figure 65. OROUTED Environment

The OROUTED protocol is based on the exchange of RIP messages. There are two types of messages:

Request message
Sent from a client (another RIP router) as a request to transmit all or part of this host routing table

Response message
Sent from OROUTED to a client (another RIP router) containing all or part of this host routing table

Definitions

OROUTED must be defined correctly to TCP/IP, and it needs to be started by a RACF authorized user ID. Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for detailed information about configuring OROUTED and TCP/IP definitions.

Diagnosing OROUTED Problems

Problems with OROUTED are generally reported under one of the following categories:

- "Abends" on page 419
- "OROUTED Connection Problems" on page 419
- "OS/390 UNIX oping Failures" on page 420
- "Incorrect Output" on page 421
- "Session Outages" on page 421

Use the information provided in the following sections for problem determination and diagnosis of errors reported against OROUTED.

Abends

An abend during OROUTED 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.

OROUTED Connection Problems

OROUTED connection problems are reported when OROUTED is unable to connect to TCP/IP. Generally, this type of problem is caused by an error in the configuration or definitions in TCP/IP.

In a CINET environment (multiple stacks), OROUTED attempts to connect to a stack whose name is determined by the TCPIPjobname keyword from the resolver configuration data set or file. If OROUTED cannot determine the TCPIPjobname, it will use a default TCPIPjobname of INET. If OROUTED cannot communicate with the stack pointed to by TCPIPjobname, it ends and issues an error message. A copy of OROUTED must be started for each stack requiring OROUTED services.

In configurations with multiple stacks, a copy of OROUTED must be started for each stack requiring OROUTED services. To associate OROUTED with a particular stack, use the environment variable RESOLVER_CONFIG to point to the data set or file that defines the unique TCPIPjobname.

OROUTED and OMPROUTE cannot run on the same stack concurrently.

Documentation

The following documentation should be available for initial diagnosis of OROUTED connection problems:

- PROFILE.TCPIP information
- TCPIP.DATA information
- OROUTED cataloged procedure, if used
- MVS system log
- OROUTED standard output and standard error
- The file (or data set) pointed to by the GATEWAYS_FILE environmental variable
- The file (or data set) pointed to by RESOLVER_CONFIG environmental variable
- The file (or data set) pointed to by ROUTED_PROFILE environmental variable

Analysis

Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for TCP/IP configuration-related problems.

Diagnostic steps for OROUTED connection problems:

1. Verify that the TCP/IP jobname is correct in the resolver configuration dataset or file. Remember, if you start OROUTED from a cataloged procedure, the file on the STDENV statement might define the RESOLVER_CONFIG environmental variable.
2. If starting from a cataloged procedure, make sure you are using the appropriate cataloged OS/390 UNIX application for OROUTED. Verify the correctness of the data set references.
3. Make sure that OROUTED is configured correctly in the PROFILE.TCPIP information. UDP port 520 must be reserved for OROUTED.

4. Make sure that OROUTED is configured correctly in the services file or data set. Verify that the assigned port number and service name are correct.
5. For network connectivity problems, see “Chapter 4. Diagnosing Network Connectivity Problems” on page 25.

OS/390 UNIX oping Failures

If an OS/390 UNIX oping command fails on a system where OROUTED is being used, a client is unable to get a response to an oping command. Before doing anything else, run onetstat -g. This should tell you which gateways are configured. If no gateways are configured, oping will not work.

Documentation

The following documentation should be available for initial diagnosis of oping failures:

- MVS system log
- PROFILE.TCPIP information
- Output from onetstat -g and onetstat -r

Analysis

Follow the steps below to diagnosis oping errors:

1. Perform oping loopback.
2. Make sure that the oping command contains a valid destination IP address for the remote host.

   If the destination IP address is a virtual IP address (VIPA), make sure that VIPA is defined correctly. See the *OS/390 IBM Communications Server: IP Configuration Reference* for more information on rules and recommendations when defining a virtual IP address.
3. Make sure that the router providing the RIP support involved in the oping transaction is active and is running with a correct level of OROUTED or some application that provides RIP support.

   If the destination router is not running RIP, make sure that static routes are defined from the destination router to the local host.
4. If the oping command was issued from a remote OS/390 host, issue an onetstat -g command from there to display its routing tables. Verify that the routes and networks are correct as defined in the *hlq*.PROFILE.TCPIP data set and the OROUTED gateways file.
5. If the oping command was issued from a remote OS/2® or DOS host, issue an onetstat -r command from there to display its routing tables. Verify that the routes and networks are correct as defined in the TCP/IP configuration and the /etc/gateways file of TCP/IP. From the OS/2 operating system, issue **ICAT** and select the Routing Information menu. From DOS, issue **IFCONFIG inet ip show** to display the TCP/IP configured routes.

   If there are any problems with the routes or networks, refer to OS/2 or DOS documentation for information about correcting NETSTAT problems.
6. If there are no problems with the routes and networks, check for broken or poorly-connected cables between the client and the remote host. This includes checking the intranet interfaces (such as token ring and Ethernet) on the OROUTED server.

Incorrect Output

Problems with incorrect output are reported when the data sent to the client is not seen in its expected form. This could be incorrect TCP/IP output, RIP commands that are not valid, incorrect RIP broadcasting information, incorrect updates of routing tables, or truncation of packets.

Documentation

The following documentation should be available for initial diagnosis of incorrect output:

- OROUTED cataloged procedure, if used
- MVS system log
- OROUTED standard output and standard error
- PROFILE.TCPIP information
- Output from `onetstat -g` and `onetstat -r`

Analysis

Table 33 shows types of incorrect output and describes the actions needed for initial diagnosis of the error.

Table 33. OROUTED Incorrect Output

Incorrect Output	Action Steps
TCP/IP Incorrect Output	1. If the TCP/IP console shows a TCP/IP error message, refer to *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)* and follow the directions for system programmer response for the message. 2. In the event of TCP/IP loops or hangs, see “Chapter 3. Diagnosing Abends, Loops, and Hangs” on page 21.
OROUTED Incorrect Output	If an OROUTED error message is displayed, refer to *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)* and follow the directions for system programmer response for the message.

Session Outages

Session outages are reported as an unexpected abend or termination of a TCP/IP connection.

Documentation

The following documentation should be available for initial diagnosis of session outages:

- OROUTED cataloged procedure, if used
- MVS system log
- OROUTED standard output and standard error
- TCPIP CTRACE (see “Chapter 5. TCP/IP Services Traces and IPCS Support” on page 47)
- PROFILE.TCPIP information
- Output from `onetstat -g` and `onetstat -r`

Analysis

Table 34 on page 422 shows types of session outages and describes the steps needed for initial diagnosis of the error.

Table 34. OROUTED Session Outages

Session Outage	Action Steps
TCP/IP session outage	1. If the TCP/IP console shows a TCP/IP error message, refer to *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)* and follow the directions for system programmer response for the message. 2. Examine the external CTRACE for information about the error. 3. In the event of an TCP/IP abend, see “Chapter 3. Diagnosing Abends, Loops, and Hangs” on page 21.
OROUTED session outage	If an OROUTED error message is displayed, refer to *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)* and follow the directions for system programmer response for the message.

OROUTED Traces and Debug Information

There are many TCP/IP traces that can be useful in identifying the cause of OROUTED problems. OROUTED traces and debug requests can be started from the OS/390 UNIX shell, or they can be started from an MVS cataloged procedure. This section discusses both methods.

Note: OROUTED trace output is sent to syslogd unless you specify otherwise. See “Where to Send OROUTED Trace Output” on page 425.

Figure 66 on page 423 shows a sample OROUTED environment.

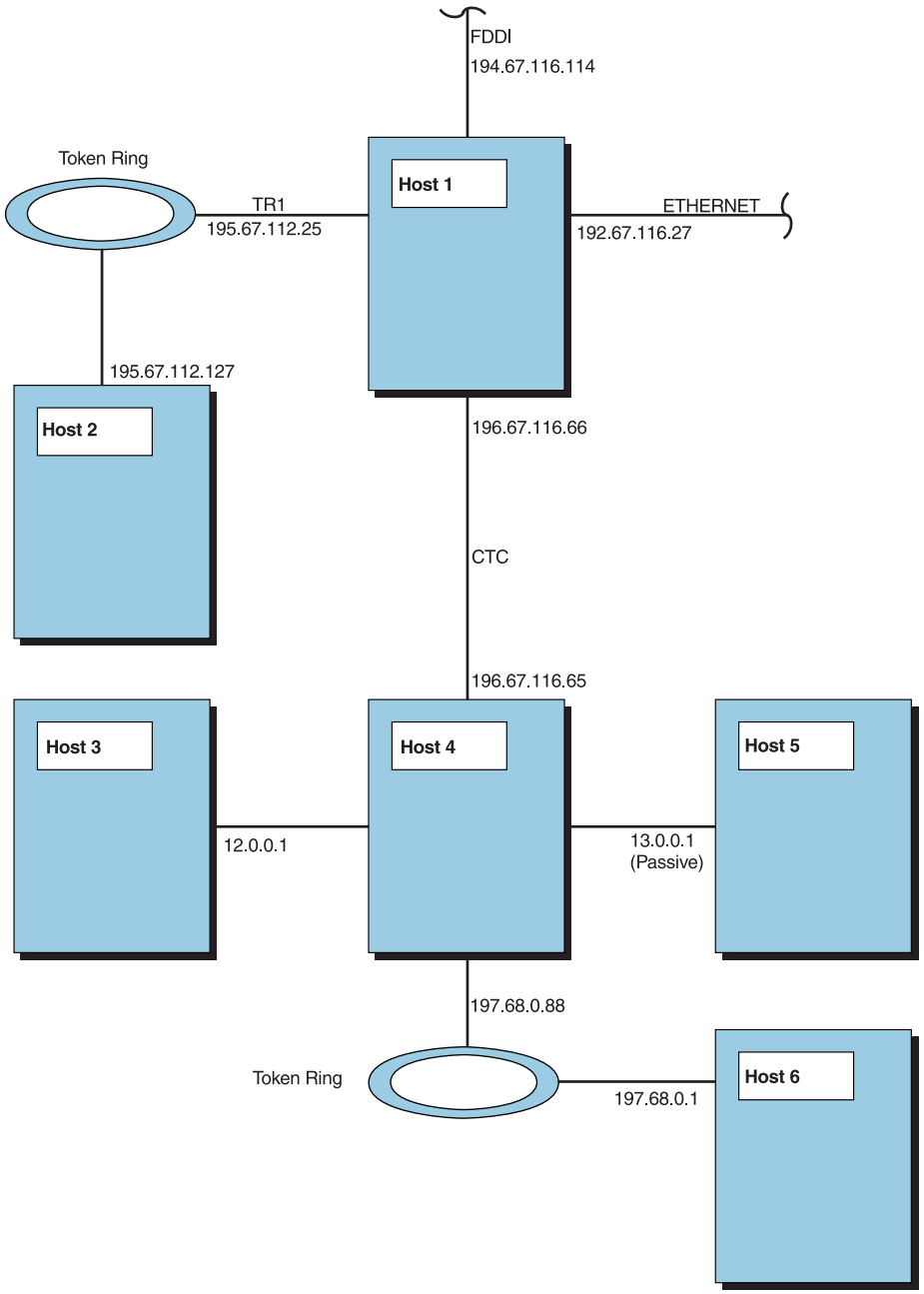


Figure 66. Sample OROUTED Environment

Starting OROUTED Traces from the OS/390 UNIX Shell

From TSO, issue the OMVS command, which puts you at a UNIX-like prompt. Then type the `orouted` command followed by one or more of the following parameters.

Note: OROUTED traces can be dynamically started and stopped using the MODIFY command. For more information, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.

-t Activates tracing of actions by the OROUTED server.

-t -t Activates tracing of actions and packets sent or received.

-t -t -t 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.

-t -t -t -t
Activates tracing of actions, packets sent or received, packet history, and packet contents. The RIP network routing information is included in the trace output.

-dp Activates tracing of packets to and from adjacent routers and received and broadcasted RIP network routing tables. Packets are shown in data format in the trace output. The information is written to STDOUT.

-ep Sends print statement to STDOUT and STDERR.

-d Enables internal debug information, which consists of internal code points. The information is written to STDOUT. Use this parameter only if IBM service requests the information.

Notes:

1. The parameters described here are only those that activate tracing. Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for more information about all of the OROUTED parameters.
2. To run orouted in the background, add an ampersand (&) to the command, as in the command `orouted &`
3. You can enter more than one parameter, with a space after each parameter; for example, `orouted -t -t -t -dp`

Starting OROUTED Traces from an MVS Catalogued Procedure

The OROUTED traces are controlled by parameters on `PARM=` in the PROC statement of the OROUTED cataloged procedure.

For example:

```
//OROUTED EXEC PGM=OROUTED,REGION=4096K,TIME=NOLIMIT,
// PARM=('POSIX(ON)',
// 'ENVAR("_CEE_ENVFILE=DD:STDENV")/-ep -t -t')
```

The OROUTED parameters that control tracing are:

Note: OROUTED traces can be dynamically started and stopped using the MODIFY command. For more information, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.

-t Activates tracing of actions by the OROUTED server.

-t -t Activates tracing of actions and packets sent or received.

-t -t -t 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.

-t -t -t -t
Activates tracing of actions, packets sent or received, packet history, and packet contents. The RIP network routing information is included in the trace output.

-dp Activates tracing of packets to and from adjacent routers and received and broadcasted RIP network routing tables. Packets are shown in data format in the trace output. The information is written to STDOUT.

-ep Sends print statement to STDOUT and STDERR.

-d Enables internal debug information, which consists of internal code points. The information is written to STDOUT. Use this parameter only if IBM service requests the information.

Notes:

1. Each parameter is separated by a blank.
2. Parameters can be specified in mixed case.
3. The parameters described here are only those that activate tracing. Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for more information about all of the OROUTED parameters.

Where to Send OROUTED Trace Output

Normally, trace output is sent to syslogd, which includes a great deal of information about many TCP/IP components, not just OROUTED. If you want to see only OROUTED output, use the `-ep` parameter when you start a trace. This sends output to syslogd and to STDOUT.

If you are running in the OS/390 UNIX shell, using the `-ep` parameter sends the output to the shell session screen. You can save this output by redirecting it into a file with the greater-than (>) character (for example, `orouted -ep -t -t > orouted.stdout`).

Note: The `-ep` parameter cannot be altered using the MODIFY command.

Stopping OROUTED

You can stop OROUTED in several ways:

- From an OS/390 UNIX shell superuser ID, issue the `kill` command to the process ID (PID) associated with OROUTED.

  To find the PID, use one of the following methods:

  – Use `D OMVS,U=USERID`. (This is the USERID that started OROUTED from the shell.)
  – Use the `ps -ef` command from the shell.
  – Write down the PID when you start OROUTED.
  – Use a shell command pipeline such as

  ```
  kill $(ps | awk '/orouted/' {print $1}')
  ```

  (In this case, you do not need to know the PID.)
- From MVS, issue the MODIFY command, specifying the parameter `-k` or `-kdr`. The `-k` parameter stops OROUTED, while the `-kdr` parameter stops OROUTED and deletes all dynamic routes. For example, the following command stops an OROUTED server started with a procedure named OROUTED.

  ```
  MODIFY OROUTED,PARMS=-k
  ```

  For more information on this command, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.
- From MVS, issue `P` *procname* where *procname* is the procedure name used to start OROUTED. If OROUTED was started from the OS/390 UNIX shell, the

procname is *useridX*, where *X* is the sequence number set by the system. To determine the sequence number, issue `/d a,l` from any MVS console, to see the programs running.

Note: To see the OS/390 UNIX application name in the output, use `D OMVS,U=USERID`, as described previously.

Changing Trace and Debug Levels with MODIFY

Whether you start OROUTED from OMVS or MVS, you can use the MVS MODIFY to change command trace levels. The MODIFY syntax is

MODIFY procname,parms=*parm*

A modify format for MVS might look like this:

```
MODIFY OROUTED,PARMS=-t -t -t
```

If you had started from the OS/390 UNIX shell, you would use something like this:

```
MODIFY useridX,PARMS=-t -t -t
```

where X is the sequence number for the OROUTED job. To determine the sequence number, see “Stopping OROUTED” on page 425.

The `-tq` parameter disables all traces.

You can also turn on debug information with the parameters `-d`, `-dp`, or both. To turn off debug information, use the `-dq` parameter.

For more information on using the MODIFY command, refer to the *OS/390 IBM Communications Server: IP Configuration Reference* .

OROUTED Trace Example and Explanation

Figure 67 on page 427 shows an example of an OROUTED trace that was generated using `-ep -t -t -t` parameters. Short descriptions of numbered items in the trace follow the figure.

```
EZZ4990I OE RouteD server initializing. Level 10.00
EZZ4980I Using catalog '/usr/lib/nls/msg/C/routed.cat' for OE RouteD messages.
 (1)  EZZ4828I Input parameter(s): -ep -t -t -t
EZZ4985I Setting High Level Qualifier (HLQ) to 'CS390'
EZZ4988I OE RouteD established affinity with 'TCPCS'
EZZ4929I Port 520 assigned to route
EZZ4932I ************************************************
EZZ5001I Opening OE RouteD profile /u/user156/r.p
EZZ4932I ************************************************
EZZ5002I RIP_SUPPLY_CONTROL: RIP2M
EZZ4932I ************************************************
EZZ4850I * Processing interface TR1
EZZ4932I ************************************************
EZZ4948I This interface is not point-to-point
EZZ4943I Adding network route for interface
EZZ4882I Tue Jan 18 14:25:32 2000:
EZZ4883I ADD destination 9.0.0.0, router 9.67.116.94, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL timer 0
EZZ4943I Adding subnetwork route for interface
EZZ4883I ADD destination 9.67.116.0, router 9.67.116.94, metric 1
flags UP state INTERFACE|CHANGED|SUBNET timer 0
EZZ4932I ************************************************
EZZ4850I * Processing interface VLINK1
EZZ4932I ************************************************
EZZ4965I Virtual interface
EZZ4948I This interface is not point-to-point
EZZ4943I Adding network route for interface
EZZ4883I ADD destination 94.0.0.0, router 94.94.94.94, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL|VIRTUAL timer 0
EZZ4943I Adding subnetwork route for interface
EZZ4883I ADD destination 94.94.94.92, router 94.94.94.94, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL|SUBNET|VIRTUAL timer 0
EZZ4943I Adding host route for interface
EZZ4883I ADD destination 94.94.94.94, router 94.94.94.94, metric 1
flags UP|HOST state INTERFACE|CHANGED|INTERNAL|VIRTUAL timer 0
EZZ4932I ************************************************
EZZ4850I * Processing interface CTC00
EZZ4932I ************************************************
EZZ4940I Point-to-point interface, using broadaddr
EZZ4949I Interface CTC00 not up
EZZ4932I ************************************************
EZZ4850I * Processing interface CTC02
EZZ4932I ************************************************
EZZ4940I Point-to-point interface, using dstaddr
EZZ4943I Adding network route for interface
EZZ4883I ADD destination 215.215.215.0, router 215.215.215.2, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL timer 0
EZZ4943I Adding host route for interface
EZZ4883I ADD destination 215.215.215.4, router 215.215.215.2, metric 1
flags UP|HOST state INTERFACE|CHANGED timer 0
```

Figure 67. Example of an OROUTED Trace (Part 1 of 7)

```
EZZ4932I ************************************************
EZZ4850I * Processing interface VIPL0B0B0B0B
EZZ4932I ************************************************
EZZ4965I Virtual interface
EZZ4948I This interface is not point-to-point
EZZ4943I Adding network route for interface
EZZ4883I ADD destination 11.0.0.0, router 11.11.11.11, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL|VIRTUAL timer 0
EZZ4943I Adding subnetwork route for interface
EZZ4883I ADD destination 11.11.11.0, router 11.11.11.11, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL|SUBNET|VIRTUAL timer 0
EZZ4943I Adding host route for interface
EZZ4883I ADD destination 11.11.11.11, router 11.11.11.11, metric 1
flags UP|HOST state INTERFACE|CHANGED|INTERNAL|VIRTUAL timer 0
EZZ4932I ************************************************
EZZ4850I * Processing interface EZASAMEMVS
EZZ4932I ************************************************
EZZ4940I Point-to-point interface, using broadaddr
EZZ4943I Adding network route for interface
EZZ4883I ADD destination 150.150.0.0, router 150.150.150.1, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL timer 0
EZZ4943I Adding subnetwork route for interface
EZZ4883I ADD destination 150.150.150.0, router 150.150.150.1, metric 1
flags UP state INTERFACE|CHANGED|SUBNET timer 0
EZZ4932I ************************************************
EZZ4934I * Opening GATEWAYS file (/u/user156/g.f)
EZZ4932I ************************************************
EZZ4925I Start of GATEWAYS processing:
EZZ4945I ifwithnet:  compare with TR1
EZZ4947I netmatch 9.67.116.88 and 9.67.116.94
EZZ4936I Adding passive net route 0.0.0.0 via gateway 9.67.116.88, metric 1
EZZ4883I ADD destination 0.0.0.0, router 9.67.116.88, metric 1
flags UP|GATEWAY state PASSIVE|CHANGED|DEFAULT timer 0
EZZ5013I RIP2 authentication disabled at interface level (TR1)
EZZ5013I RIP2 authentication disabled at interface level (VLINK1)
EZZ5013I RIP2 authentication disabled at interface level (CTC00)
EZZ5019I Joining multicast group 224.0.0.9 on interface CTC00
EZZ5013I RIP2 authentication disabled at interface level (CTC02)
EZZ5019I Joining multicast group 224.0.0.9 on interface CTC02
EZZ5013I RIP2 authentication disabled at interface level (VIPL0B0B0B0B)
EZZ5013I RIP2 authentication disabled at interface level (EZASAMEMVS)
EZZ5019I Joining multicast group 224.0.0.9 on interface EZASAMEMVS
EZZ4926I End of GATEWAYS processing
EZZ4849I OE RouteD Server started
 (2)  EZZ4899I     REQUEST to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:25:32 2000
EZZ4949I Interface CTC00 not up
EZZ4899I     REQUEST to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4899I     REQUEST to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:25:32 2000
 (3)  EZZ4829I Waiting for incoming packets
EZZ4899I     REQUEST from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4899I     REQUEST from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4899I     RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4899I     REQUEST from 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:25:32 2000
```

Figure 67. Example of an OROUTED Trace (Part 2 of 7)

```
EZZ4958I supply 9.67.116.94 -> 520 via TR1
EZZ4899I     RESPONSE to 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:25:32 2000
EZZ4899I     RESPONSE from 9.67.116.88 -> 520: ver 1 Tue Jan 18 14:25:32 2000
EZZ4882I Tue Jan 18 14:25:32 2000:
EZZ4883I ADD destination 9.67.113.0, router 9.67.116.88, metric 2
flags UP|GATEWAY state CHANGED|SUBNET timer 0
EZZ4883I ADD destination 9.67.114.0, router 9.67.116.88, metric 2
flags UP|GATEWAY state CHANGED|SUBNET timer 0
EZZ4899I     RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:25:32 2000
EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 9.67.116.88 -> 520: ver 1 Tue Jan 18 14:25:50 2000
EZZ4829I Waiting for incoming packets
 (4)  EZZ4957I 30 second timer expired (poll interfaces for status)
EZZ4957I 30 second timer expired (broadcast)
EZZ4958I supply 9.67.116.255 -> 0 via TR1
EZZ4899I     RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:26:02 2000
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:26:02 2000
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:26:02 2000
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:26:02 2000
EZZ4899I     RESPONSE from 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:26:02 2000
EZZ4899I     RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:26:02 2000
EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 9.67.116.88 -> 520: ver 1 Tue Jan 18 14:26:19 2000
EZZ4829I Waiting for incoming packets
EZZ4957I 60 second timer expired (rescan kernel for interfaces)
EZZ4957I 30 second timer expired (poll interfaces for status)
EZZ4949I Interface CTC00 not up
EZZ4957I 30 second timer expired (broadcast)
EZZ4958I supply 9.67.116.255 -> 0 via TR1
EZZ4899I     RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:26:33 2000
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:26:33 2000
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:26:33 2000
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:26:33 2000
EZZ4899I     RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:26:33 2000
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:26:33 2000
EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
 (5)  EZZ4828I Input parameter(s): -T -T -T -T
EZZ4871I Tracing packet contents enabled Tue Jan 18 14:26:46 2000
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 9.67.116.88 -> 520: ver 1 Tue Jan 18 14:26:49 2000
EZZ4902I     destination 9.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 1
EZZ4902I     destination 9.67.114.0 metric 1
```

Figure 67. Example of an OROUTED Trace (Part 3 of 7)

```
EZZ4829I Waiting for incoming packets
EZZ4957I 30 second timer expired (poll interfaces for status)
EZZ4957I 30 second timer expired (broadcast)
EZZ4958I supply 9.67.116.255 -> 0 via TR1
EZZ4899I     RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 9.0.0.0 metric 1
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 2
EZZ4902I     destination 9.67.114.0 metric 2
EZZ4902I     destination 9.67.116.0 metric 1
EZZ4902I     destination 150.150.150.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 9.0.0.0 metric 1
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 2
EZZ4902I     destination 9.67.114.0 metric 2
EZZ4902I     destination 9.67.116.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 9.0.0.0 metric 1
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 2
EZZ4902I     destination 9.67.114.0 metric 2
EZZ4902I     destination 9.67.116.0 metric 1
EZZ4902I     destination 150.150.150.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4899I     RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 9.0.0.0 metric 1
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 2
EZZ4902I     destination 9.67.114.0 metric 2
EZZ4902I     destination 9.67.116.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4899I     RESPONSE from 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
```

Figure 67. Example of an OROUTED Trace (Part 4 of 7)

```
EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
EZZ4957I 60 second timer expired (rescan kernel for interfaces)
EZZ4957I 30 second timer expired (poll interfaces for status)
EZZ4949I Interface TR1 not up
 (6)  EZZ4891I *** Packet history for interface TR1 ***
EZZ4898I Output trace:
EZZ4899I     REQUEST to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:25:32 2000
EZZ4903I request for full tables
EZZ4899I     RESPONSE to 9.67.116.94 -> 520: ver 1 Tue Jan 18 14:25:32 2000
 (7)  EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4899I     RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:26:02 2000
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4899I     RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:26:33 2000
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4899I     RESPONSE to 9.67.116.255 -> 520: ver 1 Tue Jan 18 14:27:03 2000
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4892I *** End packet history ***
EZZ4882I Tue Jan 18 14:27:33 2000:
EZZ4889I CHANGE metric destination 9.0.0.0, router 9.67.116.94, from 1 to 16
EZZ4889I CHANGE metric destination 9.67.113.0, router 9.67.116.88, from 2 to 16
EZZ4889I CHANGE metric destination 9.67.114.0, router 9.67.116.88, from 2 to 16
EZZ4889I CHANGE metric destination 9.67.116.0, router 9.67.116.94, from 1 to 16
EZZ4916I deleting route to interface TR1 (timed out)
EZZ4949I Interface CTC00 not up
EZZ4957I 30 second timer expired (broadcast)
EZZ4949I Interface TR1 not up
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:27:33 2000
EZZ4902I     destination 9.0.0.0 metric 16
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 16
EZZ4902I     destination 9.67.114.0 metric 16
EZZ4902I     destination 9.67.116.0 metric 16
EZZ4902I     destination 150.150.150.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
```

Figure 67. Example of an OROUTED Trace (Part 5 of 7)

```
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:27:33 2000
EZZ4902I     destination 9.0.0.0 metric 16
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 16
EZZ4902I     destination 9.67.114.0 metric 16
EZZ4902I     destination 9.67.116.0 metric 16
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:27:33 2000
EZZ4902I     destination 9.0.0.0 metric 16
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 16
EZZ4902I     destination 9.67.114.0 metric 16
EZZ4902I     destination 9.67.116.0 metric 16
EZZ4902I     destination 150.150.150.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4899I     RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:27:33 2000
EZZ4902I     destination 9.0.0.0 metric 16
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 16
EZZ4902I     destination 9.67.114.0 metric 16
EZZ4902I     destination 9.67.116.0 metric 16
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
EZZ4957I 30 second timer expired (poll interfaces for status)
EZZ4957I 30 second timer expired (broadcast)
EZZ4949I Interface TR1 not up
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:28:03 2000
EZZ4902I     destination 9.0.0.0 metric 16
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 16
EZZ4902I     destination 9.67.114.0 metric 16
EZZ4902I     destination 9.67.116.0 metric 16
EZZ4902I     destination 150.150.150.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:28:03 2000
EZZ4902I     destination 9.0.0.0 metric 16
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 16
EZZ4902I     destination 9.67.114.0 metric 16
EZZ4902I     destination 9.67.116.0 metric 16
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
```

Figure 67. Example of an OROUTED Trace (Part 6 of 7)

```
EZZ4829I Waiting for incoming packets
EZZ4899I     RESPONSE from 215.215.215.2 -> 520: ver 2 Tue Jan 18 14:28:03 2000
EZZ4902I     destination 9.0.0.0 metric 16
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 16
EZZ4902I     destination 9.67.114.0 metric 16
EZZ4902I     destination 9.67.116.0 metric 16
EZZ4902I     destination 150.150.150.0 metric 1
EZZ4902I     destination 150.150.0.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4899I     RESPONSE from 150.150.150.1 -> 520: ver 2 Tue Jan 18 14:28:03 2000
EZZ4902I     destination 9.0.0.0 metric 16
EZZ4902I     destination 11.11.11.0 metric 1
EZZ4902I     destination 11.0.0.0 metric 1
EZZ4902I     destination 9.67.113.0 metric 16
EZZ4902I     destination 9.67.114.0 metric 16
EZZ4902I     destination 9.67.116.0 metric 16
EZZ4902I     destination 215.215.215.0 metric 1
EZZ4902I     destination 94.94.94.92 metric 1
EZZ4902I     destination 94.0.0.0 metric 1
EZZ4829I Waiting for incoming packets
EZZ4829I Waiting for incoming packets
 (5)  EZZ4828I Input parameter(s): -KDR
EZZ4949I Interface TR1 not up
EZZ4949I Interface CTC00 not up
EZZ4958I supply 224.0.0.9 -> 0 via CTC02
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:28:08 2000
EZZ4902I     destination 9.0.0.0 metric 16
EZZ4902I     destination 11.11.11.0 metric 16
EZZ4902I     destination 11.0.0.0 metric 16
EZZ4902I     destination 9.67.113.0 metric 16
EZZ4902I     destination 9.67.114.0 metric 16
EZZ4902I     destination 9.67.116.0 metric 16
EZZ4902I     destination 150.150.150.0 metric 16
EZZ4902I     destination 150.150.0.0 metric 16
EZZ4902I     destination 94.94.94.92 metric 16
EZZ4902I     destination 94.0.0.0 metric 16
EZZ4958I supply 224.0.0.9 -> 0 via EZASAMEMVS
EZZ4899I     RESPONSE to 224.0.0.9 -> 520: ver 2 Tue Jan 18 14:28:08 2000
EZZ4902I     destination 9.0.0.0 metric 16
EZZ4902I     destination 11.11.11.0 metric 16
EZZ4902I     destination 11.0.0.0 metric 16
EZZ4902I     destination 9.67.113.0 metric 16
EZZ4902I     destination 9.67.114.0 metric 16
EZZ4902I     destination 9.67.116.0 metric 16
EZZ4902I     destination 215.215.215.0 metric 16
EZZ4902I     destination 94.94.94.92 metric 16
EZZ4902I     destination 94.0.0.0 metric 16
```

Figure 67. Example of an OROUTED Trace (Part 7 of 7)

Following are short descriptions of the numbered items in the trace:

(1) EZZ4828I shows the parameters used for this trace.

(2) EZZ4899I describes the type of packet being sent or received.

(3) EZZ4829I means the system is waiting for incoming packets.

(4) EZZ4957I shows timer information.

(5) At this point, trace parameters were changed using a MODIFY command.

(6) EZZ4891I shows packet history.

(7) EZZ4902I provides packet detail for each route contained in an RIP packet.

Documentation for the IBM Software Support Center

When contacting the IBM Software Support Center for problem resolution, some or all of the following information may be required:

- PROFILE.TCPIP information
- Output from `onetstat -g` and `onetstat -r`
- Network diagram or layout
- OROUTED profile (the file pointed to by the ROUTED_PROFILE environmental variable)
- OROUTED gateways file (the file pointed to by the GATEWAYS_FILE environmental variable)
- Standard output and standard error of OROUTED using the `-ep` parameter with full tracing on (for example, running `orouted -ep -t -t -t -t`)

Chapter 25. Diagnosing OMPROUTE Problems

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 (RIP Version 1) and in RFC 1723 (RIP Version 2). OMPROUTE provides an alternative to the static TCP/IP gateway definitions. When configured properly, the MVS host running with OMPROUTE becomes an active OSPF and/or RIP router in a TCP/IP network. Either (or both) of these two routing protocols can be 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 an OS/390 UNIX application. It requires the Hierarchical File System (HFS) to operate.
- OMPROUTE can be started from an MVS procedure, from the OS/390 shell, or from AUTOLOG.
- The OMPROUTE subagent provides an alternative to DISPLAY commands for displaying 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 *OS/390 IBM 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 via the split-stack function requires multiple instances of OMPROUTE.
- OMPROUTE and OROUTED cannot run on the same TCP/IP stack concurrently.
- All dynamic routes are deleted from the routing table upon initialization of OMPROUTE.
- Internet Control Message Protocol (ICMP) Redirects are ignored when OMPROUTE is active.
- Unlike OROUTED, OMPROUTE does not make use of the BSD Routing Parameters. Instead, the Maximum Transmission Unit (MTU), subnet mask, and destination address parameters are configured via the OSPF_Interface, RIP_Interface, and Interface statements in the OMPROUTE configuration file.
- 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 is 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
    - Communications between OMPROUTE and the TCP/IP stack

For details on using TCP/IP Services Component trace support with OMPROUTE, see “TCP/IP Services Component Trace for OMPROUTE” on page 447 and “Appendix A. Collecting Component Trace Data” on page 513.

- If you want to communicate a routing protocol over an interface, configure the interface to OMPROUTE using the OSPF_INTERFACE or RIP_INTERFACE configuration statement.
- Interfaces that are not involved in the communication of the RIP or OSPF protocol (such as 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 as specified on the INTERFACE statement are acceptable.
- 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.
- 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 static routes, and the use of static routes (defined via the GATEWAY TCP/IP configuration statement) is not recommended. Static routes may 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 via the static route. For example, if you define a host route through one interface and that interface becomes unreachable, OMPROUTE does not acknowledge your static route and does not define a host route through alternate interface.

If static routes must be defined, all static routes will be considered to be of equal cost and static routes will not be 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 configuration statement or set SEND_STATIC_ROUTES = YES on the RIP_Interface configuration statement if you want for the static routes to be advertised to other routers.

OMPROUTE must be defined correctly to TCP/IP. For detailed information about TCP/IP definitions, refer to the chapter on configuring OMPROUTE in the *OS/390 IBM 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.

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 either OMPROUTE or OROUTED 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 terminates.

For details on diagnosing problems connecting to the SNMP agent, see “SNMP Connection Problems” on page 338.

Routing Failures

If a client is unable to reach its desired destination on a system where OMPROUTE is being used, the first step in diagnosis is to issue the onetstat -r command. This command displays the routes in the TCP/IP routing table and lets you determine if the contents are as expected relative to the destination trying to be reached.

Documenting Routing Failures

The following documentation should be available for initial diagnosis of routing failures:

- MVS system log.
- Output from onetstat -r.
- SYSLOGD.
- The data set containing OMPROUTE trace and debug information. For details, see “OMPROUTE Traces and Debug Information” on page 438.
- TCP/IP and OMPROUTE CTRACE. For information about generating an OMPROUTE Component Trace, see “TCP/IP Services Component Trace for OMPROUTE” on page 447.
- Output from appropriate OMPROUTE DISPLAY commands as described in *OS/390 IBM Communications Server: IP Configuration Reference*.

Analyzing Routing Failures

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 command does not show the expected results relative to the desired destination, do one or more of the following:

- Make sure that the router(s) 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 *OS/390 IBM Communications Server: IP Configuration Reference* 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 447). 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 OS/390 Shell

If OMPROUTE is started from the OS/390 shell command line (using the omproute command), parameters can be specified to indicate the level of tracing or debugging desired.

- **-tn (where n is a supported trace level)**

  This option specifies the external tracing level. It is intended for customers, testers, service, or developers, and provides information on the operation of the routing application. This option can be used for many purposes, such as debugging a configuration, education on the operation of the routing application, verification of testcases, 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 is intended for service or developers only and provides internal debugging information needed for debugging problems. The following level is supported:

  - 1 = Internal debugging messages. Turns on DPIdebug(2).

- **-dn (where n is a supported debug level)**

  This option specifies the internal debugging level. It is intended for service or developers only and provides internal debugging information needed for debugging problems. The following levels are supported:

  - 1 = Internal debugging messages
  - 2 = Unformatted hex packet trace
  - 3 = Function entry/exit trace
  - 4 = Task add/run

Notes:

1. The -tn, -sn, and -dn options affect OMPROUTE performance. As a result, you may have to increase the Dead Router Interval on OSPF interfaces to prevent neighbor adjacencies from collapsing.

2. 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*.
3. 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=BPXBATCH,REGION=4096K,TIME=NOLIMIT,
//         PARM='PGM /usr/sbin/omproute -t1 -d2'
```

For a description of the parameters that can be specified, see "Starting OMPROUTE Tracing and Debugging from the OS/390 Shell" on page 438.

Starting OMPROUTE Tracing and Debugging Using the MODIFY Command

Whether you start OMPROUTE from the OS/390 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 that may have been set as a start option. This command is intended for customers, testers, service, or developers.
  - 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>,DEBUG=<debug-level>**

  Use the DEBUG command to change the debug level that may have been set as a start option. This command is intended for service or developers only.
  - 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>,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

Output from OMPROUTE tracing and debugging is written to the debug output destination. If debug or trace is turned on, the output destination is based on the

OMPROUTE_DEBUG_FILE environment variable. If OMPROUTE_DEBUG_FILE is not defined then the output defaults to stdout. In the special case where a MODIFY command is used to enable tracing and stdout is undefined, then the output destination defaults to the file ″omproute_debug″ in the temporary directory defined by the environment variable TMPDIR (usually /tmp).

When OMPROUTE_DEBUG_FILE is defined, the first trace file created will be the value coded on OMPROUTE_DEBUG_FILE. When that file is full, the extension will be changed to 00N, where N is in the range 1–4. The current file is always the value defined as OMPROUTE_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 data OMPROUTE traces. You tailor this parameter to your network complexity, and available HFS storage capacity, or both. See the *OS/390 IBM Communications Server: IP Configuration Guide* for details on this environment variable.

Sample OMPROUTE Trace Output

Figure 68 on page 441 is a sample OMPROUTE trace with descriptions for some of the trace entries:

```
1   EZZ7800I OMPROUTE starting
    EZZ7889I 00   ACTIVE COMP=SYSTCPRT SUB=USER1464
    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
    EZZ7937I The OSPF routing protocol is Enabled
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 default route defined
    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, neighbor 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 68. 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 68. 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 68. 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 68. 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
     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 68. 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 68. 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)

2 OMPROUTE learns of TCP/IP stack interfaces

3 Direct routes are added for each TCP/IP stack 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 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 tracing level to 2 (adds formatted packets)

26 Formatted OSPF packet

27 Formatted RIP packet

28 Request received to change tracing level back to 1

29 OMPROUTE learns of stopped TCP/IP interface

30 Routes over stopped interface are deleted

31 Neighbor over stopped interface transitions to state “Down”

32 Stopped interface transitions to state “Down”

TCP/IP Services Component Trace for OMPROUTE

CS for OS/390 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 “Appendix A. Collecting Component Trace Data” on page 513. For detailed descriptions, refer to the following books:

- *OS/390 MVS Diagnosis: Tools and Service Aids* for information about Component Trace procedures
- *OS/390 MVS Initialization and Tuning Reference* for information about the SYS1.PARMLIB member
- *OS/390 MVS System Commands* for information about trace commands
- *OS/390 MVS Authorized Assembler Services Guide* for information about procedures and return codes for CTRACE macros

Specifying Trace Options

You can specify Component Trace options at TCP/IP initialization or after TCP/IP has initialized.

Specifying Options at Initialization

A default minimum Component Trace is always started during OMPROUTE initialization. To customize the parameters used to initialize the trace, update the SYS1.PARMLIB member CTIORA00(see Figure 69 on page 449). For a description of trace options, see Table 35 on page 450.

Note: Besides specifying the trace options, you can also change the OMPROUTE trace buffer size. The buffer size can be changed only at OMPROUTE initialization.

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 1M and the MINIMUM tracing option.

```
/*******************************************************************/
/*                                                                 */
/*  TCP/IP for MVS                                                 */
/*  SMP/E Distribution Name: CTIORA00                              */
/*  Part name:               CTIORA00 sample                       */
/*  Component name:          OSPF                                  */
/*                                                                 */
/*     COPYRIGHT = Licensed Materials - Program Property of IBM.   */
/*                 This product contains "Restricted Materials of  */
/*                 IBM" 5647-A01 (C) Copyright IBM Corp. 1996,1998 */
/*                 All rights reserved.                            */
/*                 US Government Users Restricted Rights -         */
/*                 Use, duplication or disclosure restricted       */
/*                 by GSA ADP Schedule Contract with IBM Corp.     */
/*                 See IBM Copyright Instructions                  */
/*                                                                 */

/*  DESCRIPTION = This parmlib member causes component trace for   */
/*                 the CS/390 TCP/IP OMPROUTE application          */
/*                 to be initialized with a trace buffer size of   */
/*                 256K.                                           */
/*                                                                 */
/*                 This parmlib member only lists those TRACEOPTS  */
/*                 values specific to TCP/IP.  For a complete list */
/*                 of TRACEOPTS keywords and their values see      */
/*                 OS/390 MVS INITIALIZATION AND TUNING REFERENCE. */
/*                                                                 */
/*                                                                 */
/*******************************************************************/
 TRACEOPTS
 /* ------------------------------------------------------------- */
 /*   ON OR OFF: PICK 1                                           */
 /* ------------------------------------------------------------- */
          ON
 /*       OFF                                                     */
 /* ------------------------------------------------------------- */
 /*   BUFSIZE: A VALUE IN RANGE 128K TO 16M                       */
 /* ------------------------------------------------------------- */
          BUFSIZE(256K)
 /* ------------------------------------------------------------- */
 /*   OPTIONS: NAMES OF FUNCTIONS TO BE TRACED, OR "ALL"          */
 /* ------------------------------------------------------------- */
 /*       OPTIONS(            */
 /*               'ALL     '   */
 /*              ,'MINIMUM '   */
 /*              ,'ROUTE   '   */
 /*              ,'PACKET  '   */
 /*              ,'OPACKET '   */
 /*              ,'RPACKET '   */
 /*              ,'IPACKET '   */
 /*              ,'SPACKET '   */
 /*               )            */

 /* ------------------------------------------------------------- */
 /*   WRITER PROCEDURE NAME                                       */
 /* ------------------------------------------------------------- */
 /*              WTR()
```

Figure 69. SYS1.PARMLIB Member CTIORA00

Table 35 describes the available trace options.

Table 35. OMPROUTE Trace 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 CS/390 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 RIP protocol.
IPACKET	Select inbound packets sent from CS/390 TCP/IP with information regarding route or interface changes.
SPACKET	Traces inbound and outbound packets sent between the SNMP agent and the OMPROUTE subagent.

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 "Appendix A. Collecting Component Trace Data" on page 513.

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. (See "Appendix A. Collecting Component Trace Data" on page 513.) 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

You cannot use the following as options when formatting OMPROUTE component traces:

- ALL
- MINIMUM
- PACKET

Chapter 26. 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).

Figure 70 shows the NCPROUTE environment.

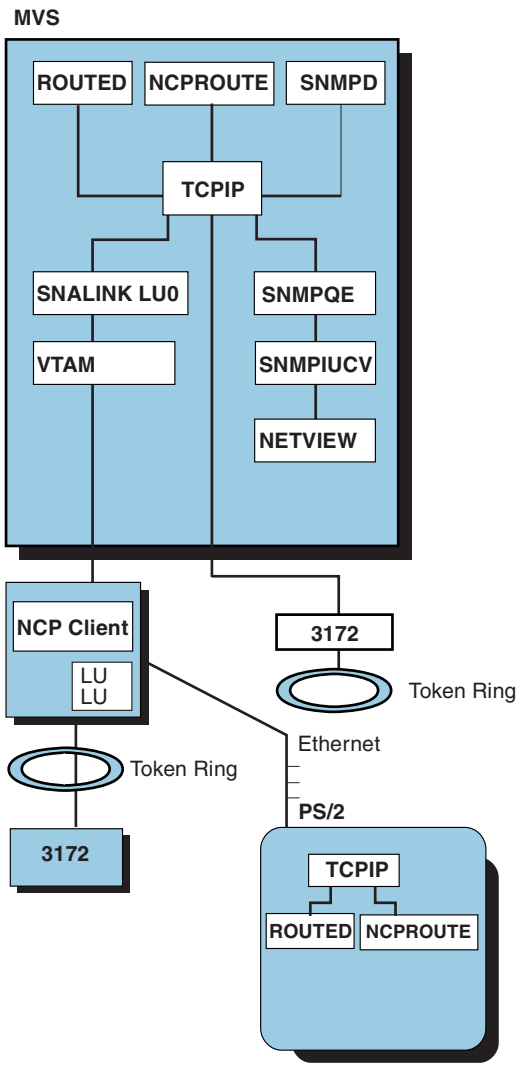


Figure 70. NCPROUTE Environment

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 tables, 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). There are 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 a NCP client routing tables. 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 tables. 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 tables.
- **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 commands for retransmitting of and responding to RIP broadcasts and trace requests. There are four types of RIP commands that can be encapsulated in a Transport PDU:

- **Request**: NCP received a request from a client (another RIP router) to retransmit RIP broadcasts.
- **Response**: Sent from NCP to its client (another RIP router) as a response to retransmit RIP broadcasts.
- **TraceOn**: NCP received a request from a client (another RIP router) to enable the actions trace provided by NCPROUTE.
- **TraceOff**: NCP received a request from a client (another 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 in 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 *OS/390 IBM Communications Server: IP User's Guide* for detailed information about the SNMP commands.

The following list describes the MIB variables registered for use by NCPROUTE:
- **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 *OS/390 IBM Communications Server: IP User's Guide* for detailed information about the MIB variables.

Definitions

NCPROUTE must be defined correctly to both NCP and TCP/IP. NCPROUTE must be in the OBEY list of the *hlq*.PROFILE.TCPIP data set, and 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 *OS/390 IBM 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.

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, hlq.SEZAINST(NCPRPROF). The SNMP agent community name must also be defined in the *hlq*.PW.SRC data set for proper verification.

Refer to *OS/390 IBM 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
- PING 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 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 *OS/390 MVS Diagnosis: Procedures* or to “Chapter 3. Diagnosing Abends, Loops, and Hangs” on page 21, in this manual 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
  - *hlq*.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 36 shows symptoms of connection problems and refers to the steps needed for initial diagnosis of the error.

Table 36. NCPROUTE Connection Problems

Connection Problem	Analysis Steps
NCP client connection failure	1, 2, 7, 8, 10, 14
NCPROUTE connection failure	1, 3, 5, 6, 7, 8, 10, 11, 14
SNALINK LU0 connection failure	1, 3, 7, 8, 10, 12
SNMP Agent connection failure	4, 9, 10, 13

Table 37 gives the diagnostic steps referred to in Table 36.

For TCP/IP configuration-related problems, refer to *OS/390 IBM Communications Server: IP Configuration Reference* for more information.

Table 37. Diagnostic Steps for NCPROUTE Connection Problems

Step	Action
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.
1a	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 will be the default UDP port 580.
1b	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 will use this port as a destination port when broadcasting RIP packets to adjacent routers.
1c	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.
1d	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 *OS/390 IBM Communications Server: IP Configuration Guide*.
1e	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.
1f	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 .PROFILE.TCPIP data set.
1g	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 .PROFILE.TCPIP data set.

Table 37. Diagnostic Steps for NCPROUTE Connection Problems (continued)

Step	Action
1h	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.
1i	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.
3a	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 .PROFILE.TCPIP data set. The cataloged procedure name (NCPROUT) is referred to on the OBEY, 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 .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.
7a	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 OBEYFILE commands. This allows TCP/IP to be reconfigured without having to shut down TCP/IP.
8	If you are using OROUTED, make sure that the routing parameters (BSDROUTINGPARMS statement) for the NCP clients are defined correctly. In addition, directly-connected passive routes to the NCP clients must be defined in the .ETC.GATEWAYS data set.
9	If you are using SNMP, make sure that the SNMP agent is configured correctly in the .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 and SNMPQE) for the SNMP agent and client are referred to on the OBEY, AUTOLOG (optional), and PORT statements.
9a	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.

Table 37. Diagnostic Steps for NCPROUTE Connection Problems (continued)

Step	Action
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.
10a	Make sure that the SNALINK devices are started by the START statement in the .PROFILE.TCPIP data set. The SNALINK devices can also be started by an OBEYFILE START command.
10b	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.
10c	Make sure that the NCP client physical and logical lines for the internet interfaces (token ring and Ethernet) are active.
10d	Make sure that NCST lines are active for the SNALINK LU0 sessions.
10e	Make sure that VTAM cross-domain resource managers (CDRMs) are active in the MVS hosts.
11	For network connectivity problems, refer to “Chapter 4. Diagnosing Network Connectivity Problems” on page 25.
13	For SNMP problems, refer to “Chapter 20. Diagnosing Simple Network Management Protocol (SNMP) Problems” on page 333.
14	For OROUTED problems, refer to “Chapter 24. Diagnosing OROUTED Problems” on page 417.

PING Failures

NCPROUTE PING failures are reported when a client is unable to get a response to a PING command for a destination in a TCP/IP network where there are NCPs acting as RIP servers. The NCP client suspected of having the problem must be determined before beginning analysis.

Documentation

The following documentation should be available for initial diagnosis of PING failures:

- NCPROUTE console log
- TCP/IP console log
- *hlq*.PROFILE.TCPIP data set
- NCP client network definitions data set (NCP generation)

Analysis

Table 38 shows symptoms of PING failures and refers to the steps needed for initial diagnosis of the error.

Table 38. NCPROUTE PING Failures

PING Failure	Analysis Steps
Incorrect response	1, 2, 3, 4, 5, 6, 7, 8
Time-outs	2, 9

Table 39 gives the diagnostic steps referred to in Table 38 on page 457 .

Table 39. Diagnostic Steps for NCPROUTE PING Failures

Step	Action
1	Make sure that 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 client on an MVS host, issue a NETSTAT GATE command to display the routing tables. Verify that the routes and networks are correct as defined in the *hlq*.PROFILE.TCPIP data set and the NCP client GATEWAYS data set, which is a member of *hlq*.NCPROUTE.GATEWAYS partitioned data set referred to by the *hlq*.SEZAINST(EZBNRPRF) data set defined in the NCPROUTE cataloged procedure.
3a	If the host is running with OROUTED, verify the routes and networks defined in the *hlq*.ETC.GATEWAYS data set. Directly-connected passive routes to the NCP clients must be defined in the *hlq*.ETC.GATEWAYS data set to ensure NCPROUTE connectivity with the NCP clients.
3b	If there are any problems with the routes and networks, refer to “Using NETSTAT and onetstat” on page 31.
4	If the PING command was issued from a client on a host running the OS/2 or DOS operating system, issue a NETSTAT -r command to display the routing tables. Verify that the routes and networks are correct as defined in the TCPIP configuration. From the OS/2 operating system, issue ICAT and select the Routing Information menu. From DOS, issue `IFCONFIG inet ip` show to display the TCP/IP configured routes.
4a	If the host is running with OROUTED, verify the routes and networks defined in the GATEWAYS file in the ETC subdirectory of TCPIP.
4b	If there are any problems with the routes or networks, refer to OS/2 or DOS documentation on correcting NETSTAT problems.
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 that 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 will need 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 25.
8	For more information about diagnosing PING problems, refer to “Using PING and oping” on page 29.
9	For more information about diagnosing PING time-outs, refer to “Correcting Timeout Problems” on page 30.

Incorrect Output

Problems with incorrect output are reported when the data sent to the client is not seen in its expected form. This could be incorrect TCTIP output, incorrect SNALINK LU0 output, RIP commands that are not valid, incorrect RIP broadcasting information, incorrect updates of routing tables, truncation of 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)
- Documentation for TCPIP incorrect output
  - TCP/IP console log
  - *hlq*.TCPIP. PROFILE data set
  - *hlq*.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 40 shows types of incorrect output and refers to the steps needed for initial diagnosis of the error.

Table 40. 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

Table 41 gives the diagnostic steps referred to in Table 40.

Table 41. Diagnostic Steps for NCPROUTE Incorrect Output

Step	Action
1	If the TCP/IP console shows a message, refer to *OS/390 IBM Communications Server: IP Messages Volume 2 (EZB)* and follow the directions for system programmer response for the message.
1a	Information in the TCP/IP console log should contain a detailed description of the error.
1b	In the event of TCP/IP loops or hangs, refer to Chapter 3. Diagnosing Abends, Loops, and Hangs.
3	If the NCPROUTE console shows a message, refer to *OS/390 IBM Communications Server: IP Messages Volume 2 (EZB)* and follow the directions for system programmer response for the message.
4	If the SNMP agent or client console shows a message, refer to *OS/390 IBM Communications Server: IP Messages Volume 2 (EZB)* and follow the directions for system programmer response for the message.
4a	For more information about diagnosing SNMP problems, refer to “Chapter 20. Diagnosing Simple Network Management Protocol (SNMP) Problems” on page 333.

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 will result 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 42 shows symptoms of session outages and refers to the steps needed for initial diagnosis of the error.

Table 42. NCPROUTE Session Outages

Session Outage	Analysis Steps
TCP/IP session outage	1
SNALINK LU0 session outage	2
NCPROUTE-to-NCP client session outage	3
NCPROUTE-to-SNMP agent session outage	4

Table 43 gives the diagnostic steps referred to in Table 42.

Table 43. Diagnostic Steps for NCPROUTE Session Outages

Step	Action
1	If the TCP/IP console shows a TCP/IP error message, refer to *OS/390 IBM Communications Server: IP Messages Volume 2 (EZB)* and follow the directions for system programmer response for the message.
1b	If TCP/IP abended, refer to Chapter 3. Diagnosing Abends, Loops, and Hangs.
3	If the NCPROUTE console shows an NCPROUTE error message, refer to *OS/390 IBM Communications Server: IP Messages Volume 2 (EZB)* and follow the directions for system programmer response for the message.
4	If the SNMP agent console shows a SNMP error message, refer to *OS/390 IBM Communications Server: IP Messages Volume 2 (EZB)* and follow the directions for system programmer response for the message.
4a	For more information about diagnosing SNMP problems, refer to “Chapter 20. Diagnosing Simple Network Management Protocol (SNMP) Problems” on page 333.

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.

Note: 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 will be traced.)

For example:

```
//NCPROUT PROC MODULE=NCPROUTE,PARMS='/-t -t'
```

Note: 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 *OS/390 IBM Communications Server: IP Configuration Reference*.

-tq Deactivate 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.

Notes:

1. A slash (/) must precede the first parameter.
2. Each parameter must be separated by a blank.
3. Mixed case is allowed for the parameters.
4. The parameters for the NCPROUTE procedure are case-sensitive.
5. There are no third- or fourth-level global tracing options like those on the GATEWAY statements in the NCPROUTE GATEWAYS data set. The system will use the higher of the two settings for a specific NCP client.
6. The data packets trace option is not available for selective tracing.
7. The parameters described here are only those that activate tracing. Refer to *OS/390 IBM 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 control selective tracing for an NCP client are is `trace.level`. The value that follows this keyword indicates the trace level to be used.

Value	Meaning
0	Do 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, packetssent 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.

Notes:

1. The selective traces must be defined prior to activation of an NCP client or prior to starting the NCPROUTE cataloged procedure.
2. Refer to *OS/390 IBM 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 71 on page 463 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 -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 will be 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 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 71. 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 71. 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 71. 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 71. 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 71. 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
                          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 71. 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 71. 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 71. 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 71. 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 71. 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 2 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 will supply 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 a network unreachable route (in the range metric 16—infinite), so NCPROUTE will update its routing tables for the NCP client.

19 NCPROUTE is deleting the NCP client from its tables.

20 The NCP client 5-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 27. Diagnosing X.25 NPSI Problems

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 72 on page 476 shows the X.25 NPSI environment.

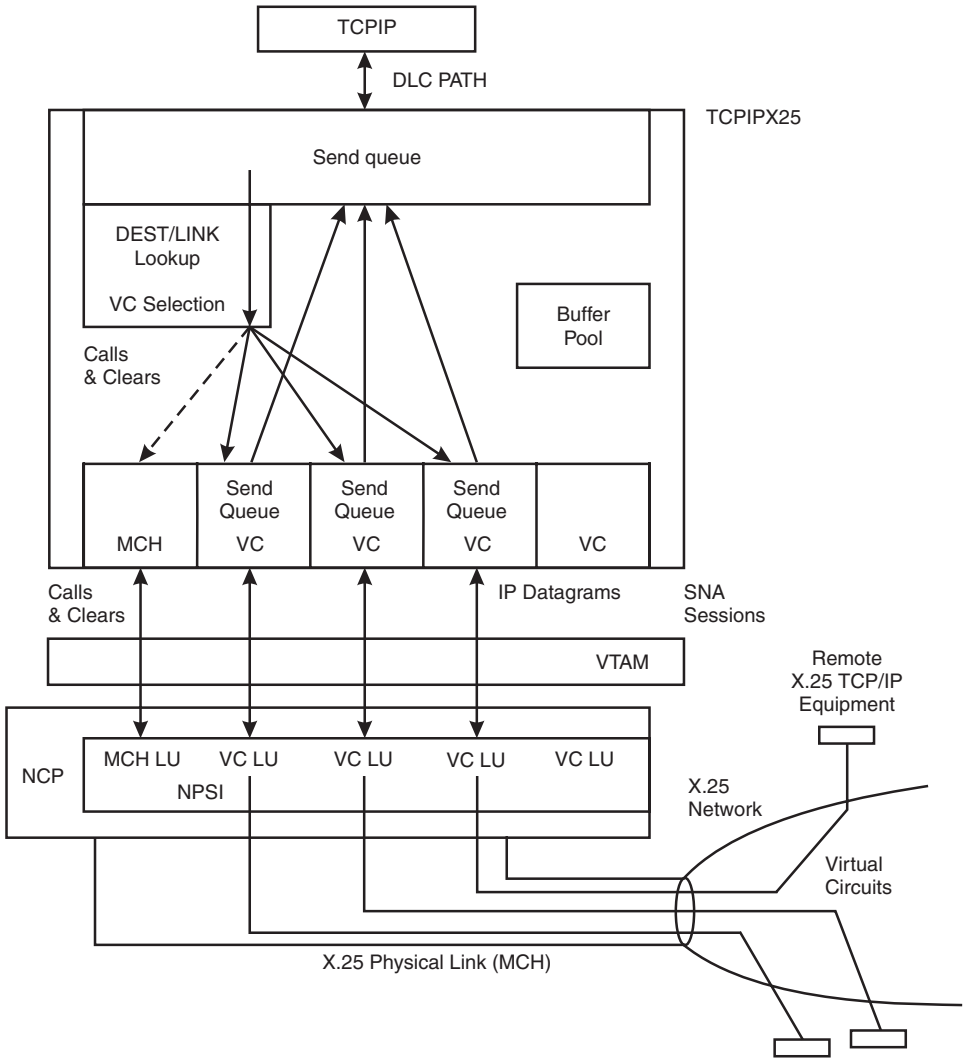


Figure 72. X.25 NPSI Environment

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 72 are:

- TCP/IP interface
- NPSI interface
- IP/X.25 address mapping

IP datagrams are transferred between TCPIP 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 TCPIP 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 TCPIP. 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. Once 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 which 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 VTAM and NPSI configuration considerations.

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 turn out to be configuration faults. Configuration files to check are:

- DEVICE, LINK, and GATEWAY entries in PROFILE.TCPIP
- The X.25 NPSI server configuration data set
- VTAM APPL definition for the X.25 NPSI server
- NPSI definitions
- VTAM SWNET definitions for NPSI

The primary diagnostic information source is the activity log produced by the X.25 NPSI server. Messages appear in the MVS system log, and can also be captured into a separate data set by including a SYSPRINT DD statement in the X.25 NPSI cataloged procedure. Normal logging records virtual circuit establishment and termination.

Additional information can be recorded about VC activity by setting the TRACE CONTROL option in the X.25 NPSI server configuration data set. This level is sufficient for almost all problem situations; interpretation of the data requires knowledge of X.25 NPSI packet formats. Tracing of the contents of IP datagrams sent to and received from NPSI is provided by the MVS CTRACE option. For details on using the CTRACE option, see “Chapter 5. TCP/IP Services Traces and IPCS Support” on page 47.

VTAM buffer traces and NPSI X.25 line traces can also be useful in diagnosing difficult problem situations.

The IP packet trace facility can be used to trace the flow of IP packets. It is useful when tracking the cause of packet loss or corruption. See “Chapter 5. TCP/IP Services Traces and IPCS Support” on page 47, for details about how to use the IP packet trace facility.

X.25 Trace Examples

The message severity codes (last position of the message ID) are:

I Information (including trace)
W Warning
E Recoverable error
S Recoverable error
T Irrecoverable error

The following example shows normal initialization:

```
EZB2111I VTAM ACB X25IPI1  opened succesfully
EZB2210I MCH XU038    packet level ready
EZB2451I IP AS path accepted for jobname 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. TCPIP establishes an IUCV 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
```

Points to note:

- The VC identifier F001XU038 ties together the events associated with a single virtual circuit. Messages for one VC will usually be mingled with messages for other VCs.
- The X.25 address originating the call (00000039) is reported in the EZB2301I message.
- 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.
- EZB2330I “call complete” indicates the virtual circuit is ready for transferring TCP/IP data.
- 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*.
- EZB2351I reports the number of IP datagrams transferred on the virtual circuit.
- 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 *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)* .

The status of the IUCV path to TCPIP is shown in the last line; 80 is normal; zero-zero (00) indicates that the TCPIP X25 NPSI device has not been started.

Termination by TCPIP STOP Device

The following examples 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 IUCV path for TCPIP.

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 478. 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 be 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 will help 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 TCPIP 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 will be indicated by a clear cause and diagnostic, as described in “Logon Problems” on page 481.

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. Using the NPSI GATE Fast Connect feature will 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 477 for details. Attempting to pass a datagram larger than MAXDATA on a virtual circuit will hang 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 28. Diagnosing IMS Problems

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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*. 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 73.

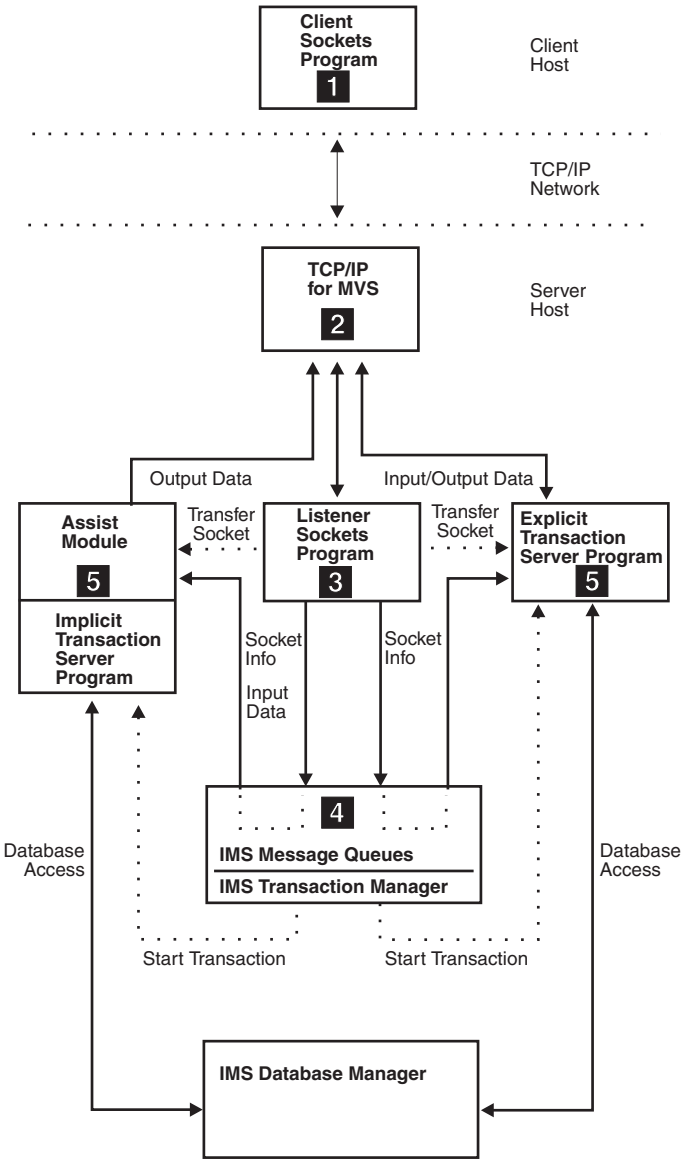


Figure 73. Components of the IMS TCP/IP Services Socket Interface System

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 from TCP/IP between the client program and the Listener.

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 is closed.

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.

Setting Up the IMS TCP/IP Services Socket Interface System

Completing the following steps establishes the system described in Figure 73 on page 485. Each step should be completed in order from the first step to the last.

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 73 on page 485. 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
   ```

   See *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* 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 *OS/390 IBM 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.

   Use the PING command on the client host, using the server host destination IP address or network name, to verify the TCP/IP network connection.
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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*.
6. **5** Write the IMS transaction that will be requested by the client program, if not already written.

   Refer to *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* for specific details about writing IMS transactions that can be requested by a TCP/IP client program.
7. Define the IMS transaction that will be 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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* for the 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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* 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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*. The JCL procedure required for starting the address space is also listed in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*.
11. Write the client program, if not already written.

Refer to *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* 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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* 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?

   Use the PING command on the client host using the same IP address or host name as specified in the client program to verify that the network to the server host is active.

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 499 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 501 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 the required definitions. See “Setting Up the IMS TCP/IP Services Socket Interface System” on page 486 for the list of definitions and configurations required.

5. Is the client program connecting to the same TCP/IP port as the Listener? See “Using NETSTAT” on page 499 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

These problems are related to starting or stopping one of the components in the IMS TCP/IP Services socket interface system.

- *The Listener terminates on startup.*

  The following is a list of possible causes and resolutions.

  Cause Incorrect configuration data set.

  Resolution
  Check for configuration error messages written to the SYSPRINT data set and correct the problems (if any).

  Cause The prerequisites for starting the Listener have not been completed.

  Resolution
  Complete the required steps listed in "Setting Up the IMS TCP/IP Services Socket Interface System" on page 486.

  Cause Incorrect method of starting.

  Resolution
  Ensure the Listener is being started as an MVS address space as described in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*. The JCL procedure required for starting the address space is also listed in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*.

- *The Listener will not terminate.*

  Cause The Listener will wait 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.

  Resolution
  Force the Listener to terminate using the command specified in the section about stopping the IMS Listener in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*.

  See "Connection Problems" on page 490 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.*

  Cause The IMS system should be started before the Listener. If the Listener is started first, the operator is prompted to start the IMS system.

  Resolution
  Reply to the console messages to start IMS.

- *An implicit IMS transaction written in C is experiencing unexpected problems at startup.*

  Cause If IMS transaction programs written in C are not built correctly, the IMS interface will fail on startup.

  Resolution
  Build the C program correctly as specified in the section about writing an IMS TCP/IP Services server program in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*.

- *The Listener is abending while accepting the TRM.*

  Cause 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 will abend because the exit runs in the same address space.

Resolution
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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*.

Connection Problems

These problems are 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.

- *The client program is experiencing intermittent reject connect responses from TCP/IP.*

  Cause The TCP/IP sockets facility has a connection request backlog queue. While this queue is full, further connection attempts will be rejected by TCP/IP. Under load, this queue can temporarily fill, causing some client program requests to be rejected.

  Resolution
  To reduce the frequency of this problem, increase the size of the backlog queue. The size of the queue is a parameter in the Listener configuration data set.

- *The TCP/IP socket connection to the client program is being broken immediately after the implicit IMS transaction is scheduled.*

  Cause The Listener configuration data set might incorrectly define the implicit IMS transaction as explicit. In this case, the Listener will not pass the input data to the IMS transaction through the message queue as expected. The transaction will start, and upon detecting no data, immediately close the TCP/IP socket connection and terminate.

  Resolution
  Verify that the TRANSACTION statements in the Listener configuration data set specify the TYPE parameter correctly.

- *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.

  Cause 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 will instruct 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.

  Resolution
  Use the IP packet trace facility to determine if the client program is sending a valid EOM segment. See “Using IP Packet Trace” on page 498 for details about the IP packet trace facility.

  Refer to the information about implicit-mode application data in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* 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.

Cause 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 will wait indefinitely.

Resolution
Use the IP packet trace facility to identify which part of the protocol is failing. See “Using IP Packet Trace” on page 498 for details about the IP packet trace facility.

Cause The Listener configuration data set might incorrectly define the explicit IMS transaction as implicit. In this case the Listener will wait for valid implicit data from the client program, or if valid data is received, the explicit IMS transaction will wait for data from the client program because the Listener has already read the data and written it to the message queue.

Resolution
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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* for more information about specifying timeouts for READs.

- *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.

  Cause If the TRM sent by the client program is incomplete, the Listener will wait indefinitely for the rest of the message.

  Resolution
  Check the length and format of the TRM by using the IP packet trace facility as described in “Using IP Packet Trace” on page 498.

  Cause If the IMS transaction does not successfully issue the takesocket to gain the connection from the Listener, the Listener will wait 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 will never be 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.

  Resolution
  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.

- *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.

Cause 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 will fail as the socket connection has already been taken from the Listener.

Resolution
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 499.

Cause An IMS transaction not defined as multisegment to the IMS transaction manager will be 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 will fail with an error return code.

Resolution
Make certain the IMS transaction is defined as multi-segment.

- *The client program is always receiving reject connect responses from TCP/IP.*

  Cause 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 will eventually reduce 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 will reject a client program connection attempt.

  Resolution
  Identify the client programs causing the problem using the NETSTAT command as specified in "Using NETSTAT" on page 499. 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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*.

- *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 will never complete.

  Cause 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 will be 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 will wait indefinitely for the takesocket from the second IMS transaction, which will never be scheduled.

Resolution
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

These problems are related to error responses.

- *The client program is receiving a request status message (RSM).*

  Cause The Listener will send this message to the client program when it detects an error condition.

  Resolution
  Use the return and reason codes from the message to look up the explanation. See “Where to Find Return Code Documentation” on page 500.

- *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/ESA Diagnosis Guide and Reference* .

  Cause The IMS assist module will perform 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.

  Resolution
  Look up the meaning of the special return codes. See “Where to Find Return Code Documentation” on page 500 .

- *The Listener error messages are written to the MVS system console instead of the SYSPRINT data set.*

  Cause If the Listener experiences data set I/O errors, it will redirect the error messages to the MVS system console.

  Resolution
  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

These problems are related to data transfer over the socket connection. They include incorrect data sent, not enough or too much data sent, and data corruption.

- *The Listener is not responding to the client program.*

  Cause If the TRM sent by the client program is incomplete, the Listener will wait indefinitely for the rest of the message.

  Resolution
  Check the length and format of the TRM by using the IP packet trace facility as described in “Using IP Packet Trace” on page 498.

  Cause 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 will not communicate with the client program as required.

Resolution

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 499.

- *All the input data sent from the client program is not being passed to the implicit IMS transaction from the Listener.*

  Cause Any input data written after the first EOM segment will be ignored by the Listener.

  Resolution

  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 498.

  Refer to the information about the implicit-mode application data in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* for the format of the EOM segment.

- *Explicit IMS transaction is receiving garbled data from or sending garbled data to the client program.*

  Cause 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.

  Resolution

  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.

- *Implicit IMS transaction is receiving garbled data from or sending garbled data to the client program.*

  Cause 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 will be 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 will be translated to spaces.

  Resolution

  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 will not validate user data from the client program.*

Cause 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.

Resolution
Check that the security exit has been coded and built correctly as specified in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*.

- *Data is corrupted after an implicit IMS transaction issues a GU.*

  Cause 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.

  Resolution
  Make certain the implicit IMS transaction has enough storage declared to hold the TIM. The size of this message is specified in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* .

- *The PL/I IMS transaction is receiving or sending message segments that are not valid.*

  Cause The message segments might be declared incorrectly. The PL/I API interface to the IMS transaction manager defines the message segments with a 4-byte length field, but the length value must only include two of those bytes plus the rest of the segment.

  Resolution
  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 will automatically convert the message segments from the PL/I API to the standard format.
  - Explicit transactions don't 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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* for more details about the PL/I API issues.

IMS Transaction Build Problems

This problem relates to building a component in the IMS TCP/IP Services socket interface system.

Unresolved external reference errors are causing the linker to fail when linking an IMS transaction.

Cause 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.

Resolution

Compare the link JCL to the sample provided in the section about JCL for linking an implicit-mode server in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*.

Cause The explicit IMS transaction link JCL is not including the MVS TCP/IP Services sockets library to resolve external references.

Resolution

Compare the link JCL to the sample provided in the section about JCL for linking an explicit-mode server in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*.

IMS Database Problems

These problems relate to unexpected IMS database actions or failures. They include changes not made or requests for changes that fail.

- *The IMS transaction is terminating without performing the required function and without issuing any error messages.*

  Cause The IMS transaction might not be checking for interface errors.

  Resolution

  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.*

  Cause 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).

  Resolution

  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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference*.

- *Local IMS transaction manager ISRT/GU/GN calls are failing when issued in IMS transactions.*

  Cause 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 will receive an unexpected TIM in response to a GU call.

  Resolution

  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 will access 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.*

  Cause The IMS assist module restricts the total output for a single IMS transaction execution to 32KB.

  Resolution
  Limit the output for an implicit IMS transaction using the IMS assist module to a total of 32KB.

- *The IMS database manager commits the changes made by an IMS transaction, but the client program receives an error.*

  Cause 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.

  Resolution
  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.

  Cause The socket connection might have been broken after the changes were committed, but before the CSM was sent. In this case, the client program will assume failure, but the changes have been committed.

  Resolution
  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.

  Note: This should be considered as an uncommon case.

- *The client program does not receive a valid CSM from an implicit IMS transaction.*

  Cause 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.

  Resolution
  Use the IP packet trace facility to determine if the IMS transaction is sending a valid EOM segment followed by a valid CSM segment. See “Using IP Packet Trace” on page 498 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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* 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.

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.

Note: 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 TCPIP address space.

See Chapter 5. TCP/IP Services Traces and IPCS Support 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 TCPIP internal traces are sent to CTRACE. These traces provide information about the internals of the TCPIP 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, for details about how to use the TCPIP 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/ESA Administration Guide: System*. For details about tracing and reading the trace reports refer to *IMS/ESA 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.

Note: 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:

```
 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 *OS/390 IBM Communications Server: IP User's Guide* 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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* for the format of the RSM and a description of the return codes.

  Note: 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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* 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 call in *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* .

- 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 *IMS/ESA 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 *IMS/ESA 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 *TCP/IP for MVS: IMS TCP/IP Application Development Guide and Reference* 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 *OS/390 IBM Communications Server: IP Messages Volume 1 (EZA)* for descriptions of the error messages in these data sets.

Chapter 29. Diagnosing Restartable VMCF/TNF Problems

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 on configuration, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.) If you choose restartable VMCF and TNF, you may encounter the following problems.

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.

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. To get them to respond, follow these steps:

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 Will 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 30. Diagnosing Problems with CICS

This chapter describes how to diagnose problems with the Customer Information Control System (CICS). 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 may be helpful in solving problems with CICS, refer to the following manuals:

- *OS/390 IBM Communications Server: IP CICS Sockets Guide*
- *CICS/ESA 5.2 Diagnosis Reference*
- *CICS/ESA 5.2 Problem Determination Guide*
- *CICS/ESA 5.2 Messages and Codes*
- *CICS/ESA 5.2 Data Areas*
- *CICS/ESA 5.2 Supplementary Data Areas*
- *OS/390 MVS Diagnosis: Tools and Service Aids*
- *CICS/ESA 5.2 Operations Guide*

Diagnostic Data

To diagnose problems with CICS, some or all of the following data may be required:

- Message logs
  - System log
  - Message log at the transient-data destination specified by ERRORTD
- CICS external-trace data set (auxtrace)

  Note: To obtain the CICS auxtrace, set the CICS user master-trace flag, the system master-trace flag, and the AP level 1 standard trace component.
- Component trace
  - Engine
  - Physical file system (PFS)
  - Socket
  - Transmission control protocol (TCP)
- Dumps
  - CICS transaction dump, if captured.
  - Supervisor Call (SVC) dump. SVC dumps are also known as *console dumps* or *system dumps*.

    Note: 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 may encounter when attempting to initialize CICS.

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 will allow the CICS Sockets Interface to initialize on CICS address startup. Refer to the *OS/390 IBM 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, start CICS with auxtrace active, 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.

CICS Listener Not Initialized

If the CICS listener did not initialize, follow the steps below:

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, and then request an SVC dump of CICS.
   b. Call the Support Center.
5. If an EZY1292I message was issued, investigate why the CICS sockets interface did not initialize. (See “CICS Sockets Interface Not Initialized”.)

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 ERRORTD field in the configuration record for the CICS region. A *region* is the CICS address space.

TCP/IP Clients Unable to Connect

If TCP/IP clients are unable to connect, follow these 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.

Child Server Transactions Not Starting

Child-server transactions are transactions started by the listener. If child-server transactions are not starting, follow these 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 may encounter with CICS sockets applications.

Hung CICS Tasks

If CICS application tasks hang, follow these 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 recreated, recreate 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 command prior to the READ command. The SELECT command returns either the number of sockets ready to be read or 0 (zero) if it times out.

Hung CICS Region

If a CICS sockets application program that is written in COBOL is erroneously link-edited without the EZACICAL stub, and then calls to the socket API will not go through the CICS sockets task related user exit (TRUE). (A *stub* is a fragment of code that is link-edited with the application load module and that is called with EZASOKET or EZACICAL calls.) When this happens, the entire CICS region may go into a hang, waiting for the socket call to complete.

Note: While all CICS sockets programs need the EZACICAL stub, the hang can only occur with programs written in COBOL.

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 *OS/390 IBM Communications Server: IP CICS Sockets Guide* that describes return codes.

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 attempt to terminate the CICS sockets interface, and then add EZACIC20 to the shutdown DFHPLT. If you do not add EZACIC20, 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 may 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 *OS/390 IBM Communications Server: IP CICS Sockets Guide* that describes external data structures.

Task Interface Element

A 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 'UEH=3' command, and searching for EZACIC01.TIE. The EZACIC01 prefix identifies it as a TIE for CICS sockets.

The UEH=3 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'.

Note: The UEH=3 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 command VERBX 'UEH=3', 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.

Trace records can be written either to CICS internal trace table or to its external-trace data set (auxtrace). 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 in TSO. Examples of options are CSA, PSA, NVC, RGN, TRT, SQA, LSQA, LPA, and so on. For a complete list of options, refer to *OS/390 MVS Diagnosis: Tools and Service Aids*.
2. Display the trace using the VERBX 'TR=2' IPCS command.

   Note: 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/ESA 5.2 Operations Guide*.

Part 4. Appendixes

Appendix A. Collecting Component Trace Data

This appendix provides short descriptions of the following tracing procedures:

Modifying Options with the TRACE CT Command

After initialization, you must use the TRACE CT command to change the component trace options for TCP/IP stacks, OMPROUTE, and packet trace. All options except for the size of the TCP/IP component trace buffer can be changed using this command. Modifying options with the TRACE CT command can be done with or without the PARMLIB member.

With PARMLIB Member

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:

- SYSTCPIP for TCP/IP stacks
- SYSTCPDA for packet trace and data trace
- SYSTCPRT for OMPROUTE

SUB Indicates the started procedure name for TCP/IP or the OMPROUTE application for which the trace is run. If you use the *S procname.jobname* method of starting TCP/IP or OMPROUTE, 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 active in one system. Only one OMPROUTE application can be active on each TCP/IP stack.

PARM Identifies the parmlib member containing the trace options (CTIEZB*xx* for TCP/IP stacks and CTIORA*xx* for OMPROUTE). All options except the size of trace buffers can be respecified. This size cannot be changed during the execution of TCP/IP or OMPROUTE. If a different size is required, you must stop TCP/IP or OMPROUTE, and then restart it after modifying the parmlib member.

If the incorrect parmlib member is specified, one of the following messages may 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 CTIEZBZZ 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
  ```

Without PARMLIB Member

To change component trace options without using a PARMLIB member, issue the TRACE CT command without the SUB= parameter and specify the options on the reply. Use the following syntax:

```
TRACE CT,ON,COMP=<component_name>,SUB=(procedure_jobname)
```

After issuing the TRACE CT command, you will be 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]
```

Note: 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 JOBNAME of the client whose TCP/IP requests are to be traced. The jobname may be:
  - The jobname associated with a client application.
  - The SNA LU associated with a TELNET session.
  - The FTP Userid associated with a FTP data connection.

OPTIONS
: Options valid for use with SYSTCPIP are listed in “Chapter 5. TCP/IP Services Traces and IPCS Support” on page 47and options valid for use with OMPROUTE are listed in “Chapter 25. Diagnosing OMPROUTE Problems” on page 435.

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 “Obtaining Component Trace Data with an External Writer” on page 516.)

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)
```

where *component_name* can be one of the following:

- SYSTCPIP for TCP/IP stacks
- SYSTCPDA for packet trace
- SYSTCPRT for OMPROUTE

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, enter the following command:

```
DISPLAY TRACE,COMP=<component_name>,SUBLEVEL,N=8
```

Stopping a Component Trace

You can stop current tracing with the TRACE CT command:

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

where the component name can be any of the following:

- SYSTCPIP for TCP/IP stacks
- SYSTCPDA for packet trace
- SYSTCPRT for OMPROUTE

TCP/IP always maintains exception tracing to aid in first failure data capture.

Obtaining Component Trace Data with a Dump

You can request a dump to obtain component trace data for OMPROUTE or a TCP/IP stack.

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 table or the data set (or sets) produced by the external writer. See "Formatting Component Traces" on page 518.

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,rgn,csa,sqa),END
```

To generate a meaningful dump, specify at least *csa* and *sqa*.

Display trace data using IPCS component trace formatting. For details, see "Formatting Component Traces" on page 518.

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_procedure_jobname>,SDATA=(rgn,csa,sqa),END
```

View the trace data contained in the dump using IPCS component trace formatting. For details, see “Formatting Component Traces” on page 518.

Obtaining Component Trace Data with an External Writer

To use an external writer to obtain component trace data for TCP/IP stacks, packet trace, and OMPROUTE, follow the steps below.

1. Enter the appropriate writer procedure in SYS1.PROCLIB, as shown in the following example. (CTTCP starts the component trace writer; use PTTCP to start the packet trace writer.) You can have multiple procedures writing to as many as 16 TRCOUT files either on disk or tape.

   ```
   //CTTCP  PROC
   //* REFER:  SYS1.PROCLIB(CTTCP)
   //* COMPID: OPER
   //* DOC:    THIS PROCEDURE IS THE IPCS CTRACE1 EXTERNAL WRITER PROCEDURE.
   //*         USED BY TCP/IP   .
   //*
   //IEFPROC  EXEC  PGM=ITTTRCWR
   //TRCOUT01 DD DSNAME=MEGA.IPCS.CTRACE1,UNIT=SYSDA,
   //             VOL=SER=STORGE,
   //             SPACE=(4096,(100,10),,CONTIG),DISP=(NEW,CATLG),DSORG=PS
   //
   ```

2. Start the external writer using the following command:

   ```
   TRACE CT,WTRSTART=<procedure_name>
   ```

3. Do one of the following:
   - To turn the trace on and connect the external writer for packet trace and OMPROUTE, enter the following command:

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

     where the component name can be SYSTCPDA (packet trace) or SYSTCPRT (OMPROUTE).
   - To turn the trace on and connect the external writer for TCP/IP stacks **during** stack initialization, add the following TRACEOPTS option to the CTIEZB*xx* member:

     ```
     WTR(CTxxx)
     ```

     where CT*xxx* is the procedure name of the external writer. Then restart the TCP/IP stack if it is running.
   - To turn the trace on and connect the external writer for TCP/IP stacks **after** stack initialization, enter the following command:

     TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcp_proc_name)
4. When the system responds, do one of the following:
   - If no changes to the default options are required, enter the following command:

     r nnnn,WTR=<procedure_name>,END

     where *nnnn* is the response number issued by the system.

- If specific options are desired, enter the following command:

  ```
  r nnnn,Options=(A1,A2...An),WTR=<procedure_name>,end
  ```

  where *nnnn* is the response number issued by the system and (A1,A2...An) is the list of desired trace options.

5. Use the DISPLAY command to check the external writer status. Include a sublevel if you are working with the packet trace component (SYSTCPDA):

   ```
   display trace,comp=systcpda,sublevel

   IEE843I 10.08.17 TRACE DISPLAY 142
           SYSTEM STATUS INFORMATION
   ST=(ON,0256k,00256k) AS=ON   BR=OFF   EX=ON   MT=(ON,016K)

     TRACENAME
     =========
     SYSTCPDA

                            MODE BUFFER HEAD SUBS
                            =====================
                            OFF         HEAD   1

     NO HEAD OPTIONS

   SUBTRACE                 MODE BUFFER HEAD SUBS
   ---------------------------------------------
   TCPIP                    ON    0128K
      ASIDS      *NONE*
      JOBNAMES   *NONE*
      OPTIONS    MINIMUM
      WRITER     PTTCP
   ```

   Note: At this point, the external writer is active for packet, data, and X.25 packet traces.

6. 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
   ```

7. Stops the external writer using the following command:

   ```
   TRACE CT,WTRSTOP=<procedure_name>
   ```

You can use the VARY TCPIP,,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 had started the address space TCPIPA and created a sequential dataset USER99.TCPIP.OBEYFIL1 containing packet trace statements, issue the following command:

```
VARY TCPIP,TCPIPA,CMD=OBEYFILE,DSN=USER99.TCPIP.OBEYFIL1
```

To start the external writer, reply using the following syntax:

```
TRACE CT,WTRSTART=PTTCP,WRAP
```

A message is displayed informing you that the external writer has been started.

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 formatter can be found in the *hlq*.SEZAMIG data set. This data set should be added as a concatenation to the STEPLIB data set. For details, refer to the *OS/390 MVS IPCS Commands* and the *OS/390 MVS IPCS User's Guide*.

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 from the option list.
4. Select option 7 from the option list.
5. Select option 1 from the option list.
6. Select option D from the option list.

The CTRACE DISPLAY PARAMETERS screen is displayed (Figure 74):

```
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 74. 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:

- SYSTCPIP: See "Chapter 5. TCP/IP Services Traces and IPCS Support" on page 47.
- SYSTCPDA: See "Chapter 5. TCP/IP Services Traces and IPCS Support" on page 47.
- SYSTCPRT: See "TCP/IP Services Component Trace for OMPROUTE" on page 447.

CTRACE Command

To format component traces using the CTRACE command, follow these steps:

1. Log on to TSO.

2. Access IPCS.
3. Select option 6 from the option list.
4. Enter a CTRACE command and options on the IPCS command line.

Tips for Using Component Trace

- Do not use the same writer to trace more than one TCP/IP stack 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 will help ensure there is enough virutal memory for trace buffers.
- Use CONTIG on the disk space allocation of the trace data when using the WRAP option. For example: SPACE=(1024,(4096,100),,CONTIG). This will ensure that the space for the trace data set is available.

Appendix B. Search Paths

The OS/390 UNIX function _ipdspx() retrieves the TCPIP.DATA keyword DATASETPREFIX. The function _iptcpn() retrieves the TCPIP.DATA keyword TCPIPjobname. To locate the TCPIP.DATA configuration file with these keywords, _ipdspx() and _iptcpn() use a search path beginning with

```
ENVIRONMENT VARIABLE "RESOLVER_CONFIG=dataset/file"
/etc/resolv.conf
```

The functions _ipdspx() and _iptcpn() stop searching at the first file they find, regardless whether the file contains DATASETPREFIX or TCPIPjobname.

For example, consider the following scenario:

1. ENVIRONMENT VARIABLE “RESOLVER_CONFIG=dataset/file” is not set.
2. File /etc/resolv.conf exists but does not contain either a DATASETPREFIX or TCPIPjobname keyword.
3. TCPIP.TCPIP.DATA exists and has a TCPIPjobname of TCPIPA.

If the SNMP agent is started and calls the function _iptcpn(), the function will return a null string because the search stopped with /etc/resolv.conf.

This problem can be resolved one of the following ways:

- If running multiple stacks, specify RESOLVER_CONFIG to point to a resolver file or data set, or
- If running only one stack, rename /etc/resolv.conf or update it with the required information.

For more information on search paths, refer to *OS/390 IBM Communications Server: IP Configuration Reference*.

Appendix C. First Failure Support Technology (FFST)

This appendix contains the following sections:

- FFST probe index
- FFST probe information
- FFST probe naming convention
- FFST probe descriptions

FFST Probe Index

Table 44 provides an index of FFST probes by probe name and component:

Table 44. 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 will continue 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 supply the service representative with the console listing that is written at the time the error and the dump data produced by the probe.

FFST Probe Naming Conventions

Table 45 lists the naming conventions for FFST probes used in TCP/IP.

Table 45. 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 44 on page 523 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 46 lists the FFST probes for IOCTL enablement (EZBIECxx).

Table 46. 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

Table 47 lists the FFST probes for Infrastructure Services (EZBITCxx).

Table 47. 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
EZBITC10	EZBPTDEF	Pat tree key too big	FULL
EZBITC11	EZBPTADD	Pat tree key exists	FULL

Table 47. Infrastructure Services Probes (continued)

Probe Name	Module	Description	Dump Type
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 48 lists the FFST probes for Pascal API (EZBPACxx).

Table 48. 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
EZBPAC15	EZBPAROP	Streams operation software failure	FULL

Table 48. FFST Probes for Pascal API (continued)

Probe Name	Module	Description	Dump Type
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
EZBPAC41	EZBPASTR	Streams operation software failure	FULL

Table 48. FFST Probes for Pascal API (continued)

Probe Name	Module	Description	Dump Type
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
EZBPAC66	EZBPAUOP	TPI protocol error	FULL

Table 48. FFST Probes for Pascal API (continued)

Probe Name	Module	Description	Dump Type
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 48. 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 49 lists the FFST probes for PFS IOCTL (EZBPFCxx).

Table 49. PFS IOCTL Probes

Probe Name	Module	Description	Dump Type
EZBPFC01	EZBPFIOC	SIOCSETTKN mismatch	FULL
EZBPFC02	EZBPFIOC	SIOCSETTKN mismatch	FULL

Table 50 lists the FFST probes for Telnet Transform (EZBTRCxx).

Table 50. 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 51 lists the FFST probes for Telnet SRV (EZBTTCxx).

Table 51. 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 52 lists FFST probes for Configuration Services (EZBCFCxx).

Table 52. 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 53 lists the FFST probe for TCP/IP Base (EZBABCxx).

Table 53. TCP/IP Base Probes

Probe Name	Module	Description	Dump Type
EZBABC01	EZBCABND	A C abend recovery failed	FULL

Table 54 on page 531 lists the FFST probes for Transmission Control Protocol (EZBTCCxx).

Table 54. 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 55 lists the FFST probes for Update Datagram Protocol Layer (EZBUDCxx).

Table 55. 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 56 lists the FFST probes for Streams (EZBSKCxx).

Table 56. 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 57 lists the FFST probes for Raw IP Layer (EZBRWCxx).

Table 57. 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 58 lists the FFST probes for Internet Protocol (EZBIPCxx).

Table 58. FFST Probes for Internet Protocol

Probe Name	Module	Description	Dump Type
EZBIPC01	EZBIPSTR	Not a Clone Open	FULL

Table 59 lists the FFST probes for the Cross-System Coupling Facility (XCF) (EZBXFCxx).

Table 59. 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

Appendix D. Overview of Internetworking

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.

Note: If IP datagrams are not passed properly over a bridge, none of the higher TCP/IP protocols or applications will 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.

Note: A bridge does ***not*** connect TCP/IP networks together. It connects physical networks together that will still form the same TCP/IP network. (A bridge does ***not*** do IP routing.)

Figure 75 on page 534 depicts a router and a bridge. The router connects Network 1 to Network 2 to form an intranet.

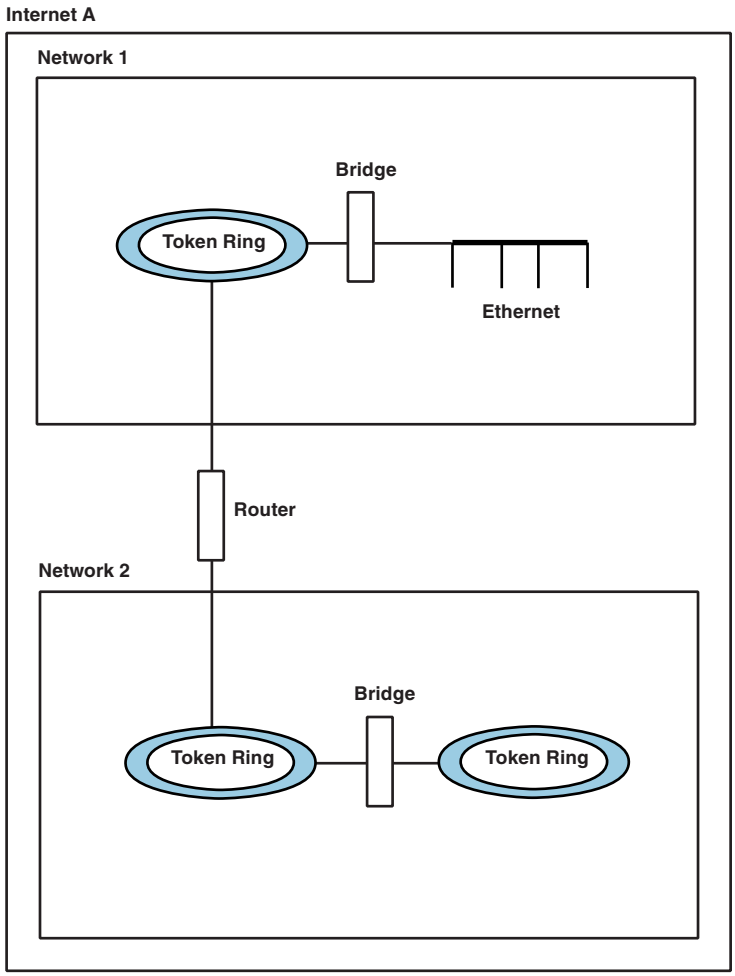


Figure 75. Routers and Bridges within an Internet

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 76 on page 535 shows the relationship between MTU and frame size.

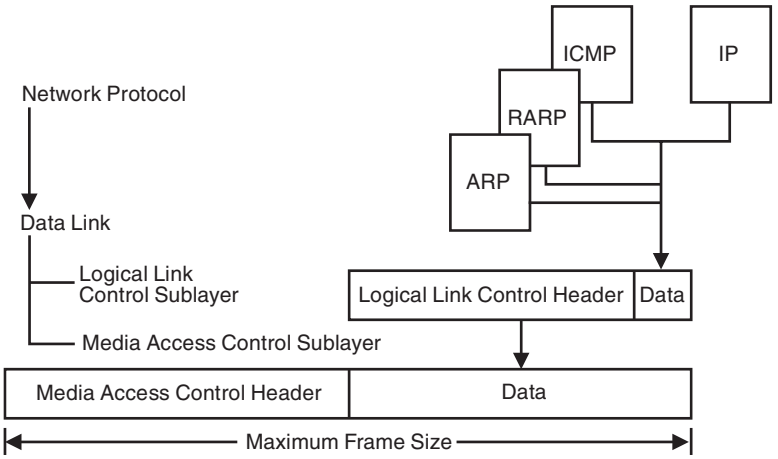


Figure 76. Relationship of MTU to Frame Size

If an IP datagram is to be sent out onto the network and the size of the datagram is bigger than the MTU, IP will fragment the datagram into multiple fragments, so that it will fit 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. For this reason, use an MTU of 576 bytes if bridging (or routing) problems are suspected.

Note: MTU is equivalent to the MAX_PACKET_SIZE value on the GATEWAY statement, or the MTU value specified on BSDROUTINGPARMS when TRUE.

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, please 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 77 for details on the relationship of the I-field to the header fields.

Note: 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 will fail.

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 will support. This value is specified in the Routing Control (RC) field within the RIF as shown in Figure 77.

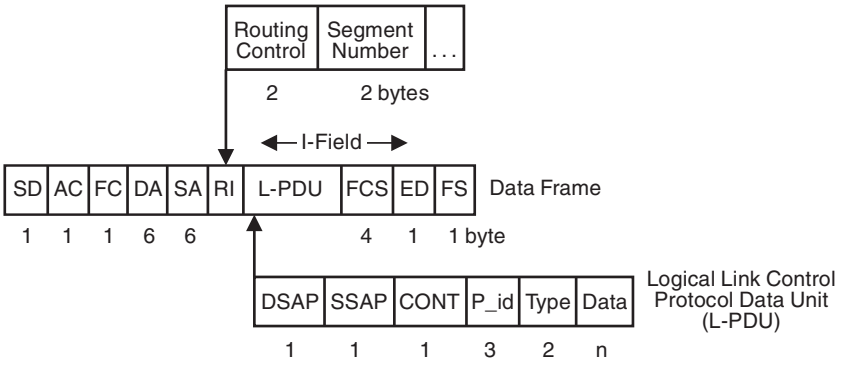


Figure 77. Format of an IEEE 802.5 Token-Ring Frame

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 is determined by the bit configuration in the RC field, and is present in all routed frames.

Table 60 on page 537 shows the relationship between the RC field and the maximum I-field values.

Table 60. 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

In Figure 77 on page 536 we can see 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 538. 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 will be 2052 bytes, and so the MTU would be set to 2044 bytes (2052 minus 8 bytes for the LLC header).

IEEE 802.3

The frame used in IEEE 802.3 Ethernet networks is shown in Figure 78.

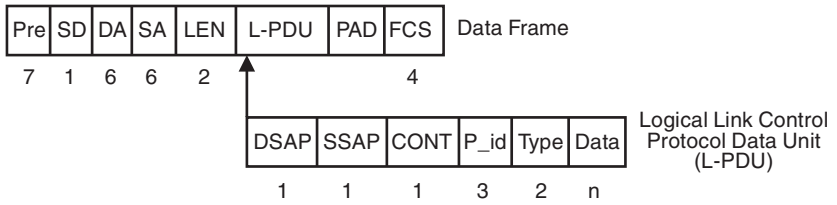


Figure 78. Format of an IEEE 802.3 Frame

The maximum size of the L-PDU for a 10Mbps network is 1500 bytes. Because 8 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 79 on page 538.

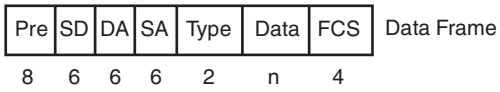


Figure 79. Format of an Ethernet V2 Frame

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; a 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 80.

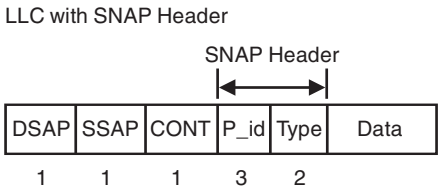


Figure 80. SNAP Header

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'08 00' — IP
X'08 06' — ARP
X'08 35' — RARP

IP Routing

IP routing is based on routing tables held within a router or internet host. These tables 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 Addressing

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 IP 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 IP 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 IP address that is used for each of these addresses is determined by the class of address. There are three commonly used classes of IP address (see Figure 81).

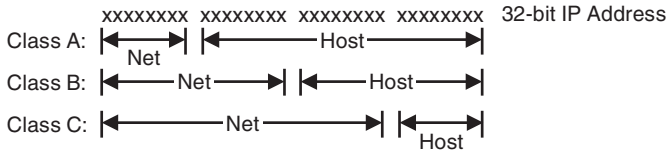


Figure 81. Classes of IP Addresses

The class of the address is determined by the first octet of the IP address. Figure 82 on page 540 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 82. Determining the Class of an IP Address

As shown in Figure 82, 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 IP address. For example, the IP 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 IP address is “9” and the host part of the IP address is “67.97.2”.

Refer to *RFC 1166 — Internet Numbers* for more information about IP addresses. Refer to *RFC 1060 — Assigned Numbers* for more information about reserved network and host IP addresses, such as a *network broadcast address*.

Figure 83 on page 541 shows a simple network with a bridge and a router.

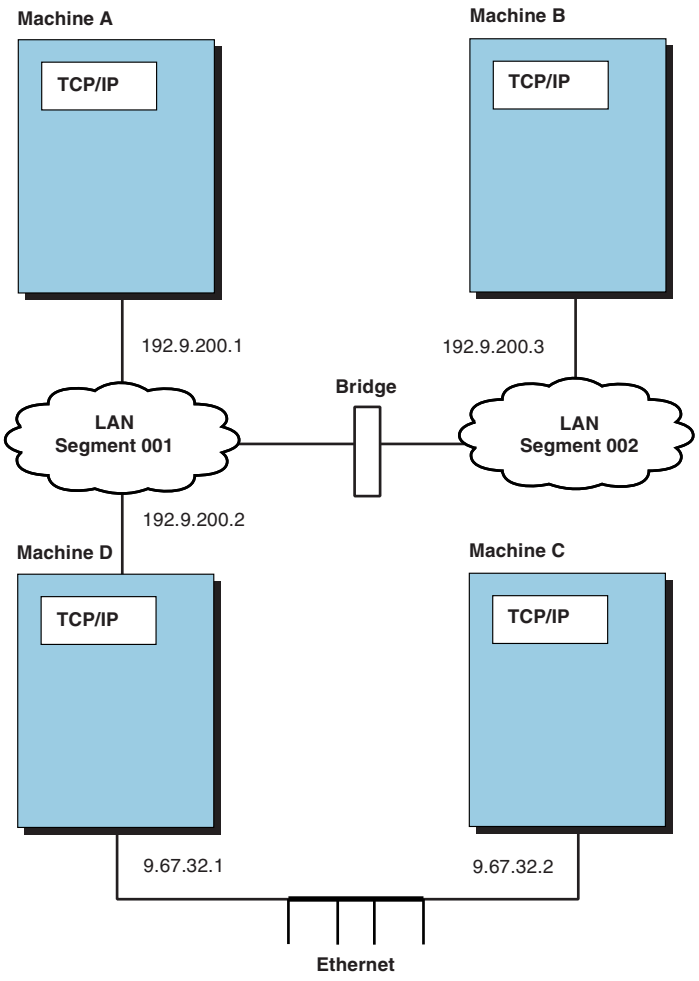


Figure 83. Routing and Bridging

Machine D is acting as an IP router and will transfer 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 *OS/390 IBM Communications Server: IP Configuration Reference* for more information about both the HOME statements and the VARY TCPIP command.

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 83 on page 541, 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 83 on page 541, 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 84 is used.

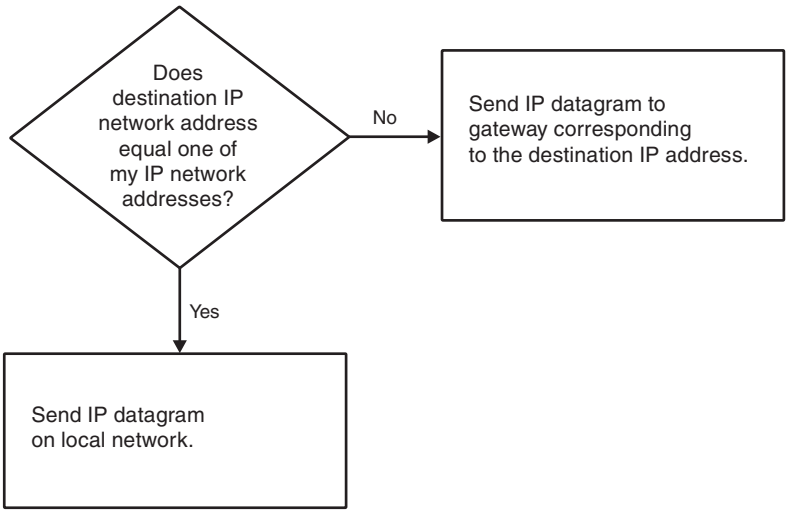


Figure 84. General IP Routing Algorithm

Using this general routing algorithm, it is very easy to determine where an IP datagram will be routed. Following is a simple example based on the configuration shown in Figure 83 on page 541.

```
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.

Subnetting

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, let us 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 85:

```
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 85. Subnetting Scheme

OS/390 TCP/IP uses a slightly different scheme for the *subnet mask* when defining the GATEWAY 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 OS/390 TCP/IP would be:

```
   0        0        0       192       0.0.0.192
00000000 00000000 00000000 11000000
```

Although OS/390 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.

Simplified IP Datagram Routing Algorithm with Subnets

When subnetting is used, the algorithm 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 86 shows the routing algorithm used with subnets.

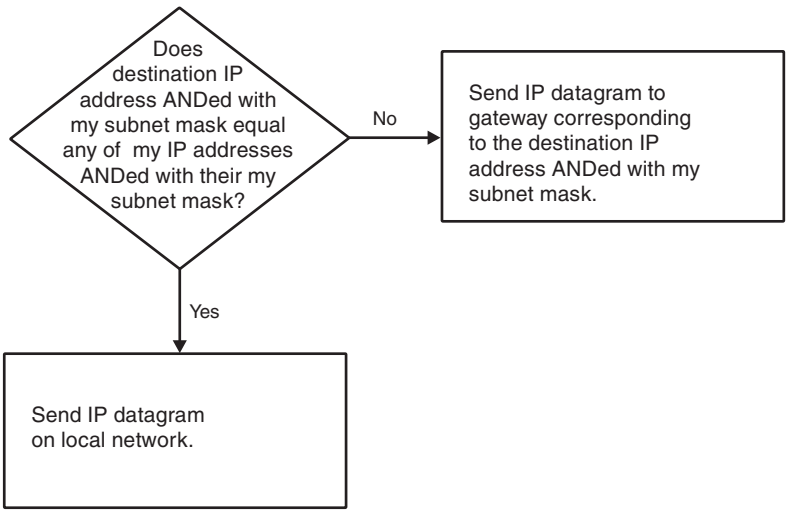


Figure 86. Routing Algorithm with Subnets

Figure 87 on page 545 shows how a subnet route is resolved.

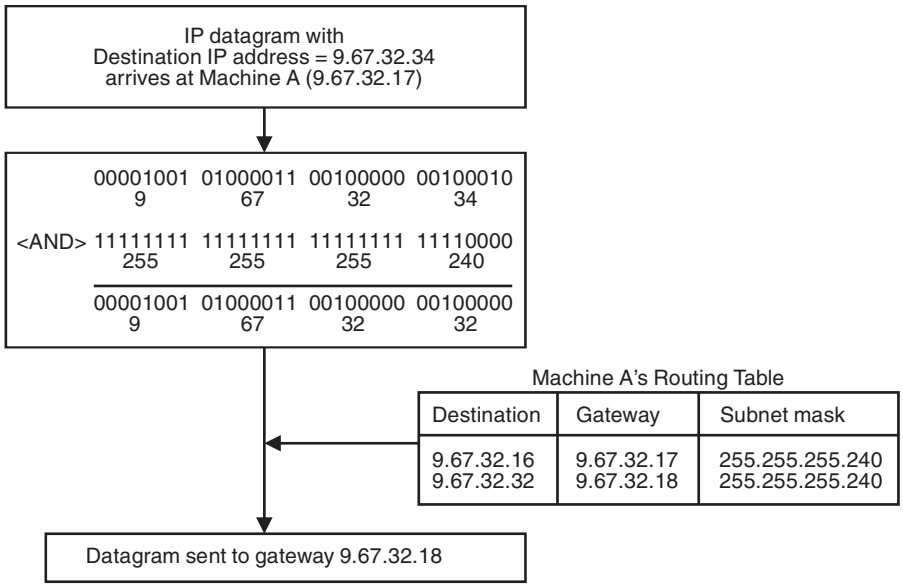


Figure 87. Example of Resolving a Subnet Route

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 will contain 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 OS/390 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 OS/390 TCP/IP can be modified by either: 1) changing the GATEWAY or BEGINROUTES statements and recycling the TCP/IP address space or 2) using the VARY TCPIP command. Refer to the *OS/390 IBM Communications Server: IP Configuration Reference* for details about defining the GATEWAY or BEGINROUTES statements and using the VARY command.

Note: When the GATEWAY or BEGINROUTES statements are updated using VARY TCPIP, all previously-defined 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. One routing protocol is the Routing Information Protocol (RIP). It is implemented by both the OROUTED and 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 25. Diagnosing OMPROUTE Problems” on page 435. For more details about OROUTED, see “Chapter 24. Diagnosing

OROUTED Problems" on page 417. For configuration information about both applications, refer to the *OS/390 IBM Communications Server: IP Configuration Reference*.

Appendix E. How to Read a Syntax Diagram

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:

►► Marks the beginning of the command syntax.

► Indicates that the command syntax is continued.

I 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.

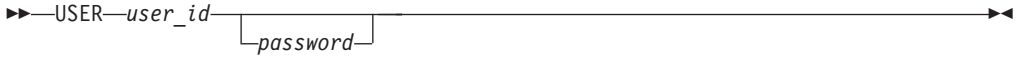
Default
Default parameters are displayed above the main path.

Parameters are classified as keywords or variables. Keywords are displayed in uppercase letters and can be entered in uppercase or lowercase. For example, a command name is a keyword.

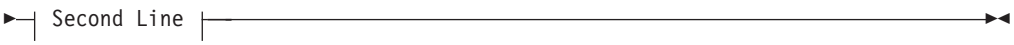
Variables are italicized, appear in lowercase letters, and represent names or values you supply. For example, a data set is a variable.

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.



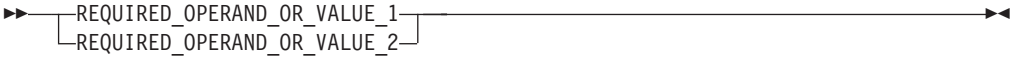
Longer than one line: If a diagram is longer than one line, the first line ends with a single arrowhead and the second line begins with a single arrowhead.



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 in alphanumeric order.



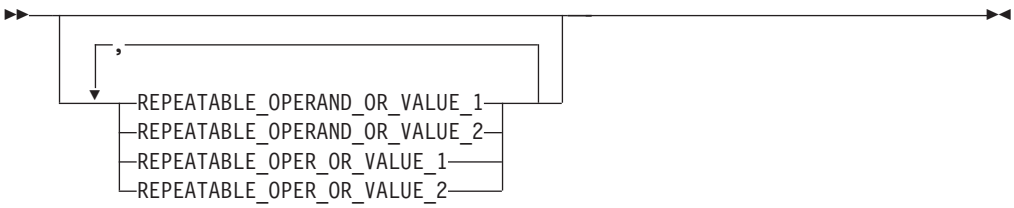
Optional values: Optional operands and values appear below the main path line.

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 in alphanumeric order 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 command means that each operand or value must be separated from the next by a comma.

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.



If an operand or value can be abbreviated, the abbreviation is described in the text associated with the syntax diagram.

Case Sensitivity: TCP/IP commands are not case sensitive. You can code them in uppercase or lowercase.

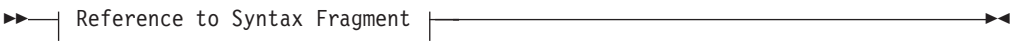
Nonalphanumeric characters: If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code OPERAND=(001,0.001).

Blank spaces in syntax diagrams: If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code OPERAND=(001 FIXED).

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

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

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



Syntax Fragment:

References to syntax notes appear as numbers enclosed in parentheses above the line. Do not code the parentheses or the number.

Notes:

1 An example of a syntax note.

Appendix F. Information Apars

This appendix lists information apars for IP-related books.

Notes:

1. Information apars contain updates to previous editions of the manuals listed below. Books updated for V2R10 contain all the updates except those contained in the information apars that may be issued after V2R10 books went to press.
2. Information apars are predefined for CS for OS/390 V2R10 and may not contain updates.

IP Information Apars

Table 61 lists information apars for IP-related books.

Table 61. IP Information Apars

Title	CS for OS/390 2.10	CS for OS/390 2.8	CS for OS/390 2.7	CS for OS/390 2.6	CS for OS/390 2.5	TCP/IP 3.3
High Speed Access Service User's Guide (GC31-8676)		ii11629	ii11566	ii11412	ii11181	
IP API Guide (SC31-8516)	II12371	ii11635	ii11558	ii11405	ii11144	
IP CICS Sockets Guide (SC31-8518)		ii11626	ii11559	ii11406	ii11145	
IP Configuration (SC31-8513)		ii11620 ii12068	ii11555 ii11637 ii11995	ii11402 ii11619 ii12066	ii11159 ii11979	ii10633
IP Configuration Guide (SC31-8725)	II12362					
IP Configuration Reference (SC31-8726)	II12363					
IP Diagnosis (SC31-8521)	II12366	ii11628	ii11565	ii11411	ii11160 ii11414	ii10637
IP Messages Volume 1 (SC31-8517)	II12367	ii11630	ii11562	ii11408		Messages and Codes ii10635
IP Messages Volume 2 (SC31-8570)	II12368	ii11631	ii11563	ii11409		
IP Messages Volume 3 (SC31-8674)	II12369	ii11632	ii11564	ii11410	ii11158	
IP Migration (SC31-8512)	II12361	ii11618	ii11554	ii11401		

Table 61. IP Information Apars (continued)

Title	CS for OS/390 2.10	CS for OS/390 2.8	CS for OS/390 2.7	CS for OS/390 2.6	CS for OS/390 2.5	TCP/IP 3.3
IP Network Print Facility (SC31-8522)		ii11627	ii11561	ii11407	ii11150	
IP Programmer's Reference (SC31-8515)		ii11634	ii11557	ii11404		ii10636
IP and SNA Codes (SC31-8571)	II12370	ii11917	Added TCP/IP codes to VTAM codes V2R6 ii11611	ii11361	ii11146 ii11097	
IP User's Guide (GC31-8514)	II12365	ii11625	ii11556	ii11403	ii11143	ii10634

Appendix G. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
P.O.Box 12195
3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs

conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, photographs and color illustrations may not appear.

Some updates of this book, such as an update between OS/390 releases, might be available only in softcopy. You can obtain softcopy from the OS/390 Online Library Collection (SK2T-6700), the OS/390 PDF Library Collection (SK2T-6718), or the OS/390 Internet Library (*http://www.ibm.com/s390/os390/*). To order the latest hardcopy edition that is available, you might need to order a lower suffix (dash) level.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

ACF/VTAM
Advanced Peer-to-Peer Networking
AFP
AD/Cycle
AIX
AIX/ESA
AnyNet
APL2
APPN
AS/400
AT
BookManager
BookMaster
CBPDO
C/370
CICS
CICS/ESA
C/MVS
Common User Access
C Set ++
CT
CUA
DATABASE 2
DatagLANce
DB2
DFSMS
DFSMSdfp
DFSMShsm
DFSMS/MVS
Domino
DRDA
eNetwork
Enterprise Systems Architecture/370
ESA/390
ESCON
ES/3090
ES/9000
ES/9370
EtherStreamer
Extended Services
FAA
Micro Channel
MVS
MVS/DFP
MVS/ESA
MVS/SP
MVS/XA
MQ
Natural
NetView
Network Station
Nways
Notes
NTune
NTuneNCP
OfficeVision/MVS
OfficeVision/VM
Open Class
OpenEdition
OS/2
OS/390
Parallel Sysplex
Personal System/2
PR/SM
PROFS
PS/2
RACF
Resource Measurement Facility
RETAIN
RFM
RISC System/6000
RMF
RS/6000
S/370
S/390
SAA
SecureWay
Slate
SP
SP2
SQL/DS
System/360

FFST
FFST/2
FFST/MVS
First Failure Support Technology
GDDM
Hardware Configuration Definition
IBM
IBMLink
IMS
IMS/ESA
InfoPrint
Language Environment
LANStreamer
Library Reader
LPDA
MCS
System/370
System/390
SystemView
Tivoli
TURBOWAYS
UNIX System Services
Virtual Machine/Extended Architecture
VM/ESA
VM/XA
VSE/ESA
VTAM
WebSphere
XT
400
3090
3890

Lotus, Freelance, and Word Pro are trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

The following terms are trademarks of other companies:

ATM is a trademark of Adobe Systems, Incorporated.

BSC is a trademark of BusiSoft Corporation.

CSA is a trademark of Canadian Standards Association.

DCE is a trademark of The Open Software Foundation.

HYPERchannel is a trademark of Network Systems Corporation.

UNIX is a registered trademark in the United States, other countries, or both and is licensed exclusively through X/Open Company Limited.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ActionMedia, LANDesk, MMX, Pentium, and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. For a complete list of Intel trademarks, see http://www.intel.com/tradmarx.htm.

Other company, product, and service names may be trademarks or service marks of others.

Bibliography

IBM Communications Server for OS/390 Publications

This bibliography contains descriptions of the books in the IBM Communications Server for OS/390 library.

Updates to books are available on RETAIN. See "Appendix F. Information Apars" on page 551 for a list of the books and the INFOAPARS associated with them.

These books are available online. Go to http://www.ibm.com/s390/os390/bkserv/ to access the OS/390 Internet Library web page.

Some books are available in both hard- and soft-copy, or soft-copy only. The following abbreviations follow each order number:

HC/SC	Both hard- and soft-copy are available
SC	Only soft-copy is available

Related Publications

For information about OS/390 products, refer to *OS/390 Information Roadmap* (GC28-1727-07 [HC/SC]). The Roadmap describes what level of documents are supplied with each release of CS for OS/390, as well as describing each OS/390 publication.

Firewall

OS/390 SecureWay Security Server Firewall Technologies Guide and Reference (SC24-5835 [HC/SC])

OSA-Express

OSA-Express Customer's Guide and Reference (SA22-7403 [HC/SC])

Softcopy Information

- *OS/390 Online Library Collection* (SK2T-6700).

  This collection contains softcopy unlicensed books for OS/390, Parallel Sysplex products, and S/390 application programs that run on OS/390. This collection is updated quarterly with any new or updated books that are available for the product libraries included in it.
- *OS/390 PDF Library Collection* (SK2T-6718).

  This collection contains the unlicensed books for OS/390 Version 2 Release 6 in Portable Document Format (PDF).
- *OS/390 Licensed Product Library* (LK2T-2499).

  This library contains unencrypted softcopy licensed books for OS/390 Version 2. If any of the books in this library are changed, it is updated quarterly. The OS/390 Licensed Product Library for Version 1 (LK2T-6702) is still available, but is no longer updated.
- *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.

Planning

OS/390 IBM Communications Server: SNA Migration (SC31-8622 [HC/SC]). This book is intended to help you plan for SNA, whether you are migrating from a previous version or installing SNA for the first time. This book also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with SNA.

OS/390 IBM Communications Server: IP Migration (SC31-8512 [HC/SC]). This book is intended to help you plan for IP, whether you are migrating from a previous version or installing IP for the first time. This book also identifies the optional and required modifications needed to enable you to use the enhanced functions provided with IP.

Resource Definition, Configuration, and Tuning

OS/390 IBM Communications Server: IP Configuration Guide (SC31-7134 [HC/SC]). This book describes the major concepts involved in understanding and configuring an IP network. Familiarity with MVS operating, IP protocols, OS/390 UNIX System Services, and IBM Time Sharing Option (TSO) is recommended. Use this book in conjunction with the *OS/390 IBM Communications Server: IP Configuration Reference*.

OS/390 IBM Communications Server: IP Configuration Reference (SC31-8725 [HC/SC]). This book presents information for people who

want to administer and maintain IP. Use this book in conjunction with the *OS/390 IBM Communications Server: IP Configuration Guide*. The information in this book includes:

- TCP/IP configuration data sets
- Configuration statements
- Operator commands
- Translation tables
- SMF records
- Protocol number and port assignments

OS/390 IBM Communications Server: SNA Network Implementation Guide (SC31-8563 [HC/SC]). This book presents the major concepts involved in implementing a SNA network. Use this book in conjunction with the *OS/390 IBM Communications Server: SNA Resource Definition Reference*.

OS/390 IBM Communications Server: SNA Resource Definition Reference (SC31-8565 [HC/SC]). This book describes each SNA definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect SNA.Use this book in conjunction with the *OS/390 IBM Communications Server: SNA Network Implementation Guide*.

OS/390 IBM Communications Server: SNA Resource Definition Reference (SC31-8566 [HC/SC]). This book contains sample definitions to help you implement SNA functions in your networks, and includes sample major node definitions.

OS/390 eNetwork Communications Server: AnyNet SNA over TCP/IP (SC31-8578 [SC]). This guide provides information to help you install, configure, use, and diagnose SNA over TCP/IP.

OS/390 eNetwork Communications Server: AnyNet Sockets over SNA (SC31-8577 [SC]). 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.

Operation

OS/390 IBM Communications Server: IP User's Guide (SC31-7136 [HC/SC]). This book is for people who want to use TCP/IP for data communication activities such as FTP and Telnet. Familiarity with MVS operating system and IBM Time Sharing Option (TSO) is recommended.

OS/390 IBM Communications Server: SNA Operation (SC31-8567 [HC/SC]). This book serves as a reference for programmers and operators requiring detailed information about specific operator commands.

OS/390 IBM Communications Server: Quick Reference (SX75-0121 [HC/SC]). This book contains essential information about SNA and IP commands.

OS/390 eNetwork Communications Server: High Speed Access Services Users Guide (GC31-8676 [SC]). This book is for end users and system administrators who want to use applications using a High Speed Access Services connection available in CS for OS/390.

Customization

OS/390 IBM Communications Server: SNA Customization (LY43-0110 [SC]). This book enables you to customize SNA, and includes:

- 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

OS/390 eNetwork Communications Server: IP Network Print Facility (SC31-8074 [SC]). This book 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.

Writing Application Programs

OS/390 IBM Communications Server: IP Application Programming Interface Guide (SC31-7187 [SC]). This book 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 book to adapt your existing applications to communicate with each other using sockets over TCP/IP.

OS/390 IBM Communications Server: IP CICS Sockets Guide (SC31-8518 [SC]). This book is for people who want to set up, write application

programs for, and diagnose problems with the socket interface for CICS using TCP/IP for MVS.

OS/390 eNetwork Communications Server: IP IMS Sockets Guide (SC31-8519 [SC]). This book is for programmers who want application programs that use the IMS TCP/IP application development services provided by IBM TCP/IP for MVS.

OS/390 IBM Communications Server: IP Programmer's Reference (SC31-8515 [SC]). This book 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 MVS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.

OS/390 IBM Communications Server: SNA Programming (SC31-8573 [SC]). This book 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.

OS/390 eNetwork Communications Server: SNA Programmers LU 6.2 Guide (SC31-8581 [SC]). This book describes how to use the SNA LU 6.2 application programming interface for host application programs. This book 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 book.)

OS/390 eNetwork Communications Server: SNA Programmers LU 6.2 Reference (SC31-8568 [SC]). This book provides reference material for the SNA LU 6.2 programming interface for host application programs.

OS/390 eNetwork Communications Server: CSM Guide (SC31-8575 [SC]). This book describes how applications use the communications storage manager.

OS/390 IBM Communications Server: CMIP Services and Topology Agent Guide (SC31-8576 [SC]). This book describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The book provides guide and reference information about CMIP services and the SNA topology agent.

Diagnosis

OS/390 IBM Communications Server: IP Diagnosis (LY43-0105 [HC/SC]). This book 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.

OS/390 IBM Communications Server: SNA Diagnosis V1 Techniques and Procedures(LY43-0079 [HC/SC]) and *OS/390 IBM Communications Server: SNA Diagnosis V2 FFST Dumps and the VIT* (LY43-0080 [HC/SC]). These books help you identify a 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.

OS/390 IBM Communications Server: SNA Data Areas Volume 1 (LY43-0111 [SC]) and *OS/390 IBM Communications Server: SNA Data Areas Volume 2* (LY43-0112 [SC]). These books describe SNA data areas and can be used to read a SNA dump. They are intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.

Messages and Codes

OS/390 IBM Communications Server: SNA Messages (SC31-8569 [HC/SC]). This book describes the ELM, IKT, IST, ISU, IVT, IUT, and USS messages. Other information in this book includes:

- Command and RU types in SNA messages
- Node and ID types in SNA messages
- Supplemental message-related information

OS/390 IBM Communications Server: IP Messages Volume 1 (EZA) (SC31-8517 [HC/SC]). This volume contains TCP/IP messages beginning with EZA.

OS/390 IBM Communications Server: IP Messages Volume 2 (EZB) (SC31-8570 [HC/SC]). This volume contains TCP/IP messages beginning with EZB.

OS/390 IBM Communications Server: IP Messages Volume 3 (EZY-EZZ-SNM) (SC31-8674 [HC/SC]). This volume contains TCP/IP messages beginning with EZY, EZZ, and SNM.

OS/390 IBM Communications Server: IP and SNA Codes (SC31-8571 [HC/SC]). This book describes codes and other information that display in CS for OS/390 messages.

APPC Application Suite

OS/390 eNetwork Communications Server: APPC Application Suite User's Guide (GC31-8619 [SC]). This book 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.

OS/390 eNetwork Communications Server: APPC Application Suite Administration (SC31-8620 [SC]). This book contains the information that administrators need to configure the APPC application suite and to manage the APING, ANAME, AFTP, and A3270 servers.

OS/390 eNetwork Communications Server: APPC Application Suite Programming (SC31-8621 [SC]). This book provides the information application programmers need to add the functions of the AFTP and ANAME APIs to their application programs.

Multiprotocol Transport Networking (MPTN) Architecture Publications

Following are selected publications for MPTN:

Networking Blueprint Executive Overview (GC31-7057)

Multiprotocol Transport Networking: Technical Overview (GC31-7073)

Multiprotocol Transport Networking: Formats (GC31-7074)

Redbooks

The following Redbooks may help you as you implement CS for OS/390.

- *OS/390 eNetwork Communication Server V2R7 TCP/IP Implementation Guide Volume 1: Configuration and Routing* (SG24–5227–01).

  This book provides examples of how to configure the base TCP/IP stack, routing daemons and the TELNET server. This book also provides information about national language support (NLS), routing, OSPF, network interfaces, diagnosis, multicasting, OS/390 UNIX System Services and security in an OS/390 UNIX System Services environment.
- *OS/390 eNetwork Communication Server V2R7 TCP/IP Implementation Guide Volume 2: UNIX Applications* (SG24–5228–01).

  This book provides information about implementing applications that run in the OS/390 UNIX environment, such as FTP, SNMP, BIND-based name server, DHCP, and SENDMAIL. This book also provides configuration samples and describes the implementation process.
- *OS/390 eNetwork Communication Server V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications* (SG24–5229–01).

  This book provides information about TCP/IP applications that run in a legacy MVS environment, including CICS/IMS Sockets, and printing (NPF, LPR, and LPD.)
- *OS/390 Secureway Communication Server V2R8 TCP/IP Guide to Enhancements* (SG24-5631-00).

  This redbook provides information to facilitate the configuration and use of the new technologies and functions supported in SecureWay Communications Server for OS/390 V2R8. Special areas of interest in this book are security and Quality of Services.
- *TCP/IP in a Sysplex* (SG24–5235–01).

  The main goals of a Parallel Sysplex are high availability and high performance. This book demonstrates how these goals can be achieved in the particular environment of SecureWay Communications Server for OS/390 and its TCP/IP applications. This book describes the WLM/DNS functions, the Network Dispatcher and the Dynamic VIPA.
- *SNA and TCP/IP Integration* (SG24–5291–00).

  This book provides information about integrating current SNA network with future TCP/IP and Web-based communication requirements. This book concentrates on routing techniques.
- *SNA in a Parallel Sysplex Environment* (SG24–2113–01).

This book provides information about implementing a VTAM-based network on a Parallel Sysplex.

- *Subarea to APPN Migration : VTAM and APPN Implementation* (SG24–4656–01).

  This book is the first of two volumes. This book provides information about the migration of a subarea network to an APPN network. Some knowledge of SNA subarea networks and familiarity with the functions, terms and data flows of APPN networks is assumed.

- *Subarea to APPN Migration : HPR and DLUR Implementation* (SG24–5204–00).

  This book is the second of two volumes. This book provides information about the coverage of a network using HPR, DLUR and APPN/HPR routers. Some knowledge of SNA subarea networks and familiarity with the functions, terms and data flows of APPN networks is assumed.

Bibliography

Index

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Readers' Comments — We'd Like to Hear from You

OS/390 IBM Communications Server
IP Diagnosis Guide
Version 2 Release 10

Publication No. SC31-8521-04

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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.

Name _____ Address _____

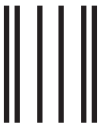
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC31-8521-04

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Software Reengineering
Department G7IA/ Bldg 503
Research Triangle Park, NC
27709-9990

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5647–A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC31-8521-04

Spine information:

OS/390 IBM Communications Server

OS/390 V2R10.0 IBM CS IP Diagnosis Guide

Version 2 Release 10