

---

---

# Unicenter

## TCPaccess Communications Server System Management Guide

Version 6.0



**Computer Associates**  
The Software That Manages eBusiness



This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2002 Computer Associates International, Inc. (CA)

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

# Contents

## Chapter 1: Introduction

Unicenter TCPAccess Operations .....	1-1
System Commands .....	1-2
Command Scripts.....	1-2
Using the System Management Facility (SMF) .....	1-3
Diagnostic Procedures .....	1-3
TSO Diagnostic Commands.....	1-3

## Chapter 2: Unicenter TCPAccess Operation

Operator Commands .....	2-2
Starting and Stopping.....	2-2
Startup .....	2-3
Shutdown .....	2-3
Starting and Stopping Task Groups .....	2-6
Shutting Down with P versus P CLEAR.....	2-6
Recycling Task Groups.....	2-7
Converting Translate Tables .....	2-8
Latch Command .....	2-8
Related VTAM Commands.....	2-8
JCL Requirements.....	2-9
JCL Descriptions.....	2-9
Sample JCL for Starting Unicenter TCPAccess .....	2-10
Startup JCL Customization.....	2-11
Starting and Stopping Unicenter TCPAccess .....	2-12
T01LOG Log Spin Function .....	2-13
STARTxx Configuration.....	2-14
Initial STARTxx Customization .....	2-14

---

System Commands .....	2-15
Command Format.....	2-15
Subsystem Recognition Character.....	2-16
Task Group Identifier (TGI) .....	2-16
Verbs.....	2-17
Objects .....	2-17
Operands.....	2-18
Comments.....	2-19
Delimiters.....	2-19
General Task Group Commands .....	2-20
SNAP .....	2-20
STATUS.....	2-21
Dynamic Configuration Commands .....	2-21
Commands Imbedded in the TCPCFGUP Member.....	2-23
DELETE.....	2-23
DEVICE .....	2-25
LNI.....	2-26
UPDATE.....	2-27
ARM Command .....	2-28
REFRESH Command.....	2-29
DNR Commands.....	2-30
DUMP.....	2-30
PURGE.....	2-31
GateD (GATED) Commands.....	2-32
IJT Commands.....	2-33
GTF.....	2-33
HELP.....	2-34
IFS.....	2-34
ILATCH.....	2-35
ILATCH.....	2-35
ILATCH CONTENTION .....	2-36
ILATCH TRON .....	2-36
LOGGING.....	2-37
MODULE .....	2-40
MEM .....	2-40
MVS .....	2-41
P.....	2-42
POOL.....	2-42
SET .....	2-45
SRC.....	2-46
START .....	2-46

---

STCK .....	2-47
STOP .....	2-47
SVCDUMP .....	2-48
TASK .....	2-49
TIME .....	2-49
TRACE .....	2-49
VSM.....	2-50
TSO Commands.....	2-50
CONVXL8 Command.....	2-50
LOADXL8 Command .....	2-52
Command Scripts.....	2-53
Sample Command Script .....	2-53
Notes .....	2-54

## Chapter 3: Using the System Management Facility (SMF)

Standard MVS SMF Record Header .....	3-2
Standard Unicenter TCPaccess SMF Record Header .....	3-2
Descriptor .....	3-2
Product .....	3-3
Server Application Event Records .....	3-4
Task Identification Section.....	3-4
User Identification Section.....	3-4
FTP Server Records (Subtypes 20/22).....	3-5
Descriptor .....	3-5
Common Data.....	3-6
FTP Data Transfer Completion/End of Volume (Subtype 20/Subtype 22) Data Section .....	3-7
FTP Data Set Modification (Subtype 21) Data Section .....	3-10
Telnet Server Record (Subtype 23) Data Section.....	3-11
Access Method and Protocol Layer Event Records .....	3-13
Access Method User Identification Section .....	3-13
Access Method Endpoint Close Records (Subtypes 150 - 152) .....	3-13
UNIX System Services Endpoint Close (Subtype 150) Data Section.....	3-14
IUCV Endpoint Close (Subtype 151) Data Section .....	3-15
Assembler API Endpoint Close (Subtype 152) Data Section.....	3-18
Protocol Layer Events (Subtypes 110 - 123).....	3-19
Protocol Layer User Identification Section .....	3-20
Protocol Layer Data Section.....	3-21
Interval Records.....	3-23
Driver Statistics (Subtype 100) Data Section .....	3-23
Virtual Storage Statistics (Subtype 80) Data Section.....	3-24

---

SMF Report Writer Program .....	3-27
Commands.....	3-28

## Chapter 4: Diagnostic Procedures

Problem Determination.....	4-1
Failures That Produce Dumps .....	4-2
Failures Due To Network Problems .....	4-2
Failure of Unicenter TCPaccess to Respond.....	4-2
Failure of Major Applications .....	4-2
Loops .....	4-3
Latch Contention.....	4-3
Storage shortages .....	4-3
Problem Reporting.....	4-4
ABENDs .....	4-5
Incorrect Output – Telnet.....	4-5
Incorrect Output – FTP.....	4-6
Incorrect Output – DNR, EPS, and so forth.....	4-6
Hardware .....	4-7
Startup and Parameter Errors .....	4-7
TCP API Trace Procedures .....	4-8
Troubleshooting GateD.....	4-9
GateD Logs .....	4-9
Controlling GateD.....	4-9
Creating Formatted Dumps for GateD .....	4-10
Troubleshooting the CDLC Driver .....	4-10
Troubleshooting IUCV Sockets .....	4-11
Diagnosing Application-Level Problems.....	4-11
Socket Function Calls .....	4-12
Diagnosing Other IUCV Failures.....	4-14
Reporting IUCV Failures .....	4-14
Macro API Diagnostic Facilities .....	4-15
Examples .....	4-16
Character Formatted Socket Trace .....	4-17

---

## Chapter 5: Diagnostic Commands

Troubleshooting Client and Server FTP3 .....	5-2
General Guidelines for Problem Isolation .....	5-2
Resolving Server FTP Problems .....	5-2
Resolving Client FTP Problems .....	5-3
ACTEST .....	5-3
DNRGET .....	5-8
NETSTAT/SYSSTAT .....	5-11
Command Syntax .....	5-12
Dotted Notation IP Address Wildcarding ( * ) .....	5-12
TSO Command Common Parameters .....	5-13
NETSTAT Diagnostic Commands .....	5-13
ARP .....	5-13
HELP .....	5-14
CNFG .....	5-14
CONN .....	5-15
RESOLVE .....	5-16
ROUTE .....	5-17
RTM .....	5-17
TELNET .....	5-18
USER .....	5-20
XCF .....	5-20
SYSSTAT Diagnostic Commands .....	5-21
CANC .....	5-21
CHANGE .....	5-22
OSPFMON .....	5-23
Example .....	5-23
Local Commands .....	5-24
Remote Commands .....	5-24
PING .....	5-26
RIPQUERY .....	5-27
RPCINFO .....	5-29
STROBE .....	5-30

---

TCPEEP .....	5-31
User Interface.....	5-31
Trace Data Collected .....	5-32
Viewing Trace Data .....	5-32
Trace Operation.....	5-32
TCPEEP Syntax .....	5-33
Useful TCPEEP Commands .....	5-44
TCPEEP Examples .....	5-45
TRACE.....	5-48
MVS Component Trace.....	5-48
Trace Address Space.....	5-48
Exit .....	5-49
External Writer .....	5-49
Component Trace JCL .....	5-49
TRACE Command.....	5-50
TRACE Command Reply.....	5-52
TRACE Command Reply Options.....	5-53
Comparing TCPEEP and MVS TRACE Syntax .....	5-59
Trace Command Examples.....	5-59
Processing Trace Data .....	5-61
TRACERT.....	5-62
Usage Guidelines .....	5-63
Example.....	5-64

## Index

# Introduction

---

The System Management guide provides information on maintaining Unicenter TCPAccess Communications Server.

The organization of the guide is as follows:

- [Operations](#) – Describes operator commands, system, general task group, dynamic configuration, APP, DNR, and IJT commands, JCL requirements, the STARTxx member of the PARM library, and command scripts.
- [Using the System Management Facility \(SMF\)](#) – Describes the SMF records, provides samples of the record headers and other records and an example of the SMF report writer program.
- [Diagnostic Procedures](#) – Describes the available diagnostic utilities to troubleshoot various Unicenter TCPAccess failures and the documentation needed to report problems to customer support.
- [Diagnostic Commands](#) – Describes how to troubleshoot Client and Server FTP3 and describes the diagnostic commands.

## Operations

Unicenter TCPAccess can be run as a batch job or as a started task. The chapter “Unicenter TCPAccess Operation” describes Unicenter TCPAccess startup and shutdown.

The operator commands to start and shutdown Unicenter TCPAccess are described, including orderly shutdown and fast shutdown, shutting down with **P** versus **P CLEAR**, recycling of task groups, and converting translate tables.

A JCL is provided on the distribution tape and this chapter provides a table of statements and description of the associated Unicenter TCPAccess JCL. A sample startup JCL and customization information is also given.

The T01LOG logspin utility lets you specify that the T01LOG SYSOUT file be closed and re-opened on a regular basis determined by either the number of lines, a period of time, or by a combination of the two.

The START<sub>xx</sub> configuration member is described and customization tips are provided.

## System Commands

The format of system commands and some of the most commonly used commands are described.

The following task groups and their commands are described along with syntax examples:

- General Task Group Commands
- Dynamic Configuration Commands
- APP Task Group Commands
- DNR Task Group Commands
- GateD Task Group Commands
- IJT Task Group Commands
- UNIX System Services Commands (formerly known as TSO OpenEdition)

## Command Scripts

Command scripts are a prearranged executable sequence of IFS task group commands and command script statements. A sample command script is provided and syntax is described.

## Using the System Management Facility (SMF)

This chapter provides sample of the following SMF record formats:

- Standard MVS SMF record headers
- Standard Unicenter TCPAccess SMF record headers
- Server application event records
- Access method and protocol layer event records
- Interval records
- SMF report writer program

## Diagnostic Procedures

This chapter provides information to help with problem determination and problem reporting.

Problem determination information describes how to troubleshoot IUCV sockets and Gated.

Problem reporting tells you about ABENDs, incorrect output for Telnet, FTP, DNR, EPS, and so forth, hardware and startup and parameter errors, and TCP API trace procedures.

## TSO Diagnostic Commands

This chapter provides tips for troubleshooting and describes the diagnostic commands.



# Unicenter TCPaccess Operation

---

This chapter describes the basic operation of Unicenter TCPaccess.

It includes the following topics:

- [Operator Commands](#) – Describes the operator commands, including starting and stopping and related VTAM commands
- [JCL Requirements](#) – Describes the JCL issues for Unicenter TCPaccess operation and how to execute Unicenter TCPaccess as a started task
- [STARTxx Configuration](#) – Describes information for the STARTxx member
- [System Commands](#) – Describes functions, syntax, and arguments of Unicenter TCPaccess system commands
- [General Task Group Commands](#) – Describes the SNAP and STATUS commands
- [Dynamic Configuration Commands](#) – Describes the dynamic configuration commands
- [REFRESH Command](#) – Describes the REFRESH command that is processed by either the APP task group or the TCP task group
- [DNR Commands](#) – Describes the DUMP command processed by the DNR task group
- [GateD \(GATED\) Commands](#) – Describes commands processed by GateD (GTD) task group
- [IJT Commands](#) – Describes the commands processed by the job step (IJT) task group
- [TSO Commands](#) – Describes the TSO commands available for Unicenter TCPaccess UNIX System Services (formerly OpenEdition) socket users
- [Command Scripts](#) – Describes how to build command scripts and includes a sample command script and some notes

## Operator Commands

This section describes the operator commands available to Unicenter TCPAccess users.

### Starting and Stopping

Unicenter TCPAccess can be run as a batch job or as a started task. Many installations route started JCL output to a purge output class.

Running Unicenter TCPAccess as a batch job (as opposed to a started task) can provide valuable information in the JES logs if problems arise. Due to cross-memory services restrictions in MVS, the JES initiator is terminated after Unicenter TCPAccess, running as a batch job, is stopped. For this reason it is probably better to run Unicenter TCPAccess as a started task. Use the appropriate start command to start the Unicenter TCPAccess JCL procedure.

Orderly shutdown of Unicenter TCPAccess notifies the application layer facilities that a shutdown was requested. These facilities can then terminate prior to the termination of the underlying protocols. Orderly shutdown also lets the IFS task groups that make up Unicenter TCPAccess (TCP, APP, DNR, MAP, and SNM) terminate gracefully with regard to the various interdependencies and interfaces that exist between them.

If you are using the Inter-User Communications Vehicle (IUCV), examine the T01LOG output data set for message T01IU001I (Connection to IUCV Established). If Unicenter TCPAccess cannot connect to IUCV, message T01IU004I displays.

If Unicenter TCPAccess fails to connect to IUCV, it continues to retry at 30-second intervals until successful connection.

**Note:** The stepname given to the Unicenter TCPAccess started task becomes the TCPIPJOBNAME that applications use to connect via IUCV to the Unicenter TCPAccess started task. If no stepname is given, then the JOBNAME (or task name) is used.

## Startup

Unicenter TCPAccess starts by executing IFS. IFS initializes the address space as a whole and then starts the other task groups. To do so, IFS reads and executes the start command script defined by CMND=START $xx$  in the PARM field of the EXEC statement and contained in the PARM data set defined by the SYSPROC DD statement.

Each task group is started by an IFS START command in the command file. This command, in turn, specifies the configuration file member containing the parameters used by the task group as it starts. The main configuration member for the task group is specified by CNFG( $xx$ ) on the IFS START command for the task group. Some task groups also use secondary configuration members from the PARM data set as specified in their main configuration file. Typically, a task group dynamically loads various other programs and tables from the STEPLIB data set during initialization and later, as needed.

**Note:** The IJT task group uses IJTCFG $xx$ , since this task group controls all other task groups. Therefore, you cannot restart this task group without stopping and restarting the entire Unicenter TCPAccess address space.

## Shutdown

You can initiate shutdown using these methods:

- Issue the MVS STOP command (**P**) for the Unicenter TCPAccess job name (for example, **P TCPAccess**)
- Use the IFS subsystem P command ( subsystem recognition character preceding P).
- Use the MVS MODIFY command (for example, **F jobname,P**)

The MVS operator is prompted after the STOP command to confirm the shutdown operation.

**Note:** See the NOPROMPT command in the *Unicenter TCPAccess Customization Guide* to turn off the WTOR prompt.

The first stop request places Unicenter TCPAccess into drain or *slow shutdown* mode. In this mode, Unicenter TCPAccess lets existing protocol and API activity continue for an indefinite period and does not proceed with other termination functions until the completion of these activities. User Level Protocol Processes (ULPPs) (for example, FTP, SMTP) and API applications are notified that drain mode is in effect. As a result, API applications should start their termination procedures.

The first P command causes the following for each task group:

- TCP APEND and TPEND are driven with DRAIN and STOP for all applications and GateD is stopped.  
If all applications issue TClose and AClose, then TCP continues shutdown automatically, stopping each LNI and freeing resources.
- APP All Ptasks are notified of termination. They should also see APEND and TPEND from TLI (for example, TCP).  
If all Ptasks terminate, then the APP task group stops.
- MAP Exit immediately.
- SNM Exit immediately.
- DNR Tries to stop the receive, send and timer threads.  
When all threads end, shutdown continues.
- RTM Attempts to stop all active threads. When all threads have ended, shutdown continues.

Since no new API endpoints or protocol sessions can be established when drain mode is in effect, listening endpoints receive a return code indicating the drain state. The socket interface interprets this return code and terminates certain socket functions in drain mode. Idle server tasks should disconnect from the API in this situation, letting Unicenter TCPaccess shutdown.

- Fast Shutdown Mode If a second operator stop request is issued, Unicenter TCPaccess enters *stop* or *fast shutdown* mode. In fast shutdown mode, any remaining API endpoints and ULPPs are notified to terminate immediately. This results in the termination of all protocol connections and the purging of all pending or outstanding API requests.

The second P command causes the following for each task group:

- TCP If any active TLI, OE socket, or IUCV socket applications are active, message T01SO010 appears. This indicates TERM phase and displays the number of active address spaces using API services.  
APEND and TPEND with the TERM indication is issued to all remaining TLI users.  
If PROMPT was specified in IJTFCFGxx, message T01SO012 appears asking the operator if shutdown should continue. If the operator replies Y, shutdown of TCP continues immediately, otherwise shutdown is delayed.

- APP Any active Ptasks are again notified to stop.  
All active TLI connections are immediately closed if still open (usually TPEND STOP from the first P command causes them to close immediately).
- DNR Continues shutdown.
- MAP Continues shutdown.
- SNM Continues shutdown.
- RTM Continues shutdown.

#### Cancel Shutdown Mode

Normally, only one or two operator stop requests are necessary to terminate Unicenter TCPaccess. However, it is possible for API applications to require a third stop request. The third stop request terminates the remaining Unicenter TCPaccess components regardless of an API application's refusal to issue the required API termination functions. A third stop request also forces detaching of MVS tasks within the APP task group. More than three operator stop requests have no effect on the termination of Unicenter TCPaccess. If necessary, the MVS CANCEL command can be used.

The third P command causes the following for each task group:

- TCP Stop immediately regardless of active users.  
**Note:** If the operator replied N to message T01SO012, this is treated as a second stop and message T01SO012 is reissued.
- APP Stops immediately.
- DNR Stop immediately.
- MAP Stop immediately.
- SNM Stops immediately.
- RTM Stops immediately.

- If the address space does not terminate normally after three stop commands:
  1. Issue **F TCPACCES,ILATCH** to see if there are any hung latches.
  2. If any latches are hung, issue **F TCPACCES,ILATCH FREE LATCH(*nn*)**.  
Where *nn* is the latch number for each hung latch. Shutdown should then proceed.
- If there are no hung latches after the third P command:
  1. Issue **F TCPACCES,SVCDUMP**.
  2. When the SVCDUMP is complete, issue **C TCPACCES** to cancel the address space
  3. Contact technical support, supplying the SVCDUMP and address space for problem resolution.

### Starting and Stopping Task Groups

The individual IFS task groups within Unicenter TCPaccess can be stopped and started independently with the IFS subsystem STOP and START commands.

#### Example

If the DNR task group terminates abnormally, you can restart it without terminating and restarting the rest of Unicenter TCPaccess. See [START](#) for more information on restarting task groups.

**Note:** Any task group can be stopped and started to pick up new configuration information, with the exception of the IJT and TCP task groups.

### Shutting Down with P versus P CLEAR

The normal method of shutting down Unicenter TCPaccess is to issue the MVS STOP (P) command.

```
f TCPaccess,P
```

or

```
%P (substituting your subsystem identification character for %)
```

If you plan to install maintenance on the Unicenter TCPaccess base product before restarting Unicenter TCPaccess, use the following command to stop Unicenter TCPaccess:

```
f runtcp,P CLEAR
```

or

```
%P CLEAR (substituting your subsystem identification character for %)
```

The P CLEAR command clears modules and control blocks from CSA and the subsystem hooks installed by this address space.

Additionally, the P CLEAR command also performs the normal processing associated with an MVS STOP (P) command.

**Note:** Use the CLEAR option to stop Unicenter TCPaccess before applying the updates. This command also lets Unicenter TCPaccess use the most recent versions of the base modules when it is restarted.

The command must be issued using the subsystem recognition character, (that is, P CLEAR) or the MVS modify command:

```
F TCPaccess,P CLEAR
```

When a P CLEAR is issued, it is possible that TCPaccess takes an SOC1 or SOC4 during termination. This is normal due to the missing CSA control blocks and modules. Only one P CLEAR is needed to clear the CSA.

After issuing a P CLEAR, it may be necessary to follow additional stopping instructions for Unicenter TCPaccess. For more information, see the section on [STOP](#).

## Recycling Task Groups

While typically not needed, individual task groups can be stopped and restarted while Unicenter TCPaccess is running.

To stop a task group, issue the following command from the MVS operator console:

```
IFS STOP tgi
```

To restart a task group, issue the following command from the MVS operator console

```
IFS START tgi
```

Where *tgi* is the task group identifier.

One or more IFS START commands can be issued from a command script as shown in the startup example in [Startup JCL Customization](#).

## Converting Translate Tables

The CONVXL8 TSO command converts a table from editable text to binary. See the description of this command, later in this chapter.

The LOADXL8 TSO command works like the CONVXL8 command, but reads the load module from compiled Unicenter TCPAccess translate tables as input. Refer to the description of the LOADXL8 command, later in this chapter.

## Latch Command

The IJT command, ILATCH, is a latch lock utility that displays and frees latches used by UNIX System Services sockets. It can be used to serialize resources in the local address space. See the description of the [ILATCH](#) command, later in this chapter.

## Related VTAM Commands

To activate the major node to VTAM for Unicenter TCPAccess, issue the following command:

```
V NET,ACT,ID=major_node_name
```

**Note:** The distributed major node is A03ACCES.

This command also activates the LUs required for Server Telnet and the client user commands (for example, **FTP**, **FTP2**, and so on.).

## JCL Requirements

The JCL provided on the distribution tape is functionally equivalent to that listed in the following topics, but may differ in the order of JCL statements or the contents of comment statements included in the JCL. (See the following table, in [JCL Descriptions](#), which contains statements and descriptions of the Unicenter TCPaccess JCL).

**Note:** The sample startup JCL, RUNTCP, is in *TRGINDX.CNTL(RUNTCP)*.

## JCL Descriptions

The following table lists Unicenter TCPaccess JCL DD statements and descriptions.

DD Statement	Description
ABNLIGNR DD	Disables the Abend-Aid program product.
ARPAHELP DD	Defines a PDS whose members contain the HELP text displayed by Server Telnet and Server FTP.
DNRERR DD	DNR error log file.
DNRLOG DD	DNR log file.
GTDERR DD	GateD error log file.
GTDLOG DD	GateD log file.
GTDTRC DD	GateD trace log file.
MAPERR DD	Port mapper error log file.
MAPLOG DD	Port mapper logging file.
SNMLOG DD	SNM task group logging file.
STEPLIB DD	Defines the load library for Unicenter TCPaccess. These data sets must be APF-authorized.
SYSHELP DD	Defines the help data set for the Unicenter TCPaccess task groups.
SYS Parm DD	Defines the PARM data set that contains all the configuration file members, which provide the parameters for the various task groups within Unicenter TCPaccess.
SYS PRINT DD	DNR and Port Mapper logging file.
SYS PROC DD	Defines the PARM data set from which command scripts (command file members) are obtained by the IJT task group.

DD Statement	Description
	In particular, the PARM data set contains the command file member START $xx$ , which contains the commands to start Unicenter TCPaccess.
SYSSNAP DD	Defines the data set to receive dynamic formatted dumps created via the IFS SNAP operator command (if no SNAP parameter list is specified).
SYSUDUMP DD	Contains formatted system dumps.
T01LOG	Defines the log file.
SYSABEND	Contains system dumps.

### Sample JCL for Starting Unicenter TCPaccess

Use this Unicenter TCPaccess JCL with all interfaces including LOOPBACK. Edit the appropriate member, supplying the correct start member and verifying other symbolic parameters for accuracy.

```
//RUNTCP JOB
//*
//*   SAMPLE JCL PROCEDURE TO RUN TCP/IP
//*   THIS JCL CAN BE USED WITH ANY INTERFACE
//*
//*   EDIT THE TRGINDX, SSN, SRC, SOUT, CMND SYMBOLIC PARAMETERS
//*
//*   VERIFY THAT THE JOB CARD AND NAMING CONVENTIONS MEET
//*   YOUR SITE'S JCL REQUIREMENTS, THEN SUBMIT THIS JOB.
//*
//TCPIP PROC TRGINDX='TRGINDX', TARGET LIBRARIES DSN INDEX
//      SSN=ACSS,           DFLT SUBSYSTEM NAME
//      SRC='%*',           DFLT SUBSYSTEM RECOGNITION CHAR
//      SOUT='*',           CHOOSE A HOLD NONPURGE SYSOUT CLASS
//      CMND=START00        DFLT STARTUP COMMAND SCRIPT NAME
//      CNFG=00             IJTCFGxx SUFFIX
//*
//TCPIP EXEC PGM=IFSSTART,REGION=6144K,TIME=1440,
// PARM=' IFSINIT,U=&SSN,P=T01,SR=&SRC,SO=&SOUT,CM=&CMND,CF=&CNFG'
//*
//STEPLIB DD DISP=SHR,DSN=&TRGINDX..LOAD
//          DD DISP=SHR,DSN=&TRGINDX..SASLINK
//*
//* WARNING: THE LOAD DATA SET MUST NEVER BE ADDED TO THE LINK LIST.
//*           TCPACCESS' ELEMENT NAMES ARE NOT UNIQUE AND COULD AFFECT
//*           THE OPERATIONS OF OTHER SOFTWARE. THE LOAD DATA SET SHOULD
//*           ALWAYS BE REFERENCED THROUGH A STEPLIB OR JOBLIB STATEMENT.
//*
//*           CONFIGURATION DATA SETS
//*
//SYSPARM DD DISP=SHR,DSN=&TRGINDX..PARM
//SYSPROC DD DISP=SHR,DSN=&TRGINDX..PARM
//*
```

```

//*      LOG DATA SETS
//*
//T01LOG DD  SYSOUT=&SOUT
//SYSPRINT DD  SYSOUT=&SOUT
//DNRLLOG DD  SYSOUT=&SOUT
//DNRERR DD  SYSOUT=&SOUT
//GTDLOG DD  SYSOUT=&SOUT
//GTDERR DD  SYSOUT=&SOUT
//GTDTRC DD  SYSOUT=&SOUT
//MAPLOG DD  SYSOUT=&SOUT
//MAPERR DD  SYSOUT=&SOUT
//SNMLOG DD  SYSOUT=&SOUT
//*
//*      DUMP DATA SETS
//*
//SYSUDUMP DD  SYSOUT=&SOUT
//*
//*      MISC DATA SETS
//*
//ARPAHELP DD  DISP=SHR,DSN=&TRGINDX..HELP
//SYSHelp DD  DISP=SHR,DSN=&TRGINDX..HELP
//ABNLIGNR DD  DUMMY                /* DISABLE ABEND-AID PROCESSING */

```

## Startup JCL Customization

Make the following changes to the startup JCL on the PROC statement:

**TRGINDX=** Enter the high level qualifier of the Unicenter TCPaccess data sets.

**SSN=** Enter the subsystem ID you are using.

Alias: U.

Default: ACSS.

**SRC=** Enter the subsystem recognition character you are using. (Verify that it does not conflict with any other subsystems you are running.)

Alias: SUBS.

Default: %.

**SOUT=** Choose an installation defined SYSOUT class that will be held and not automatically purged at job or started task termination.

IFS uses this SYSOUT class for dynamically allocated SYSOUT data sets produced during task ABENDs, output from the operator SNAP command, and task initialization to print copies of the task startup PARM members.

Alias: SYSO.

Default: \*.

**CMND=** Enter the name of the STARTxx member in the PARM library.

Alias: CMD.

Default: START00.

- CNFG= Enter the suffix of the IJTFCGxx configuration member.  
Alias: CFG.  
Default: 00.
- PRFX= Application prefix.  
Alias for PRFX is PFX=. Default is T01.  
Note: This parameter is deprecated and may be discontinued in a later release. Computer Associates recommends that you omit it from your JCL, and use the IFSPARM APPLICATIONS keyword instead to specify the applications you want to run.

**CAUTION!** Many installations route started task JCL output to a purge class for deletion at started task termination. If an installation's default SYSOUT class is purged at job or started task termination and is using SOUT='\*', then all dumps or SNAPS produced will be lost.

**Note:** If you have not link-listed the Unicenter TCPaccess LINK library, you must not reference it in the startup JCL. However, if you use any of the client commands such as FTP2 or TCPEEP, you need to STEPLIB to it in the TSO logon procedure or JCL streams if you run the commands in batch.

These are the only changes required. Copy the JCL stream from TRGINDX.CNTL(RUNTCP) to your started task procedure library.

You are now ready to start Unicenter TCPaccess. Before doing so, it is a good idea to review the installation steps in the Unicenter TCPaccess Communications Server *Release Notes* to ensure that you have made all the necessary changes.

## Starting and Stopping Unicenter TCPaccess

To start Unicenter TCPaccess, issue the MVS **START** command.

To shut down the address space you can use a MVS MODIFY command (F *TCPaccess*, P), the subsystem recognition character (for example, %P) or the P **TCPaccess** MVS STOP command.

The following system message appears requesting confirmation:

```
T01IF013R Confirm request to stop A/S -- Reply 'Y' or 'N'
```

As termination continues, messages are issued indicating that various components are terminating.

## T01LOG Log Spin Function

The T01LOG log spin function lets you specify that the T01LOG SYSOUT file should be closed and reopened on a regular basis determined by the number of lines, a period of time, or by a combination of the two. The log spin function is implemented by including the necessary spin parameter on the LOGGING statement in the IJTFCGxx member. See the Unicenter TCPaccess *Customization Guide* for information about the IJTFCG member.

This feature is useful for customers who have long uptimes where spool usage can grow to the point that it impacts Unicenter TCPaccess performance.

This feature, along with the addition of FREE=CLOSE on the T01LOG DD statements, lets you examine, print, or purge output at the convenience of the installation and prevent uncontrolled growth of the spool.

If you specify FREE=CLOSE on the T01LOG DD statement, it causes the SYSOUT data set to be freed and made available on the output queue. Without this DD parameter, at the closing of the T01LOG file, the output remains as part of the total job output with each iteration of output appended to the last.

**Note:** There is currently no ability to spin the logs at a specified time of day (for example, at 8 A.M. each day). You can only specify hours since Unicenter TCPaccess was started.

## STARTxx Configuration

The START<sub>xx</sub> member in the PARM library is an IFS command script that tells Unicenter TCPaccess which task groups to start and which configuration members to use.

The START<sub>xx</sub> member is the highest-level configuration file, since it points to the configuration members for each of the task groups. It is pointed to by the CMND= symbolic parameter in the startup JCL procedure.

The default file, START00, as distributed, contains the following:

```
DISPLAY IFS
DISPLAY SRC
START TCP CNFG(00)
START RTM CNFG(00)
START APP CNFG(00)
START DNR CNFG(00)
START MAP CNFG(00)
START SNM CNFG(00)
SET TEST ON TGB(IJT)
```

### Initial STARTxx Customization

Individual commands are described later in this chapter. Do **not** eliminate any of them. For installation purposes, the ones of interest are the START commands for the TCP and DNR task groups. When initially installing Unicenter TCPaccess, there is no need to make changes to any of the other groups listed.

1. Make a copy of the START00 member, giving it a new name (such as START01).
2. Change the CNFG parameters on the START TCP and START DNR commands to match the two character suffixes you used for the TCPCFG<sub>xx</sub> and DNRCFG<sub>xx</sub> parameter files.
3. Once you complete the previous steps, modify the JCL startup procedure and point to your updated START<sub>xx</sub> member.

## System Commands

This section describes, in reference form, the functions, syntax, and arguments of all the Unicenter TCPaccess system commands.

The most commonly used commands are:

TASK	Displays active task groups.
STATUS	Displays maintenance status of a task group.
P CLEAR	Removes system hooks at shutdown.
STOP <i>TGI</i>	Stops the task group (indicated by <i>TGI</i> ).
START <i>TGI</i> CNFG ( <i>xx</i> )	Starts the task group <i>TGI</i> using CNFG( <i>xx</i> ), where <i>xx</i> is the task group identifier for the task group you are starting.
REFRESH LUPARM ( APPLUP <i>xx</i> )	Refreshes LUPOOL and LU information for Server Telnet usage.

## Command Format

A command consists of a Subsystem Recognition Character (SRC), optionally followed by a Task Group Identifier (TGI), followed by a command verb, a command object, and usually, by one or more operands. Commands are referred to by the command object. These terms are described in the sections that follow. This examples uses % as the SRC:

```
%P CLEAR
```

You can also use these commands with the MVS modify command, specifying the job name. For example:

```
F jobname,P CLEAR
```

**Note:** Commands are limited to 126 characters.

## Subsystem Recognition Character

The SRC provides the method for an MVS subsystem address space to have operator commands directed to it. The person responsible for installing the subsystem sets the SRC and it is specified in the RUNTCP job to start Unicenter TCPaccess. It is one of the parameters used in the execute step for IFSSTART.

The SRC can be any valid SRC supported by JES2 or JES3 (JES2 uses \$, JES3 uses \*). Ask your JES systems programmer which character to use and ensure that it is not a character used by an installation subsystem. The *JES2/JES3 Initialization and Tuning* documentation defines valid SRCs to use when passing commands from local consoles to subsystems (see CONDEF statement, CONCHAR argument, for JES2; and CONSTD statement, SYN argument, for JES3).

SRC is an optional parameter. If no SRC is specified, there will not be an SRC (and all commands issued to the Unicenter TCPaccess address space must be done via the MVS MODIFY command).

## Task Group Identifier (TGI)

Most commands are processed by an implied task group depending on the command object. Some commands provided by Unicenter TCPaccess, such as SNAP, can be directed to a specific task group; in this case, the task group identifier must be placed between the SRC and the verb.

Use the following command to direct the SNAP command to the API task group:

```
TCP SNAP ALL
```

The SNAP command, entered as **SNAP**, is directed to the job step task group (IJT) since the task group identifier is omitted. The keyword ALL indicates to SNAP the IFS trace table.

The following is a list of valid task group identifiers:

TCP	TCP/IP stack.
APP	TCP/IP applications.
DNR	Domain Name Resolver.
IJT	IFS Jobstep Task.
MAP	Port Mapper.
SNM	Simple Network Management Agent Task.
RTM	TelnetRTM .

## Verbs

The following is a list of command verbs and the action taken for each verb:

**Note:** If you do not specify one of the following verbs, DISPLAY is assumed.

- |         |   |
|---------|---|
| DISPLAY | Displays the status of the command object.  |
| MODIFY  | Modifies or changes the value of the command object.<br>The keyword ADD, DELETE, or UPDATE usually appears as an operand in this command. |
| VARY    | Changes the status of the command object in an orderly manner.<br>The keyword ON or OFF usually appears as an operand in this command.    |

Some command objects, such as SNAP, support only the DISPLAY form and ignore any other specified verb.

You can enter verbs spelled exactly as they are shown or you can use an acceptable abbreviation. You can abbreviate any verb by entering only the significant characters. That is, you must type as much of the verb as is necessary to distinguish it from other verbs. DISPLAY, MODIFY, and VARY can be abbreviated as D, M, and V, respectively.

## Objects

Verb action is performed on command objects. SNAP is a command object. The SNAP command can be entered in these ways:

```
DISPLAY SNAP
SNAP
```

You can enter command objects spelled exactly as they are shown or you can use an acceptable abbreviation. You can abbreviate any object by entering only the significant characters. That is, you must type as much of the object as is necessary to distinguish it from other objects.

## Operands

Operands provide the specific information required for the command to perform the requested operation.

These operands for the SNAP command specify either a JES SYSOUT class or, alternately, an MVS SNAP:

```
SNAP CLASS(A)          /* JES SYSOUT CLASS    */
SNAP MVS                /* MVS SNAP          */
```

**Note:** These examples issue the SNAP against the IFS Jobstep Task Group (IJT) by default.

The types of operands used with commands are:

- Positional
- Keyword

Positional Operands      Positional operands follow the command object in a prescribed sequence.

You must replace the TGI with the actual three-character task group identifier when you enter the command.

When you want to enter a positional operand that is a list of several names or values, the list **must** be enclosed within parentheses. The names or values must not include unmatched parentheses.

Keyword Operands      Keywords are specific names or symbols that have a particular meaning to the system. You can include keywords in any order following the positional operands. In the command descriptions within this document, keywords are shown in uppercase characters. A typical keyword is ADD.

Some keywords let you specify values. Place the value inside parentheses following the keyword.

The following is a typical keyword with a value:

```
TG ( TGI )
```

You select the task group identifier desired and substitute that value for *TGI* when you enter the operand, for example:

```
TG ( IJT )
```

**Note:** If conflicting keywords are entered, the last keyword entered overrides the previous ones.

### Abbreviating Keyword Operands

You can enter keywords spelled exactly as they are shown or you can use an acceptable abbreviation.

You can abbreviate any keyword by entering only the significant characters. That is, you must type as much of the keyword as is necessary to distinguish it from the other keywords of the command object.

The SNAP command includes the keywords CLASS and MVS.

- Abbreviations for CLASS are C, CL, CLA, and CLAS
- Abbreviations for MVS are M and MV

In addition, some commands allow unique abbreviations or aliases for some of their keywords.

### Comments

Comments can be added to a command anywhere a blank might appear. Enter them within the comment delimiters, /\* and \*/ as shown below:

```
SNAP C      /* OVERRIDE DEFAULT SYSOUT CLASS */
```

### Delimiters

When you type a command, you must separate the command task group identifier, verb, object, and operands from each other using one or more blanks or a comma.

```
POOL ( SRB XWA )
```

**Note:** Do not use a semicolon as a delimiter. Characters entered after a semicolon are ignored.

## General Task Group Commands

Unicenter TCPaccess general task group commands can be processed by any active task group in a Unicenter TCPaccess address space. If the task group identifier is not specified, the command is processed by the job step task group (IJT). The commands are SNAP and STATUS.

### SNAP

Use the SNAP command for debugging purposes to spin off (dynamically allocate and free) a SYSOUT data set containing a formatted snap dump of control blocks for a task group:

**Note:** The SNAP command is not supported for the MAP task group.

```
[ TGI ] SNAP [ MVS ] [ ALL ] [ CLASS ( SYSOUT_class ) ]
```

**TGI** Specifies the three-character task group identifier of the task group to process the command. If not specified, IJT is assumed.

You can also append the task number to the task group identifier. For example, to snap the control structures for APP task group, task #2:

```
(CellCmdEnv)APP2 SNAP ALL CLASS(X)
```

**MVS** Specifies an MVS formatted snap dump instead of a Unicenter TCPaccess formatted snap dump.

The output goes to the SYSSNAP DD data set that is not dynamically freed. That is, the address space must be stopped to make it available for printing if SYSSNAP is a SYSOUT data set.

**ALL** Include the Unicenter TCPaccess internal trace table in a Unicenter TCPaccess formatted snap dump. Applications can also include extra information.

**CLASS(SYSOUT\_class)** Specifies an override value for the SYSOUT class to dynamically allocate.

The default class is specified by the SOUT= symbolic parameter in the JCL procedure for this address space.

**Examples** The following are examples of the SNAP command:

```
SNAP
SNAP MVS
SNAP ALL CLASS( A )
APP2 SNAP ALL CLASS(X)
```

## STATUS

Use the STATUS command to display the maintenance status of a task group. The version and release numbers are displayed.

```
[ TGI ] STATUS
```

### Examples

The following are examples of this command:

```
STATUS  
APP STATUS
```

## Dynamic Configuration Commands

All the configuration variables contained in the TCPCFGxx member that are processed during startup, can be dynamically reprocessed with a set of operator commands. The net effect can be compared to a limited startup with only selected items from the TCPCFGxx member, but without recycling the entire Unicenter TCPAccess address space. For more details, refer to the description of the TCPCFGxx member in the Unicenter TCPAccess *Customization Guide*.

The LNI and DEVICE commands activate and deactivate local network hardware interfaces.

The DELETE command removes specified items from the active configuration.

The UPDATE command is not a single action command, but instead reads a member containing a list of commands or configuration statements. The configuration statements are identical to those in the startup TCPCFGxx member, and the LNI, DEVICE, and DELETE commands can be included. The default member name is TCPCFGUP, but command syntax allows you to use any name.

The provision to include commands among configuration statements becomes significant where state conditions are enforced. For example, to replace an active LNI, it must first be stopped and then deleted, before the configuration statement is processed. Finally, since this is not in startup, the device must be started manually. The member to the UPDATE command can contain a list of these activities, and the entire sequence can be activated with one command.

The distinction between adding to a configuration or replacing a configuration is based on whether the item itself allows multiples. For example, multiple LNIs are permissible, so a replacement requires a prior deletion. Inversely, a statement like TCP is always a total replacement.

For example, you have a configuration with:

- A MEDIA named CETIETH
- CETI device with the address 884
- DEVICE name CETI0884

You want to replace the CETI controller with one with an address of E50. The MEDIA name came from the NAME keyword on the MEDIA statement, and the DEVICE name was internally generated. Both names can be displayed with a NETSTAT CNFG command.

The typical sequence of steps is:

1. Extract the CETI statement from the TCPCFG $xx$  startup member into a new member (conveniently named TCPCFGUP).

This CETI statement contains the following:

```
DEVADDR(E50),MEDIANAME(CETIETH)
.
.
```

2. Since this is a replacement, the previous LNI must be stopped and deleted. Ahead of the CETI statement, add the following commands:

```
DEVICE STOP NAME(CETI0884)
DELETE LNI NAME(CETI0884)
```

3. Since this is **not** started, LNI activation is not automatic, so following the CETI statement, add the following:

```
DEVICE START NAME(CETI0E50)
```

The DEVICE name is predictable in this case, as the last four characters consist of the hardware channel/device address.

4. Execute the **UPDATE** command.

**Note:** If this is an addition of a new LNI, rather than a replacement of an existing one, Step 2 is not required.

## Commands Imbedded in the TCPCFGUP Member

Dynamic configuration commands can be executed as individual commands (entered from the console), or can be imbedded in the TCPCFGUP member for activation via the UPDATE command where they are executed as individual commands entered from the console. This enables you to combine a series of sequences, such as deletions and additions, into one member, and then execute all of them with one command.

All statements in TCPCFGUP are assumed additions to the existing configuration.

In addition to the startup statements in the TCPCFGUP member, TCPCFGUP can also contain any of the other three dynamic configuration operator commands. When placed in the TCPCFGUP member, these commands are executed as individual commands entered from the console. This enables you to combine a series of sequences, such as deletions and additions, into one member, and then execute all of them with one command.

## DELETE

The DELETE command deletes various configuration components. Specific rules of association and state are enforced. For example, a MEDIA block cannot be deleted until all its NETWORK blocks are also deleted. An LNI cannot be deleted unless it is stopped, and no ingredient belonging to the LOOPBACK configuration can be manipulated at all.

**DELETE** *target* **PARAMETER1** ... **PARAMETERn**

*target*

Item to be deleted.

Choices are:

ARP                      Invalidates an ARP table entry.

DEVICE                  Deletes a device and its associated LNIs.

**Note:** The LNI state must be stopped.

LNI                        Deletes an LNI. If the LNI is the only one associated with a device, the device is also deleted.

**Note:** Driver configurations that support multiple LNIs tied to one device may require several LNI deletions. The current LNI state must be stopped.

	MEDIA	Deletes a MEDIA block.  <b>Note:</b> All associated NETWORKs and LNIs must be deleted prior to issuing a MEDIA deletion.
	NETWORK	Deletes a NETWORK block.
	ROUTE	Deletes a ROUTE table entry.
	VIPA	Deletes (inactivates) a dynamic VIPA that was activated via SIOCSVIPA IOCTL or by an implicit BIND to a specific address.
	VIPANET	Deletes a subnet in which requests to activate a dynamic VIPA, via BIND or SIOCSVIPA IOCTL, are honored.
PARAMETER	The type of target.  Choices are:	
	DEST( <i>value</i> )	Dotted decimal IP address. Must be specified when deleting a ROUTE entry. See ROUTER, below.
	IPADDRESS( <i>value</i> )	Dotted decimal IP address. Must be specified whenever a protocol address is required.  Used to identify a NETWORK block, dynamic VIPA, VIPANET, ARP entry, or ROUTE entry.
	Alias: PA	
	MAC( <i>hex_value</i> )	12-character hexadecimal string specifying the six-byte hardware address used when deleting ARP entries.
	MEDIANAME( <i>name</i> )	Must be specified whenever there is an association with a specific MEDIA block. Use when deleting ARP or ROUTE entries.  <b>Note:</b> A MEDIA block deletion can be identified by either
	NAME( <i>name</i> )	Must be included whenever a specific identification of a DELETE target is required. Use when deleting MEDIA, LNI, or DEVICE entries.

ROUTER( *value* )      Dotted decimal IP address; must be specified when deleting a ROUTE entry.

Both DEST and ROUTER parameters are functionally identical to those on the ROUTE statement in the TCPCFGxx startup member.

Alias: GATEWAY

## DEVICE

The DEVICE command changes the operational status, but not the administrative status of the device. When a device is stopped using this command, the operational status is changed to down, but the administrative status is still up. Therefore, the device automatically starts when the restart time expires (see driver statements in the Network Configuration chapter of the *Configuration Guide*).

When the device is using the LCS driver, the LCS statement is the DEVICE and each LINK statement is an LNI. For all other drivers, there is a one-to-one correspondence between DEVICE and LNI.

The internally generated names for both DEVICES and LNIs are available via the NETSTAT CNFG command.

**DEVICE START | STOP NAME ( *dev\_name* )**

START                      Initiates activity on a DEVICE. The current state must be stopped.

STOP                        Terminates device activity.

The current state must be active.

NAME( *dev\_name* )      DEVICE name.

If the name is not known, issue the following command to identify it:

**NETSTAT CNFG**

## LNI

The LNI command changes the operational status and the administrative status of the LNI. When an LNI is stopped using this command, both the operational status and the administrative status are changed to down. Therefore, the LNI must be restarted using the LNI command.

When the device is using the LCS driver, all LNIs must be stopped before the device is stopped. To do so use the following command:

```
LNI START | STOP NAME ( lni_name )
```

START Initiates activity on an LNI driver.

The current state must be stopped.

STOP Terminates driver activity.

The current state must be active.

NAME(*lni\_name*) Name of the LNI.

If the name is not known, issue the following command to identify it:

```
NETSTAT CNFG
```

## UPDATE

The UPDATE command behaves differently from the other dynamic configuration commands. It contains no information itself, but activates commands in a member.

The UPDATE command also allows for some items that are not in the startup file, in addition to the other three dynamic configuration commands.

The UPDATE command adds new configuration data to an operational gateway. The parameters for the UPDATE command specify a member name that contains configuration statements as they would appear in the startup member TCPCFG $xx$ . The update member name can default to TCPCFGUP, and must be available in the same DD definitions as the TCPCFG $xx$  member.

For more details, see the description of the TCPCFG $xx$  member in the *Customization Guide*.

```
UPDATE [ CNFG ( mem_name ) ] [ MEMBER ( mem_name ) ] [ IGNORE | TERM ]
```

CNFG( *mem\_name\_suf* ) Two-character suffix for a member name starting with TCPCFG.

Alias: CFG, CONFIG

Default: UP.

MEMBER( *mem\_name* ) The entire member name.

If the default name (TCPCFGUP) is not used, either CNFG or MEMBER should be specified. If both are specified, CNFG is ignored.

Alias: MBR

Default: TCPCFGUP.

IGNORE | TERM Dictates the reaction when encountering an invalid statement in the TCPCFGUP member:

IGNORE Issues an error message and proceeds to the next statement.

TERM Issues an error message and terminates.

Default: TERM.

## ARM Command

ARM *request* Address space operator command.

The valid *request* types are:

STARTI	Synonym for the ENABLE, LISTEN, REGISTER, and WAITPRED commands, issued in that order.
ENABLE	Enables ARM processing for the address space, and allows the issuing of other commands.
LISTEN	Requests the address space to listen for ARM-related ENF events.
REGISTER	Registers the address space as an ARM element in the sysplex.
WAITPRED	Requests that the address space wait for its ARM predecessors to become ready.
READY	Makes the address space eligible for restart by ARM.
STOP	Synonym for the DISABLE, NOLISTEN, and Deregister commands, issued in that order.
DISABLE	Disables ARM processing for the address space. When it completes, any further ARM-related ENF events are ignored.
NOLISTEN	Requests the address space to stop listening for ARM-related ENF events.
DEREGISTER	Deregisters the address space as an ARM element in the sysplex.

Normally, you will not need to issue ARM commands – IFS automatically performs the required ARM actions.

## REFRESH Command

This section describes the REFRESH command, which is processed by either the APP task group or the TCP task group.

### REFRESH

Use the REFRESH command to refresh certain configuration parameters of the APP task group. It can be used to refresh the LU pool, greeting member, the USS Table used by Server Telnet, or the bind security configuration parameters.

```
[ APP | TCP ] REFRESH TASK( n ) [ LUPARM( mem_name ) | GREETING( mem_name ) |
USSTAB( tbl_name ) | BINDSEC ( mem_name ) ]
```

**APP | TCP** Specifies the name of the task group to which the command is to be directed. If you specify the BINDSEC keyword, the command must be directed to the TCP task group. All other REFRESH commands must be directed to the APP task group. The default, if not specified, is to direct the command to the APP task group.

Default: APP.

**TASK ( *n* )** Specifies the task number of the task within the APP task group to which the command is directed. For commands directed to the TCP task group, this keyword is ignored.

Default: Task number one.

**LUPARM(*mem\_name*)** The LU pool is refreshed from this member.

**GREETING(*mem\_name*)**

Specifies the name of the member of the ARPAHELP data set from which the new server telnet greeting will be read.

Subsequent Telnet sessions will be presented with the greeting found in this member.

**USSTAB (*tbl\_name*)** Specifies the USS table (*tbl\_name*) coded in the active APPCFGxx SERVICE segment should be refreshed.

**BINDSEC ( *mem\_name* )**

Specifies the name of the member of the SYSPARM configuration data set from which the refresh is performed. The bind security configuration parameters are refreshed from this member.

The REFRESH BINDSEC command will not affect existing, running applications. The new configuration will only take effect upon the start of an application. Existing applications will have to be stopped and restarted for the new configuration to take effect.

**Note:** LUPARM, GREETING, USSTAB, and BINDSEC are mutually exclusive.

### Examples

```
REFRESH LUPARM ( APPLUP00 ) TASK ( 2 )
APP REFRESH TASK ( 1 ) GREETING( GREETING )
REFRESH USSTAB ( T01USS01 ) TASK ( 1 )
TCP REFRESH BINDSEC ( TCPBND00 )
```

## DNR Commands

This section describes DUMP and PURGE commands that are processed by the DNR task group.

### DUMP

Use the DUMP command to produce formatted dumps of the:

- DNR cache
- Configuration table

These dumps are written to DNRLOG.

```
[ DNR ] DUMP [
  CACHE [ ( DATA ( xxxx ) | NAMES ) ] | NAMESERVER ( xxxx )
  STATIC [ (
    ALIAS | HOST | NAMESERVER | NETPREF | NETWORK | RPC |
    PROTOCOL | SEARCHLIST | SERVICES
  )
] ]
```

#### CACHE

Specifies that DNR cache data is to be dumped:

**DATA** Specifies the host name or Internet address for which a cache dump is to be performed.

**NAMES** Specifies that all cached domain names be dumped. This is the default if CACHE is specified with no operands.

#### NAMESERVER

Specifies that, for the specified domain name, the name servers used to resolve names within that domain are to be dumped.

**STATIC** Specifies that static configuration data should be dumped. If no argument is specified, all static configuration data is dumped.

The available arguments are:

<b>ALIAS</b>	Dump DNRALCxx configuration data.
<b>HOST</b>	Dump DNRHSTxx configuration data.
<b>NAMESERVER</b>	Dump DNRNSCxx configuration data.
<b>NETPREF</b>	Dump DNRNPCxx configuration data.
<b>NETWORK</b>	Dump DNRNETxx configuration data.
<b>RPC</b>	Dump DNRRPCxx configuration data.
<i>PROTOCOL</i>	<i>Dump DNRPRTxx configuration data.</i>
<i>SEARCHLIST</i>	<i>Dump DNRSLCxx configuration data.</i>
<i>SERVICES</i>	<i>Dump DNRVCxx.</i>

**Default** If the DUMP command is issued with no arguments, then all cache and static configuration data is dumped.

**Usage Guidelines** The DNR task group identifier is optional; the DUMP command is automatically directed to the DNR task group, even if it is omitted.

**Examples** The following are examples of the DUMP command:

```
DNR DUMP CACHE
DNR DUMP CACHE( DATA( A.OUR.COM. ) )
DNR DUMP NAMESERVER( OUR.COM. )
DNR DUMP STATIC
DNR DUMP STATIC( RPC )
```

## PURGE

The PURGE command removes all entries from the DNR cache. DNR must then access your name server to resolve addresses while the cache is rebuilt. Use this command when your DNR cache contains entries that are no longer valid and you want them refreshed immediately.

```
[ DNR ] PURGE
```

## GateD (GATED) Commands

This section describes commands processed by GateD, formerly known as GateD (GTD) task group.

**Note:** GateD task does **not** automatically restart after a STOP command. Use the GATED START command to restart GateD:

```
GATED START [ CNFG( config_mem_name ) ]
```

The following is a summary of GATED commands:

DUMP	Generates a formatted dump. <b>GATED DUMP</b>
RELOAD	Reinitializes GateD—It rereads the GTDCFGxx configuration file. <b>GATED RELOAD</b>
DUMP	Generates a formatted dump.
SCAN	Performs an immediate rescan of the interface. <b>GATED SCAN</b>
START	Starts the GateD subtask, using the configuration member name supplied. If no name is supplied, it uses the one specified on the GATED parameter of the IP statement of TCPCFGxx. If there is no GATED parameter, an error is returned. <b>GATED START [ CNFG( <i>config_mem_name</i> ) ]</b>
STOP	Shuts down GateD. <b>GATED STOP</b>
TRACE	Suspends or resumes tracing (in other words, this is a toggle). <b>GATED TRACE</b>

## IJT Commands

This section describes commands processed by the job step (IJT) task group.

### GTF

Use the GTF command to display or modify the settings of GTF trace event flags. A trace event is recorded only when an event is turned on (with this command) and the task group executing a module that invokes a trace event is in GTF mode (for more information, see [SET](#)). If there is no job name the trace applies only to the TCP address space.

```
[ DISPLAY | MODIFY ] GTF [ ON | OFF ]
[ EI ( event_id [ ... ] ) | CB ( cb_id [ ... ] ) | MOD ( mod_name [ ... ] ) | ALL
]
```

DISPLAY   MODIFY	Specifies whether to display or modify the settings of GTF trace event flags.
ON   OFF	Select only those events that are turned on or off for display, or to turn on or off a specified event(s) for modify.  If neither ON nor OFF is specified with DISPLAY, the on/off state of an event is not considered for inclusion in the display.  <b>Note:</b> Either ON or OFF is required with MODIFY.
EI ( <i>event_id</i> )	Select the event identifiers listed (one- to eight-alphanumeric characters).
CB ( <i>cb_id</i> )	Select the events for the control block identifiers listed (one- to four-alphanumeric characters).
MOD ( <i>mod_name</i> )	Select the events generated by the modules listed (one- to four-alphanumeric characters).  <b>Note:</b> A name stem is permitted. That is, if IFSP is specified, all modules having names starting with IFSP are included.
ALL	Select all events.
Examples	The following are examples of this command:

```
GTF OFF
GTF EI( MESSAGE CALLPC SRBDISP )
GTF CB( SSOB ISRB )
MODIFY GTF OFF ALL
MODIFY GTF ON CB( MODI SDWA )
DISPLAY GTF
```

## HELP

Use the HELP command to get online information about the function, syntax, and operands of commands. This reference information is contained in the SYSHELP DD data set(s) and is displayed on your console in response to your request for help.

Enter HELP:

- Without operands to get an introduction to using the help facility
- With the operand COMMANDS to view a list of all the Unicenter TCPaccess commands for which help is available

**HELP [ *cmd\_name* | COMMANDS | GENERAL POOL ]**

*cmd\_name* Specifies the full name of a command for which help information is requested.

COMMANDS Requests a list of all commands for which help information is available.

GENERAL Requests a display of the general format and syntax of Unicenter TCPaccess commands.

POOL Requests a display of Unicenter TCPaccess pools.

Example The following are examples of this command:

```
HELP
HELP COMMANDS
HELP POOL
```

## IFS

Use the IFS command to display environmental settings for the address space and to display key subsystem-related control block addresses.

**IFS**

This command has no arguments or keywords.

## ILATCH

Latches are locking mechanisms that serialize resources more granular than an address space. For example, use a latch to serialize resources in the local address space.

**Note:** Unicenter TCPaccess latches do not use the IBM latch facilities.

If a latch is allocated by a program but not freed, it may cause other programs requesting that same latch to hang.

The IJT command ILATCH displays and frees latches used by UNIX System Services (GateD) sockets.

The alias for the ILATCH command is ILA.

There are three different versions of the ILATCH command:

ILATCH

ILATCH CONTENTION

ILATCH TRON

## ILATCH

```
ILATCH [ DISPLAY | FREE ] [ TIME ( seconds ) ] [ LATCH ( latch_num ) ]
      [ MSG | NOMSG ]
```

**DISPLAY** Displays all latches held more than the time specified by TIME.

It can be restricted to display a specific latch number (*latch\_num*) if it was held more than the specified time.

Default: DISPLAY.

**FREE** Frees all latches or the latch specified by *latch\_num* if held for more than the time specified by TIME(*seconds*).

**TIME(*seconds*)** Specifies the time, in seconds that the latch has been held.

Default: 60 seconds.

**LATCH(*latch\_num*)** Specifies the latch number (*latch\_num*).

Default: All latches.

MSG | NOMSG

MSG Specifies that the message “T00IF041 END OF ILATCH COMMAND” should display.

NOMSG Suppresses that message.

Default: DISPLAY TIME( 60 ) MSG.

### ILATCH CONTENTION

ILATCH CONTENTION [ MSG | NOMSG ]

CONTENTION Lists the history of prior latch contentions.

MSG | NOMSG

MSG Specifies that the “T00IF041 END OF ILATCH COMMAND” should display.

NOMSG Suppresses that message.

Default: DISPLAY TIME( 60 ) MSG.

### ILATCH TRON

ILATCH TRON | TROFF | TRACE | TRESET [ ENTRIES ( *nn* ) ] [ KEEP | NOKEEP ]  
[ MSG | NOMSG ]

TRON Activates ILATCH tracing.

TROFF Stops adding to existing table entries.

The table itself is not released, and is still available for display.

TRACE Displays table entries.

The RC is the return code leaving the GET request.

All other displayed information is based on entry to the GET routine.

TRESET Release the storage.

This command is not required between a TRON/TROFF cycle and another TRON activation with new parameters, but should be used for clean up after all tracing is done.

ENTRIES(*nn*) Dictates the table size (the number of entries you want to monitor).

Default: 31.

**Note:** This may not be sufficient if KEEP is specified.

KEEP | NOKEEP

KEEP Controls whether tracing provides a historic record or keeps track of unreleased latches.

NOKEEP Specifies not to retain released entries. The NOKEEP table can be relatively small.

Default: NOKEEP.

MSG | NOMSG

MSG Specifies that the message T00IF041 END OF ILATCH COMMAND should display.

NOMSG Suppresses that message.

Default: DISPLAY TIME( 60 ) MSG.

## LOGGING

Use the LOGGING command to reparse and update the entire LOGGING statement, as contained in the IJTFCGxx startup member. Any parameter can be changed, within valid limits, and the entire statement can be reprocessed on an active gateway.

```
LOGGING [ CLASS ( class ) ]
         [ DEST ( destination ) ]
         [ NOW ]
         [ PRINT ( subparm [ , subparm [ , ... ] ] ) ]
         [ ROUTCDE ( list ) ]
         [ SPIN ( LINES ( lines ) | MINUTES ( minutes ) | SYNC ) | NOSPIN ]
         [ WTO ( subparm [ , subparm [ , ... ] ] ) ]
```

CLASS (*class*) Specifies the SYSOUT class.

Default: Class specified as SOUT= keyword of PARM field.

DEST ( *destination* ) Specifies the SYSOUT destination.

Default: No destination.

**NOW** When this parameter is issued through the console, an immediate logspin is performed.

**Note:** NOW works in conjunction with the SPIN parameter.

**PRINT ( subparm)** Subparameters are processed left to right.

Valid values:

ALL WTO/PRINT all messages, all types.

NONE WTO/PRINT no messages.

(ALL, types) WTO/PRINT given types for all components.

(component, ALL) WTO/PRINT all messages for given component.

(component, NONE) WTO/PRINT no messages for given component.

(component, types) WTO/PRINT given messages for given component.

**ROUTCDE (list)** Specifies the MVS routing codes for console messages.

*list* One or more valid MVS routing codes, separated by commas. You specify routing code ranges by separating them with a hyphen.

IFSPARM LOGGING ROUTCDE(2)

IFSPARM LOGGING ROUTCDE(3,4,8-11)

IFSPARM LOGGING ROUTCDE(9-11)

Range: 1-16.

Default: No routing code.

**Note:** This means console messages are routed according to the defaults specified in the MVS SYSGEN.

---

SPIN ( LINES (*lines*) | MINUTES (*minutes* | SYNC ) | NOSPIN

Determines when the log file closes and reopens.

LINES                      Number of lines to be written to each log file before it is closed and reopened.

MINUTES                    Duration of time before logout is done.

Alias for MINUTES is TIME.

SYNC                        Synchronizes to the hour.

Default: NOSPIN.

WTO ( *subparameter* )    Subparameters are processed left to right.

Valid values:

ALL                         WTO/PRINT all messages, all types.

NONE                        WTO/PRINT no messages.

(ALL, types)                WTO/PRINT given types for all components.

(component ,ALL)          WTO/PRINT all messages for given component.

(component, NONE)        WTO/PRINT no messages for given component.

(component, types)        WTO/PRINT given messages for given component.

You can also change logging dynamically with the MODIFY command. For example:

**MODIFY *job\_name* LOGGING *parameter(s)***

## MODULE

Use the MODULE command to display information about a resident module such as call count and assembly date and time.

**MODULE [ ( *mod\_name* [ ... ] ) | ALL | \* ]**

*mod\_name* [ ... ]      Module name(s) to display (one- to eight- alphanumeric characters).  
ALL or \*                Displays all resident modules.

Examples                The following are examples of this command:

**MODULE \***  
**MODULE IFSSCALL IFXPOST**

## MEM

Use the MEM command to display up to 1024 bytes of virtual storage. There are two versions of the command:

MODIFY MEM  
DISPLAY MEM.  
**MODIFY MEM *addr Vdata Rdata***

*addr*                    Storage address to alter, expressed as a hex string.  
*Vdata*                  Value of existing data at that location, expressed as a hex string.  
*Rdata*                  Value of replacement data, expressed as a hex string

Usage Notes            1. Storage in key zero cannot be modified.  
                          2. *addr*, *Vdata*, and *Rdata* must contain an even number of characters 0-9, A-F. All three are mandatory.  
                          3. *Rdata* and *Vdata* can be 2 - 512 characters long, or 1 - 256 data bytes.  
                          4. *Rdata* may be longer than *Vdata*.

**DISPLAY MEM *addr* | \* [ DECLN ( *nnn* ) | HEXLEN ( *xxx* ) MOD ( *mod\_name* ) ]**  
*addr* or \*                Starting display address.

It can be entered as an explicit address or as an asterisk (\*) with a module name (MOD(*mod\_name*)) parameter

MOD(*mod\_name*)        Module name.

---

DECLEN | HEXLEN    Length. It can be specified as decimal (DECLEN(*mmm*)) or hexadecimal (HEXLEN(*mmm*)), with a maximum of 1024.

If not specified, default value = 16.

Alias for DECLEN is LEN (abbreviated as DECL).

**Note:** HEXLEN has no alias (abbreviated as HEXL).

## MVS

Use the MVS command to display the MVS environment and, optionally, the contents of selected control blocks.

**MVS [ IFS | JESCT | LNKLST | SCVT | SMCA | SSCT ]**

**IFS**                    Displays the subsystem communication vector table address and name of each defined IFS-based subsystem.

**JESCT**                Displays the address of the JES control table and the name of the primary JES.

**LNKLST**              Displays the names of the data sets in the MVS Link Library List.

**SCVT**                Displays, in dump format, the MVS secondary communications vector table.

**SMCA**                Displays, in dump format, the SMF Control Area.

**SSCT**                Displays the subsystem communication vector table address and name of each defined subsystem.

**Examples**            The following are examples of this command:

**MVS**  
**MVS SSCT**

## P

The P command terminates all task groups and the address spaces. Optionally, it removes the subsystem hooks installed by this address space at initialization.

**P [ CLEAR ]**

**CLEAR** Specifies to clear the subsystem hooks installed by this address space before returning to MVS.

To update Unicenter TCPAccess after applying maintenance, use the CLEAR option to stop Unicenter TCPAccess before applying updates. In addition to providing normal stop processing, this also clears control blocks and certain modules in the CSA and lets Unicenter TCPAccess use the most up-to-date versions of the base modules when it is restarted.

**Note:** S0C1/S0C4 messages during termination after a P CLEAR are normal and can be ignored. Only one P CLEAR is needed to clear the CSA.

Examples The following are examples of this command:

```
P
P CLEAR
```

## POOL

Use the POOL command to display the statistics or attributes of data area pools. A pool is a collection of fixed-length data areas residing in a single MVS storage subpool managed by Unicenter TCPAccess without the overhead of GETMAIN/FREEMAIN.

**POOL [ ( *pool\_name* [ ... ] ) | \* ] [ ATTR ]**

*pool\_name* Specifies the name of pool(s) to be displayed.

Options are:

ATCB	Address space task block.
DSRB	Domain Name Resolution Request Block.
FRR	IFS Recovery Element.
IPTH	IUCV only, path to TCP.
MB1	Buffer pool for moving data - 128 bytes.
MB2	Buffer pool for moving data - 384 bytes.
MB3	Buffer pool for moving data - 640 bytes.

MB4	Buffer pool for moving data - 1152 bytes.
MB5	Buffer pool for moving data - 1536 bytes.
MB6	Buffer pool for moving data - 2048 bytes.
MB7	Buffer pool for moving data - 5120 bytes.
MB8	Buffer pool for moving data - 10240 bytes.
MB9	Buffer pool for moving data - 16384 bytes.
MBA	Buffer pool for moving data - 32768 bytes.
MSRB	Message Service Request Block.
MWA	Module Work Area.
QCB	Queue Control Element for pools.
RTPB	TelnetRTM port block.
RTOB	TelnetRTM owner block.
SAW	Socket API function.
SEPM	Socket endpoint.
SNMP	SNMP data.
SPCB	Transport Provider (only three required total).
SRB	IFS Service Request Block.
STAK	Module Stack Block for work areas.
XAE	SNMP Request/Response header.
XWA	Cross Memory Work Area.

**ATTR** Specifies to display the pool attributes instead of statistics. Pool attributes are the values used to control expansion and minimum or maximum values. (Set by the POOLDEF configuration parameter statement or the default.)

Default values for the POOL options are described in the following table.

<b>Value</b>	<b>Initial</b>	<b>Minimum</b>	<b>Expand</b>	<b>Contract</b>
ATCB	32	32	16	32
FRR	100	200	50	0
IPTH	64	64	32	64
MB1	32	128	16	128
MB2	32	256	16	256
MB3	32	192	16	128
MB4	32	160	16	128
MB5	16	128	8	64
MB6	16	96	4	32
MB7	8	48	8	16
MB8	4	32	4	16
MB9	4	16	4	16
MBA	2	8	2	8
MSRB	200	400	100	0
MWA	100	200	50	75
QCB	100	200	50	0
RTPB	4	8	4	4
RTOB	32	256	64	128
SAW	64	512	32	256
SEPM	16	256	32	128
SPCE	3	3	1	3
SRB	100	200	50	0
STAK	40	20	20	0
XWA	112	160	24	0

The DSRB, SNMP, and XAE pools have no defaults; you must specify explicit values for them.

**Note:** It is best to use the defaults at first, and issue the POOL command every so often to display pool usage. If you find pools being expanded and staying at the higher value, you can override the default and specify a higher minimum value.

#### Examples

The following are examples of this command:

```
POOL
POOL SRB ATTR
POOL * ATTR
```

## SET

Use the SET command to set execution options for a task group.

```
SET [ DEMO | TEST | GTF ]
    [ ON | OFF ]
    [ TG( TGI [ ... ] ) | ALL ]
```

DEMO | TEST | GTF Specifies which mode to set:

DEMO Some messages or processing is performed only in DEMO mode.

TEST Some messages or processing is performed only in TEST mode. Some API and APP trace events are recorded only in TEST mode.

GTF GTF events are recorded only when GTF mode is on.

ON | OFF Specifies to turn the specified mode on or off.

TG( TGI [ ... ] ) | ALL Specifies the three-character task group identifier affected by the mode change or ALL.

If not specified, all active task groups are affected.

#### Examples

The following are examples of this command:

```
SET DEMO ON
SET GTF OFF TG( APP DNR )
SET TEST OFF TG( IJT )
SET GTF ON
```

## SRC

Use the SRC command to display or modify the subsystem recognition character for the address space.

```
[ DISPLAY ] SRC
MODIFY SRC [ char ]
```

*char* Specifies the new subsystem recognition character (one alphanumeric) for the address space.

The change is effective immediately. This must be a character acceptable by MVS console services. If no character is entered, there is no subsystem recognition character.

Examples The following are examples of this command:

```
SRC
MODIFY SRC #
DISPLAY SRC
```

## START

Use the START command to start a task group and to specify initialization parameter overrides.

```
START      [ TGI CNFG( xx ) MEMBER( mem_name ) ]
           [ CNFG( xx ) ]
           [ MEMBER( mem_name ) ]
```

*TGI* Specifies the three-character task group identifier of the task group to start.

CNFG(*xx*) Specifies the two-character suffix to use in constructing the name of the configuration parameter file.

The name takes the form TGICFGxx, where TGI is the task group identifier specified with the STARTxx command.

**Note:** If CNFG() is not specified, a value of 00 is assumed for xx.

The configuration parameter member resides in the SYSPARM DD data sets and provides site-specific configuration parameter values for the task group being started.

MEMBER(*mem\_name*) Specifies the name of a command script to invoke when task group initialization completes (one- to eight-alphanumeric characters).

The command script resides in the SYSPROC DD data sets.

Examples The following are examples of this command:

```
START DNR CNFG( 01 )
START APP MEMBER( APPCFG05 )
```

## STCK

Use the STCK command to convert the binary eight-byte clock value to a useful date and time.

```
STCK X'hex_string'
```

*hex\_string* Specifies the eight-byte binary clock value as stored by the STCK instruction.

This value is expressed as a 16-character hex string within quotes.

Example The following is an example of this command:

```
STCK X'AF82198271FB4401'
```

## STOP

Use the STOP command to terminate a task group. The task group performs an orderly shutdown before terminating. Static control blocks are left in a state for reuse if the task group is started again. Depending on the specific task group, the stop request can be delayed to let work in progress complete.

```
STOP [ TGI ]
```

```
TASK( n )
```

**Note:** You cannot STOP a TCP task group.

The TASK parameter is valid only when you stop an APP task group.

You can have up to four APP task groups active concurrently.

*TGI* Specifies the three-character task group identifier of the task group to be terminated.

### Usage Guidelines

A second STOP command performs fast shutdown.

A third STOP performs CANCEL shutdown.

### Example

```
STOP DNR
STOP APP TASK( 1 )
```

## SVCDUMP

Use the SVCDUMP command to generate a system formatted dump.

```
SVCDUMP [ ASID ( asid_list ) ]  
        [ JOBname ( jobname_list ) ]  
        [ GRPLIST ( group_list ) ]
```

**ASID ( *asid\_list* )** A list of address space IDs. Unicenter TCPaccess includes these address spaces in the dump.

You can combine this parameter with the JOBname parameter for five address spaces.

The address space IDs can be listed in decimal or in hexadecimal notation, as x'999'.

**JOBNAME ( *jobname\_list* )**

A list of one- to eight-character job names.

Unicenter TCPaccess includes these address spaces in the dump.

You can combine this parameter with the ASID parameter for five address spaces.

**Note:** If there is more than one job active in the system with the same job name, Unicenter TCPaccess dumps the first one found in the Address Space Vector Table.

**GRPLIST( *group\_list* )** The name of one or more XCF groups whose member systems are included in the SVC dump.

This parameter allows SVC dumps to be requested for remote systems in the sysplex. When you specify this parameter, an SVC dump is captured on each system in the sysplex for which a member is active in one or more of the XCF groups specified.

The value specified must be from one- to eight-bytes in length and can be expressed as a quoted character string ('ABC') or a non-quoted character string where a separator or delimiter indicates the end.

The names specified can include wild card characters with the character ? (question mark) denoting a single mask character and the character \* (asterisk) denoting zero or more mask characters.

## TASK

Use the TASK command to display the active task groups. Information displayed includes execution-related flags, dispatch count, and the date/time of task group initialization.

```
TASK [ ( TGI [ ... ] ) ]
```

*TGI* Specifies the three-character task group identifier of the task group to display.

Default: If *TGI* is omitted, all task groups are displayed.

Examples The following are examples of this command:

```
TASK
TASK ( TCP DNR )
```

## TIME

Use the TIME command to display the current date and time in all useful forms. This command uses no parameters.

## TRACE

Use the TRACE command to:

- Display current trace table status
- Turn internal tracing on or off
- Set the internal trace table size

The trace table is formatted and included in an IFS-formatted snap dump if an ABEND occurs.

```
[ DISPLAY ] TRACE
MODIFY TRACE [ ON | OFF ] [ SIZE( number | 16 ) ] [ FIXED ]
```

ON | OFF Specifies to enable or disable internal tracing.

SIZE( *number* ) Specifies the size (*number*) of the trace table as a number of four KB (4096) pages (one or two decimal digits). Each entry in the trace table is 64 bytes long. The maximum number of four KB pages is 2048.

Use the SIZE parameter only when ON is also specified.

Default: 16 4 KB pages(64 KB).

FIXED Specifies that the trace table be in fixed storage, allows trace capture of I/O-related events.

Normally this facility is not required, but you can request it for diagnostic purposes.

Examples

The following are examples of this command:

```
MODIFY TRACE OFF
TRACE
MODIFY TRACE ON
MODIFY TRACE ON SIZE( 8 )
```

## VSM

Use the VSM command to display virtual storage usage statistics.

**Note:** This command uses no parameters.

## TSO Commands

This section describes the TSO commands available for Unicenter TCPaccess UNIX System Services socket users.

### CONVXL8 Command

The TSO command CONVXL8 converts a table from editable text to binary.

CONVXL8 creates a data set with three records. Each record is 256 bytes in length.

- The first record has “\*TCP/IP translate tables” (in EBCDIC) starting in column one, with the remainder of the record padded with EBCDIC blanks (X'40').
- The second record has 256 EBCDIC values representing the ASCII-to-EBCDIC translation.
- The third record has 256 ASCII values representing the EBCDIC-to-ASCII translation.

File names use TSO prefix as defined by TSO rules.

- A fully qualified data set name needs to be enclosed in quotes.
- A data set name without quotes may have a user-specified prefix placed before the name. This prefix is defined by the user's TSO profile.

Refer to TSO documentation for more information about prefixes.

**CONVXL8** *INPUT OUTPUT*

---

<i>INPUT</i>	<p>Specifies the source data set to convert. The data set must be in standard IBM format for SBCS translation tables. This parameter is required</p> <p><b>Note:</b></p> <ul style="list-style-type: none"><li>■ If input is a PDS member, <i>INPUT</i> should be specified as <i>dsname(member)</i>.</li><li>■ Unicenter TCPaccess translate tables in the SAMP data set are not in this format. Use the TSO command LOADXL8 to prepare them for use with UNIX System Services. Refer to the following section, <a href="#">LOADXL8 Command</a>, for more information.</li></ul> <p>Default: None.</p>
<i>OUTPUT</i>	<p>Specifies the output data set created by the conversion.</p> <p>This parameter is required.</p> <p>If output is a PDS member, <i>OUTPUT</i> should be specified as <i>dsname(member)</i>.</p> <p>Default: None.</p>
Examples	<p>This command reads USER.LIB.SOURCE(TRANS) and creates a translate table in USER.LIB.TRANTAB:</p> <pre>CONVXL8 LIB.SOURCE(TRANS) LIB.TRANTAB</pre> <p>This command reads SYSTEM.TCP.DATA(TRAN) and creates a translate table in SYSTEM.BIN.TRANS:</p> <pre>CONVXL8 'SYSTEM.TCP.DATA(TRAN)' 'SYSTEM.BIN.TRANS'</pre>

## LOADXL8 Command

LOADXL8 has the same functionality as the CONVXL8 command, but it reads the load module from compiled translate tables as input.

**Note:** It does not read the source. It loads the module from STEPLIB or TSO TASKLIB.

The load module referred to is the Unicenter TCPaccess translate table used by the rest of Unicenter TCPaccess. It can be converted for UNIX System Services use with this command:

**LOADXL8 MODULE OUTPUT**

*MODULE*

Specifies the load module name in the STEPLIB/TASKLIB data sets.

**Note:** You do not explicitly specify the data set name in the command line; just the member name.

Default: None.

*OUTPUT*

Specifies the destination of the output data set created by the conversion. If output is a PDS member, OUTPUT should be specified as dsname(*member*).

This parameter is required.

Default: None.

These output files are used by UNIX System Services DNR services only. Unicenter TCPaccess continues to use its own translation load modules within the product. You can convert several members and place them in a PDS but UNIX System Services uses only the one placed in the sequential data set 'PREFIX.STANDARD.TCPXLBIN'.

**LOADXL8 ENGLISH 'PREFIX.STANDARD.TCPXLBIN'**

## Command Scripts

This section describes how to build command scripts that are a prearranged executable sequence of IFS task group commands and command script statements that you can invoke by specifying the command script name prefixed with a percent sign (%). The START $xx$  PARM member is an example of a command script.

Command scripts are read from the SYSPROC DD data sets and a sample START00 script is provided in the PARM data set. You can invoke a command script at address space initialization by specifying the command script name with the CMND= parameter in the JCL procedure for an IFS address space. Command scripts can also be invoked using the IJT START command with the MEMBER( $xxx$ ) option.

The command script to start Unicenter TCPaccess is member START $xx$  in the PARM data set. The command data set is specified by DDNAME SYSPROC in the runtime JCL. Member START $xx$  is specified by CMND=START $xx$  in the PARM field on the EXEC statement. Member START $xx$  was established during installation and customization as described in the Unicenter TCPaccess Communications Server *Customization Guide*. Alternate START $xx$  members can be created and overridden on the PROC CMND field. That is:

```
S RUNTCP,CMND=START99.
```

## Sample Command Script

The following is a sample command script to start GTF and turn tracing on for some events in a task group named TGI, and then to invoke another command script named TGICMNDS:

```
SET GTF OFF                /* TURN GTF MODE OFF FOR ALL TASK GROUPS    */
SET GTF ON TG(TGI)        /* TURN GTF MODE ON FOR TGI                  */
MOD GTF OFF ALL           /* TURN OFF ALL GTF EVENTS                   */
MOD GTF ON CB(SDWA PARM MODI) /* TURN ON DESIRED EVENTS                     */
%TGICMNDS                 /* INVOKE COMMAND SCRIPT FOR TGI TASK GROUP  */
```

Special command statements valid only within a command script are provided to perform these tasks:

- Control the display of command statements before being executed
- Control command statement processing if a command fails

The special commands are as follows:

#### FLUSH | NOFLUSH

**FLUSH** Specifies that the command input stack be purged (flushed) when execution of a command statement fails.

This is useful to suppress further command execution if a critical command in a sequence fails.

**NOFLUSH** Specifies that the command input stack continue to be processed even if execution of a command statement fails.

Default is NOFLUSH.

#### LIST | NOLIST

**LIST** Specifies that command statements should be displayed before execution.

**NOLIST** Specifies that command statements not display.

Default is LIST.

### Notes

- The SRC should not be specified.
- A complete command statement must be contained in one input source record.
- Comment statements can be included anywhere. Start a comment statement with an \* in column one or by enclosing the comment in /\* and \*/ (for example: /\* *comment* \*/).
- Sequence numbers, if included in source input, are assumed to be in the last eight columns for fixed-length records and in the first eight columns for variable-length records. Otherwise, the entire record is assumed to contain text.

# Using the System Management Facility (SMF)

This chapter provides samples of SMF record formats and discusses the following topics:

- [Standard MVS SMF Record Header](#) – Sample of the standard MVS record header
- [Standard](#) – Tables for descriptor, product, task identification, and user identification sections
- [Server Application Event Records](#) – Tables for descriptor, common data, Subtype 20, and Subtype 21 data sections
- [Access Method and Protocol Layer Event Records](#) – Table describing the SMF subtype 150 (OpenEdition – UNIX System Services – socket endpoint close), 151 (IUCV Endpoint close), and 152 (Assembler API Endpoint close)
- [Interval Records](#) – Tables describing Driver Statistics (Subtype 100) and Virtual Storage Statistics (Subtype 80)
- [SMF Report Writer Program](#) – Example of JCL to generate reports for the various SMF records written by Unicenter TCPaccess

Unicenter TCPaccess generates SMF records in accordance with the record formats provided in IBM publication *GC28-1030*. The Unicenter TCPaccess SMF record is variable length but begins with the MVS SMF record header and the Unicenter TCPaccess SMF record header. The Unicenter TCPaccess SMF record header is the anchor block used to determine which other sections are present.

On the Unicenter TCPaccess software tape, there is a sample of the JCL to dump Unicenter TCPaccess SMF records from dump data sets. In the MAC library, the SMFRECS macro maps the SMF records.

The SMF statement in member *IJTFCGxx*, described in the *Customization Guide*, specifies the Unicenter TCPaccess SMF recording.

## Standard MVS SMF Record Header

The following table is a sample of the standard MVS record header.

Offset	Field	Length	Format	Source	Description	
0	0	SMFACLEN	2	Binary	Internal	Record length
2	2	SMFACSEG	2	Binary	Internal	Segment descriptor
4	4	SMFACFLG	1	Binary	SVC 83	Header flag byte
5	5	SMFACRTY	1	Binary	Internal	Record type
6	6	SMFACTME	4	Binary	SVC 83	Time, in hundredths of a second, that the record was moved to the SMF buffer
10	A	SMFACDTE	4	Packed	SVC 83	Date record was moved to SMF buffer, in the form 00YYDDF, where F is the sign byte
14	E	SMFACSID	4	EBCDIC	SMCASID	System ID (taken from the SID parameter)

## Standard Unicenter TCPaccess SMF Record Header

This section provides samples of standard Unicenter TCPaccess SMF record headers. The record headers include Descriptor and Product sections.

### Descriptor

The following table contains the descriptor section of the SMF header.

Offset	Field	Length	Format	Source	Description	
18	12	SMFACNOD	2	Binary	Internal	Number of descriptors in this section
20	14	SMFACPOF	4	Binary	Internal	Offset to product section
24	18	SMFACPLN	2	Binary	Internal	Length of product section
26	1A	SMFACPNO	2	Binary	Internal	Number of product sections
28	1C	SMFACTOF	4	Binary	Internal	Offset to task information section
32	20	SMFACTLN	2	Binary	Internal	Length of task information section
34	22	SMFACTNO	2	Binary	Internal	Number of task information sections
36	24	SMFACUOF	4	Binary	Internal	Offset to user identification section

Offset	Field	Length	Format	Source	Description
40 28	SMFACULN	2	Binary	Internal	Length of user identification section
42 2A	SMFACUNO	2	Binary	Internal	Number of user identification sections
44 2C	SMFACDOF	4	Binary	Internal	Offset to data section
48 30	SMFACDLN	2	Binary	Internal	Length of data section
50 32	SMFACDNO	2	Binary	Internal	Number of data sections

## Product

The following table contains the product section of the SMF header.

Offset	Field	Length	Format	Source	Description
0 0	SMFACTYP	2	Binary	Internal	Subtype identification
2 2	SMFACRVN	2	EBCDIC	Internal	Record version number - 02
4 4	SMFACNAM	8	EBCDIC	Internal	Product name
12 C	SMFACVER	2	EBCDIC	Internal	Unicenter TCPAccess version number
14 E	SMFACACB	8	EBCDIC	APPCFGxx	Unicenter TCPAccess ACB name
22 16	SMFACJBN	8	EBCDIC	Internal	Jobname of Unicenter TCPAccess job
30 1E	SMFACJBI	8	EBCDIC	Internal	JES job ID of Unicenter TCPAccess job
38 26	SMFACASI	2	Binary	Internal	Address space ID of Unicenter TCPAccess job
40 28	SMFACSSI	4	EBCDIC	Internal	Subsystem ID of Unicenter TCPAccess job

## Server Application Event Records

### Task Identification Section

The following table contains the task identification section (pseudo-task data) of the Unicenter TCPaccess SMF header.

Offset	Field	Length	Format	Source	Description
0 0	SMFACPGM	8	EBCDIC	PTAPGM	Program name

### User Identification Section

The following table contains the user identification section (AC#U data) of the Unicenter TCPaccess SMF header:

Offset	Field	Length	Format	Source	Description
0 0	SMFACUID	8	EBCDIC	SECUID	RACF (or other external security system) user ID
8 8	SMFACGID	8	EBCDIC	SECGRP	RACF (or other external security system) group ID
16 10	SMFACACT	144		AC#UACCT	Accounting information

The following is the format of the SMFACACT field:

- First byte of field contains the number (binary) of accounting fields
- Following bytes contain accounting fields, where each entry for an accounting field contains a one-byte length field, followed by the field

## FTP Server Records (Subtypes 20/22)

### Descriptor

The following table shows the descriptor section of the FTP server record header.

Offset	Field	Length	Format	Source	Description
0	0	2	Binary	Internal	Reserved
2	2 FTPDNOD	2	Binary	Internal	Number of descriptors in this section
4	4 FTPDCSOF	4	Binary	Internal	Offset to FTP server common data section
8	8 FTPDCSLN	2	Binary	Internal	Length of FTP server common data section
10	A FTPDCSNO	2	Binary	Internal	Number of FTP server common data sections
12	C FTPDDTOF	4	Binary	Internal	Offset to FTP server data transfer completion data section
16	10 FTPDDLTLN	2	Binary	Internal	Length of FTP server data transfer completion data sections
18	12 FTPDDTNO	2	Binary	Internal	Number of FTP server data transfer completion data sections

## Common Data

The following table shows the common data section of the FTP server record header.

Offset	Field	Length	Format	Source	Description	
0	0	FTPCRHST	4	Binary	CACCRHST	Remote host Internet address for control connection
4	4	FTPCLHST	4	Binary	CACCLHST	Local host Internet address for control connection
8	8	FTPCRPT	2	Binary	CACCRSCK	Remote host TCP port number for control connection
10	A	FTPCLPRT	2	Binary	CACCLSCK	Local host TCP port number for control connection

### FTP Data Transfer Completion/End of Volume (Subtype 20/Subtype 22) Data Section

The Subtype 20 record is written by the FTP server at the end of a data transfer.

The Subtype 22 record is written at end-of-volume during a data transfer of a multi-volume data set. Its layout is identical to that of the Subtype 20 record.

The following table provides a sample of the Subtype 20/Subtype 22 data section:

Offsets	Field	Length	Format	Source	Description	
0	0	F20DTTY	1	Binary	CAFLGA1	Type of data transfer operation: 1 = NLST (data set name list without details) 2 = LIST (data set name list with details) 8 = RETR (retrieve) 16 = APPE (append) 32 = STOR (store)
1	1		1	Binary	Internal	Reserved
2	2	F20TYPE	1	Binary	CATYPE	Data type (from FTP TYPE command): 0 = ASCII 4 = EBCDIC 8 = Binary 16 = local byte
3	3	F20FORM	1	Binary	CAFORM	Format control (from FTP TYPE A or E command): 0 = None 4 = Telnet 8 = ASA CC
4	4	F20STRUA	1	Binary	CASTRU	Data structure (from FTP STRU command): 0 = File 4 = Record
5	5	F20MODE	1	Binary	CAMODE	Transmission mode (from FTP MODE command): 0 = Stream 4 = Block 8 = Compressed

Offsets	Field	Length	Format	Source	Description	
6	6	F20DSN	44	EBCDIC	CAPADSN	Dataset name: RETR APPE STOR Or path name: NLST LIST
50	32	F20MEM	8	EBCDIC	CAPAMEM	Member name
58	3A	F20DSSN	6	EBCDIC	CAPADSSN	Data set serial number (the first or only volser for the data set)
58	3A	F20VOL	6	EBCDIC	CAPADSSN	Same as F20DSSN. (This field kept for compatibility)
64	40	F20DDNM	8	EBCDIC	CADDNAME	DD name
72	48	F20RHST	4	Binary	CADTRHST	Remote host Internet address for data connection
76	4C	F20LHSTA	4	Binary	CADTLHST	Local host Internet address for data connection
80	50	F20RPRT	2	Binary	CADTRSCK	Remote host TCP port number for data connection
82	52	F20LPRT	2	Binary	CADTLSCK	Local host TCP port number for data connection
84	54	F20STRT	4	Binary	CATIME	Time, in hundredths of a second, that data transfer started
88	58	F20DURA	4	Binary	CATIME+4	Elapsed time, in hundredths of a second, for data transfer operation
92	5C	F20MSG	2	Binary	CAREPFTP	Last FTP reply number
94	5E	F20SBMSG	2	Binary	CAREPSUB	Sub-reply number for last FTP reply
96	60	F20DAIR	4	Binary	CADAIR	PDYNAL reason code: SVC 99 reason code if RETR, APPE, or STOR SUPERLOCATE reason code if NLST or LIST
100	64	F20DARC	1	Binary	CADARC	PDYNAL return code: SVC 99 return code if RETR, APPE, or STOR SUPERLOCATE return code if NLST or

Offsets	Field	Length	Format	Source	Description	
LIST						
101	65	3	Binary	Internal	Reserved	
104	68	F20DSKB	4	Binary	CABYTES	Number of disk bytes read or written
108	6C	F20NETB	4	Binary	CALBYTES	Number of network bytes sent or received
112	70	F20RECT	4	Binary	CATRUNC	Number of records truncated
116	74	F20RECP	4	Binary	CAPADD	Number of records padded
120	78	F20RECF	4	Binary	CAFOLD	Number of records folded
124	7C	F20RECS	4	Binary	CASUSP	Number of records suspected
128	80	F20RSTM	4	Binary	CARSTM	Number of restart markers sent or received
132	84	F20RSTI	4	Binary	CARSTIGN	Number of restart markers ignored
136	88	F20AERR	4	Binary	CA#TERR	API composite return code
140	8C	F20EXPD	7	EBCDIC	CAPAEXP	Expiration date
147	93	F20RETP	2	Binary	CAPARETP	Retention period
149	95	F20VOLI	6	EBCDIC	CACRVOL	Volume serial number. (For subtype 22 records, this is the volume at EOVS; for subtype 20 records, this is the last volume for the data set.)
155	9B	F20VLSQ	2	Binary	CACRVLSQ	Volume sequence number. (For subtype 22 records, this is the sequence number of the volume at EOVS; for subtype 20 records this is the number of volumes for the data set.)
155	9B	F20NVOL	2	Binary	CACRVLSQ	Same as F20VLSQ
157	9D	F20DACL	30	Character	CAPADACL	SMS Data class
187	BB	F20STCL	30	Character	CAPASTCL	SMS Storage class
217	D9	F20MGCL	30	Character	CAPAMGCL	SMS Management class

**Note:** The subtype 20 and subtype 22 records have identical formats.

The totals fields in the subtype 22 EOVS records (for example, F20DURA – elapsed time – and F20DSKB – disk bytes written) are calculated from the beginning of the data transfer (not the beginning of processing for the volume).

### FTP Data Set Modification (Subtype 21) Data Section

The subtype 21 record is written by the FTP server when a request to modify a data set (other than a data transfer) is completed. This includes the rename, delete, and make directory (MKD) commands.

The following table provides a sample of the Subtype 21 data section.

Offset	Field	Length	Format	Source	Description	
0	0	F21DMTY	1	Binary	CAFLGS1	Data set modification type 16 = MKD (make directory) 64 = DELE (delete) 128 = RENM/RNTO (rename)
1	1		1	Binary	Internal	Reserved
2	2	F21DDNM	8	EBCDIC	CADDNAME	DD name
10	A	F21FDSN	44	EBCDIC	CAPADSN	Rename from or delete data set name
54	36	F21FMEM	8	EBCDIC	CAPAMEM	Rename from or delete member name
62	3E	F21DSSN	6	EBCDIC	CAPADSSN	Data set serial number (the first or only volser for the data set)
62	3E	F21VOL	6	EBCDIC	CAPADSSN	Same as F21DSSN (this field kept for compatibility)
68	44	F21TDSN	44	EBCDIC	CAPADSN	Rename to data set name
112	70	F21TMEM	8	EBCDIC	CAPAMEM	Rename to member name
120	78	F21MSG	2	Binary	CAREPFTP	Last FTP reply number
122	7A	F21SBMSG	2	Binary	CAREPSUB	Sub-reply number for last FTP reply
124	7C	F21DAIR	4	Binary	CADAIR	PDYNAL reason code
128	80	F21DARC	1	Binary	CADARC	PDYNAL return code

Offset	Field	Length	Format	Source	Description
129	81 F21NVOL	2	Binary	CAPANVOL	Number of volumes
131	83 F21VOLS	8-2040	Structure		Beginning of the volume section (This section is not present for data sets residing on a single volume)

For each volume of a multi-volume data set, there is an eight-byte section with the format shown in the following table:

Offset	Field	Length	Format	Source	Description
0	0	2	Binary		Reserved
2	2 F21VOLI	6	EBCDIC	CAPAVOLS	Volume serial number

## Telnet Server Record (Subtype 23) Data Section

The SMF subtype 23 record is written by the telnet server upon termination of a session with a VTAM session partner. Several subtype 23 records may be written for a single telnet session. The subtype 23 record contains a User and a Task section. These are identical in format with the User and Task sections for the FTP records (20-22).

Offset	Length	Label	Format	Source	Description
00		S23DATA			Data section
00	04	S23LIPA	Binary	TCCLINET	Local host IP address
04	04	S23RIPA	Binary	TCCRINET	Remote host IP address
08	02	S23LPOR	Binary	TCCLINET	Local host port
0A	02	S23RPOR	Binary	TCCRINET	Remote host port
0C	08	S23SNET	EBCDIC		Netid associated with SLU (TN3270E server only)
14	08	S23SLU	EBCDIC	LURQLUN	ACB name (SLU)
1C	08	S23PNET	EBCDIC		Netid associated with PLU (TN3270E server only)
24	08	S23PLU	EBCDIC	VCBPART	PLU name

Offset	Length	Label	Format	Source	Description
	81				Length of Subtype 23 data section
2C	04	S23STIME	Binary	TIME macro	Session start time, in hundredths of a second since midnight
30	04	S23SDATE	Binary	TIME macro	Session start date, in form 0cyydddF, where: c = century indicator (0=19, 1=20) yy = decimal year within century ddd = decimal Julian day
34	04	S23TTIME	Binary	TIME macro	Session termination time
38	04	S23TDATE	Binary	TIME macro	Session termination date
3C	04	S23DURA	Binary		Session duration, in hundredths of a second
40	04	S23CLIN	Binary	internal	Number of bytes received from Telnet client
44	04	S23CLOU	Binary	internal	Number of bytes transmitted to Telnet client
48	04	S23VTIN	Binary	internal	Number of bytes received from VTAM application
4C	04	S23VTOU	Binary	internal	Number of bytes sent to VTAM application
50	01	S23TTYPL	Binary	TCTYPEL	Length of terminal type field
51	40	S23TTYPE	EBCDIC	TCCTYPE	Terminal type negotiated with Telnet client
79	08	S23LMODE	EBCDIC	VCBPART	Logmode
	81				Length of Subtype 23 data section
<b>The following fields are present for T01S3270 (TN3270E server records only)</b>					
81	4	S23SESS	Binary		TCP connection number
85	8	S23APPL	EBCDIC		Requested application name
8D	4	S23VBK1	Binary		VTAM Bucket 1 = time < 1 second
91	4	S23VBK2	Binary		VTAM Bucket 2 = time < 2 second
95	4	S23VBK3	Binary		VTAM Bucket 3 = time < 5 second
99	4	S23VBK4	Binary		VTAM Bucket 4 = time < 10 second
9D	4	S23VBK5	Binary		VTAM Bucket 5 = time >= 1 second
A1	4	S23TBK1	Binary		TCP Bucket 1 = time < 1 second
A5	4	S23TBK2	Binary		TCP Bucket 2 = time < 2 second

Offset	Length	Label	Format	Source	Description
A9	4	S23TBK3	Binary		TCP Bucket 3 = time < 5 second
AD	4	S23TBK4	Binary		TCP Bucket 4 = time < 10 second
B1	4	S23TBK5	Binary		TCP Bucket 5 = time >= 1 second

## Access Method and Protocol Layer Event Records

### Access Method User Identification Section

The SMF subtype 150 (UNIX System Services – socket endpoint close), 151 (IUCV Endpoint close), and 152 (Assembler API Endpoint close) are written when a Socket API endpoint is closed. They contain statistics on access method calls for the socket.

Offset	Length	Label	Format	Source	Description
00		SMFPUSER			User section
00	08	SMFPJOB	EBCDIC	ATCBJOB	Job Name of User
08	08	SMFPSTEP	EBCDIC	ATCBSTEP	Step Name of User
10	08	SMFPPROC	EBCDIC	ATCBPROC	Procedure Name of User
18	04	SMFPTCB	Binary	ATCBTCB	TCB Address of User
1C	04	SMFPSEPM	Binary	Internal	SEPM Address of User
1C	04	SMFPINOD	Binary	SEPMINOD	Address of OE/PFS INODE (redefines S10SEPM).
20	02	SMFPASID	Binary	ATCBASID	ASID of User
	22	SMFPULEN	Binary		Length of User Identification Section

### Access Method Endpoint Close Records (Subtypes 150 - 152)

This section describes data sections for the access method endpoint close records. These are the following:

- Subtype 150, UNIX System Services Endpoint Close
- Subtype 151, IUCV Endpoint Close
- Subtype 152, Assembler API Endpoint Close.

**UNIX System Services Endpoint Close (Subtype 150) Data Section**

Offset	Length	Label	Format	Source	Description
00	64	S150DATA	Binary		SMF subtype 150 - OpenEdition Endpoint Close
00	01	S150SOPR	Binary	SEPMMSOPR	Socket Protocol (mapped by SMFMSOPR): X'00' SMFMSPIP = IP (RAW default) X'01' SMFMSPICM = ICMP (RAW) X'06' SMFMSPTCP = TCP X'11' SMFMSPPUDP = UDP X'59' SMFMSPOSP = OSPF (RAW) X'FF' = SMFMSPPRAW (RAW)
01	01	S150TYPE	Binary	SEPMSTYPE	Access method type (mapped by SMFSTYPE): X'00' SMFSTYOE = OpenEdition X'01' SMFSTYIU = IUCV X'02' SMFSTYAA = Assembler API
02	02				Reserved
04	04	S150CONO	Binary	SEPMUNIQ	Connection Number
08	04	S150LIPA	Binary	SEPMLHST	Local IP Address
0C	04	S150RIPA	Binary	SEPMRHST	Remote IP Address (TCP and UDP only)
10	02	S150LPOR	Binary	SEPMLPOR	Local Port Number (TCP and UDP only)
12	02	S150RPOR	Binary	SEPMRPOR	Remote Port Number (TCP and UDP only)
14	04	S150COSO	Binary	SEPMCOSO	Number of OpenEdition Socket Calls (1)
18	04	S150COAC	Binary	SEPMCOAC	Number of OpenEdition Accept Calls (2)
1C	04	S150COBD	Binary	SEPMCOBD	Number of OpenEdition Bind Calls (3)
20	04	S150COCL	Binary	SEPMCOCL	Number of OpenEdition Close Calls (4)
24	04	S150COCO	Binary	SEPMCOCO	Number of OpenEdition Connect Calls (5)
28	04	S150COGN	Binary	SEPMCOGN	Number of OpenEdition GetPeerName Calls (6)
2C	04	S150COGO	Binary	SEPMCOGO	Number of OpenEdition GetSockOpt Calls (7)
30	04	S150COSS	Binary	SEPMCOSS	Number of OpenEdition SetSockOpt Calls (8)
34	04	S150COIO	Binary	SEPMCOIO	Number of OpenEdition IOctl Calls (9)
38	04	S150COLI	Binary	SEPMCOLI	Number of OpenEdition Listen Calls (10)
3C	04	S150CORD	Binary	SEPMCORD	Number of OpenEdition Read Calls (11)

Offset	Length	Label	Format	Source	Description
40	04	S150COWR	Binary	SEPMCOWR	Number of OpenEdition Write Calls (12)
44	04	S150COVR	Binary	SEPMCOVR	Number of OpenEdition ReadV Calls (13)
48	04	S150COVW	Binary	SEPMCOVW	Number of OpenEdition WriteV Calls (14)
4C	04	S150CORE	Binary	SEPMCORE	Number of OpenEdition Recv Calls (15)
50	04	S150COSE	Binary	SEPMCOSE	Number of OpenEdition Send Calls (16)
54	04	S150COFR	Binary	SEPMCOFR	Number of OpenEdition RecvFrom Calls (17)
58	04	S150COST	Binary	SEPMCOST	Number of OpenEdition SendTo Calls (18)
5C	04	S150CORM	Binary	SEPMCORM	Number of OpenEdition RecvMsg Calls (19)
60	04	S150COSM	Binary	SEPMCOSM	Number of OpenEdition SendMsg Calls (20)
64	04	S150COSL	Binary	SEPMCOSL	Number of OpenEdition Select Calls (21)
68	04	S150COSN	Binary	SEPMCOSN	Number of OpenEdition SetPeerName Calls (22)
6C	04	S150COSH	Binary	SEPMCOSH	Number of OpenEdition Shutdown Calls (23)
70	04	S150COGH	Binary	SEPMCOGH	Number of OpenEdition GetHost Calls (24)
74	04	S150COSB	Binary	SEPMCOSB	Number of OpenEdition BatchSelect Calls (25)
	78	S150DLEN	Binary		Length of Subtype 150 record

### IUCV Endpoint Close (Subtype 151) Data Section

Offset	Length	Label	Format	Source	Description
00	A8	S151DATA	Binary		SMF subtype 151 - IUCV Endpoint Close
00	01	S151SOPR	Binary	SEPMSOPR	Socket Protocol (mapped by SMFPSOPR): X'00' SMFPPIP = IP (RAW default) X'01' SMFPPICM = ICMP (RAW) X'06' SMFPPTCP = TCP X'11' SMFPPUDP = UDP X'59' SMFPPOSP = OSPF (RAW) X'FF' = SMFPPRAW (RAW)
01	01	S151TYPE	Binary	SEPMTYPE	Access method type (mapped by SMFPTYPE): X'00' SMFPTYOE = OpenEdition X'01' SMFPTYIU = IUCV X'02' SMFPTYAA = Assembler API

Offset	Length	Label	Format	Source	Description
02	02				Reserved
04	04	S151CONO	Binary	SEPMUNI Q	Connection Number
08	04	S151LIPA	Binary	SEPMLHST	Local IP Address
0C	04	S151RIPA	Binary	SEPMRHS T	Remote IP Address (TCP and UDP only)
10	02	S151LPOR	Binary	SEPMLPO R	Local Port Number (TCP and UDP only)
12	02	S151RPOR	Binary	SEPMRPO R	Remote Port Number (TCP and UDP only)
14	04	S151CIAC	Binary	SEPMCICAC	Number of IUCV Accept Calls (1)
18	04	S151CIBD	Binary	SEPMCIBD	Number of IUCV Bind Calls (2)
1C	04	S151CICL	Binary	SEPMCICL	Number of IUCV Close Calls (3)
20	04	S151CICO	Binary	SEPMCICO	Number of IUCV Connect Calls (4)
24	04	S151CIFC	Binary	SEPMCIFC	Number of IUCV Fcntl Calls (5)
28	04				Unused for IUCV (6)
2C	04	S151CIGH	Binary	SEPMCIG H	Number of IUCV GetHostID Calls (7)
30	04	S151CIGJ	Binary	SEPMCIGJ	Number of IUCV GetHostName Calls(8)
34	04	S151CIGN	Binary	SEPMCIG N	Number of IUCV GetPeerName Calls (9)
38	04	S151CIGM	Binary	SEPMCIG M	Number of IUCV GetSockName Calls (10)
3C	04	S151CIGO	Binary	SEPMCIGO	Number of IUCV GetSockOpt Calls (11)
40	04	S151CIIO	Binary	SEPMCIO	Number of IUCV IOctl Calls (12)
44	04	S151CILI	Binary	SEPMCILI	Number of IUCV Listen Calls (13)
48	04	S151CIRD	Binary	SEPMCIRD	Number of IUCV Read Calls (14)
4C	04				Unused for IUCV (15)
50	04	S151CIRE	Binary	SEPMCIRE	Number of IUCV Recv Calls (16)
54	04				Unused for IUCV (17)
58	04				Unused for IUCV (18)
5C	04	S151CISL	Binary	SEPMCISL	Number of IUCV Select Calls (19)

Offset	Length	Label	Format	Source	Description
60	04	S151CISE	Binary	SEPMCISE	Number of IUCV Send Calls (20)
64	04				Unused for IUCV (21)
68	04	S151CIST	Binary	SEPMCIST	Number of IUCV SendTo Calls (22)
6C	04	S151CISS	Binary	SEPMCISS	Number of IUCV SetSockOpt Calls (23)
70	04	S151CISH	Binary	SEPMCISH	Number of IUCV Shutdown Calls (24)
74	04	S151CISO	Binary	SEPMCISO	Number of IUCV Socket Calls (25)
78	04	S151CIWR	Binary	SEPMCIW R	Number of IUCV Write Calls (26)
7C	04				Unused for IUCV (27)
80	04				Unused for IUCV (28)
84	04	S151CIER	Binary	SEPMCIER	Number of IUCV LastErrno Calls (29)
88	04	S151CIID	Binary	SEPMCIID	Number of IUCV GetClientID Calls (30)
8C	04	S151CIGV	Binary	SEPMCIQV	Number of IUCV GiveSocket Calls (31)
90	04	S151CITS	Binary	SEPMCITS	Number of IUCV TakeSocket Calls (32)
94	04				Unused for IUCV (33)
98	04				Unused for IUCV (34)
9C	04	S151CIGI	Binary	SEPMCIQI	Number of IUCV GetIBMSockOpt Calls (35)
A0	04	S151CISI	Binary	SEPMCISI	Number of IUCV SetIBMSockOpt Calls (36)
A4	04				Unused for IUCV (37)
A8	04				Unused for IUCV (38)
AC	04				Unused for IUCV (39)
B0	04				Unused for IUCV (40)
B4	04				Unused for IUCV (41)
B8	04	S151CICA	Binary	SEPMCICA	Number of IUCV Cancel Calls (42)
	BC	S151DLEN	Binary		Length of Subtype 151 record

**Assembler API Endpoint Close (Subtype 152) Data Section**

Off set	Length	Label	Format	Source	Description
00		S152DATA			SMF subtype 152 - API Endpoint Close
00	01	S152SOPR	Binary	SEPMSOPR	Socket Protocol (mapped by SMFPSOPR): X'00' SMFPPIP = IP (RAW default) X'01' SMFPPICM = ICMP (RAW) X'06' SMFPPTCP = TCP X'11' SMFPPUDP = UDP X'59' SMFPPOSP = OSPF (RAW) X'FF' = SMFPPRAW (RAW)
01	01	S152TYPE	Binary	SEPMTYPE	Access method type (mapped by SMFPTYPE): X'00' SMFPTYOE = OpenEdition X'01' SMFPTYIU = IUCV X'02' SMFPTYAA = Assembler API
02	02				Reserved
04	04	S152CONO	Binary	SEPMUNIQ	Connection Number
08	04	S152LIPA	Binary	SEPMLHST	Local IP Address
0C	04	S152RIPA	Binary	SEPMRHST	Remote IP Address (TCP and UDP only)
10	02	S152LPOR	Binary	SEPMLPOR	Local Port Number (TCP and UDP only)
12	02	S152RPOR	Binary	SEPMRPOR	Remote Port Number (TCP and UDP only)
14	04	S152CAAC	Binary	SEPMCAAC	Number of API Taccept calls
18	04	S152CAAD	Binary	SEPMCAAD	Number of API Taddr calls
1C	04	S152CABD	Binary	SA52CABD	Number of API Tbind calls
20	04	S152CATC	Binary	SEPMCATC	Number of API Tclear calls
24	04	S152CACL	Binary	SEPMCACL	Number of API Tclose calls
28	04	S152CACF	Binary	SEPMCACF	Number of API Tconfirm calls
2C	04	S152CACO	Binary	SEPMCACO	Number of API Tconnect calls
30	04	S152CADI	Binary	SEPMCADI	Number of API Tdisconnect calls
34	04	S152CAIN	Binary	SEPMCAIN	Number of API Tinfo calls
38	04	S152CALI	Binary	S152CALI	Number of API Tlisten calls
3C	04	S152CAOP	Binary	SEPMCAOP	Number of API Topen calls
40	04	S152CAOT	Binary	SEPMCAOT	Number of API Toption calls

Off set	Length	Label	Format	Source	Description
44	04	S152CARE	Binary	SEPMCARE	Number of API Trecv calls
48	04	S152CARR	Binary	SEPMCARR	Number of API Trecvrr calls
4C	04	C152CARF	Binary	SEPMCARF	Number of API Trecvfr calls
50	04	S152CARJ	Binary	SEPMCARJ	Number of API Treject calls
54	04	S152CARK	Binary	SEPMCARK	Number of API Trelack calls
58	04	S152CARL	Binary	SEPMCARL	Number of API Trelease calls
5C	04	S152CART	Binary	SEPMCART	Number of API Tretract calls
60	04	S152CASE	Binary	SEPMCASE	Number of API Tsend calls
64	04	S152CAST	Binary	SEPMCAST	Number of API Tsendto calls
68	04	S152CAUN	Binary	SEPMCAUN	Number of API Tunbind calls
6C	04	S152CAUS	Binary	SEPMCAUS	Number of API Tuser calls
70	14				Reserved
84	04	S152CACK	Binary	SEPMCACK	Number of API Tcheck calls
88	04	S152CAER	Binary	SEPMCAER	Number of API Terror calls
8C	04	S152CATE	Binary	SEPMCATE	Number of API Tstate calls
	90	S152DLEN	Binary		

## Protocol Layer Events (Subtypes 110 - 123)

The SMF subtype 110-123 records are written to record certain protocol-related events, such as a bind or connect request. The format for each is identical, although the lengths vary. It contains a user and a data section.

The following table describes the relationship of these subtypes.

Subtype	Description
110	Endpoint create
111	TCP connect request
112	TCP accept request
113	TCP connection close
114	TCP bind
115	TCP unbind

Subtype	Description
116	TCP listen
117	UDP bind
118	UDP connect
119	UDP close
120	RAW open
121	RAW close
122	Endpoint destroy
123	Inbound connection failures

### Protocol Layer User Identification Section

The User Identification section identifies the application address space and task on whose behalf the event was completed (for SMF record subtype 110 through 123).

Offset	Length	Label	Format	Source	Description
00		SMFPUSER			User section
00	08	SMFPJOB	Binary	ATCBJOB	Job Name of User
08	08	SMFPSTEP	Binary	ATCBSTEP	Step Name of User
10	08	SMFPPROC	Binary	ATCBPROC	Proc Name of User
18	04	SMFPTCB	Binary	ATCBTCB	TCB Address of User
1C	04	SMFPSEPM	Binary	Internal	SEPM Address of User
1C	04	SMFPINOD	Binary	SEPMINOD	Address of OE/PFS INODE (redefines S10SEPM).
20	02	SMFPASID	Binary	ATCBASID	ASID of User
	22	SMFPULEN	Binary		Length of User Identification Section

## Protocol Layer Data Section

The Data section identifies the connection and provides connection statistics for SMF record subtype 110 through Subtype 123.

Offset	Length	Label	Format	Source	Description
00		SMFPDATA			Data section - PROTOCOL record
00	01	SMFPSOPR	Binary	SEPMSOPR	Socket Protocol: X'00' SMFPPIP = IP (RAW default) X'01' SMFPPICM = ICMP (RAW) X'06' SMFPPTCP = TCP X'11' SMFPPUDP = UDP X'59' SMFPPOSP = OSPF (RAW) X'FF' = SMFPPRAW (RAW)
01	01	SMFPTYPE	Binary	SEPMTYPE	Access method type: X'00' SMFPTYOE = OpenEdition X'01' SMFPTYIU = IUCV X'02' SMFPTYAA = Assembler API
02	02				Reserved
04	04	SMFPCONO	Binary	SEPMUNIQ	Connection Number
	08	S120DLEN	Binary		Length of data section for Subtype 120
08	04	SMFPLIPA	Binary	SEPMLHST	Local IP Address
0C	04	SMFPRIPA	Binary	SEPMRHST	Remote IP Address (TCP and UDP only)
10	02	SMFPLPOR	Binary	SEPMLPOR	Local Port Number (TCP and UDP only)
	12	S114DLEN	Binary		Length of data section for Subtype 114
	12	S115DLEN	Binary		Length of data section for Subtype 115
	12	S116DLEN	Binary		Length of data section for Subtype 116
	12	S117DLEN	Binary		Length of data section for Subtype 117
12	02	SMFPRPOR	Binary	SEPMRPOR	Remote Port Number (TCP and UDP only)
	14	S110DLEN	Binary		Length of data section for Subtype 110
	14	S111DLEN	Binary		Length of data section for Subtype 111
	14	S112DLEN	Binary		Length of data section for Subtype 112
	14	S118DLEN	Binary		Length of data section for Subtype 118
	14	S123DLEN	Binary		Length of data section for Subtype 123

Offset	Length	Label	Format	Source	Description
	20	S124DLEN	Binary		Length of data section for Subtype 124
14	04	SMFPPKTI	Binary	SEPMRECD	Packets In
18	04	SMFPPKTO	Binary	SEPMSENT	Packets Out
1C	04	SMFPBYTI	Binary	SEPMDATI	Bytes In
20	04	SMFPBYTO	Binary	SEPMDATO	Bytes Out
24	04	SMFPCSER	Binary	SEPMCSER	Checksum Errors
28	04	SMFPFMER	Binary	SEPMFORM	Format Errors
2C	04	SMFPPKTD	Binary	SEMPDRO	Packets Dropped
30	04	SMFPBYTD	Binary	SEPMDROP	Bytes Dropped
	30	S119DLEN	Binary		Length of data section for Subtype 119
	30	S121DLEN	Binary		Length of data section for Subtype 121
34	04	SMFPREXO	Binary	SEPMRECT	Retransmits Sent
38	04	SMFPACKI	Binary	SEPMACKI	Acks In
42	04	SMFPACKO	Binary	SEPMACKO	Acks Out
3C	02	SMFPMSS	Binary	SEPMTCMS	MSS In Effect (Outbound)
	40	S113DLEN	Binary		Length of data section for Subtype 113
	3E	S122DLEN	Binary		Length of data section for Subtype 122
	3E	SMFPDLEN	Binary		Maximum length of PROTOCOL record

## Interval Records

The periodic, interval-driven records are written every so often, based on the INTERVAL parameter of the SMF statement in member IJTFCGxx. The interval-driven records include subtypes 80 and 100. The INTERVAL parameter is described in the *Customization Guide*.

### Driver Statistics (Subtype 100) Data Section

The SMF record subtype 100 (DRIVER) is written periodically (governed by the INTERVAL parameter of the SMF statement in the IJTFCGxx configuration member) to record statistics for physical devices. It contains no Task or User Identification sections.

Offset	Length	Label	Format	Source	Description
00		S100DATA			Data portion of subtype 100 (DRIVER) record.
00	08	S100NAME	EBCDIC	LNICNAME	Interface name
08	02	S100TYPE	Binary	SNLNTYPE	Interface type 6 S100ETHR - Ethernet 9 S100TOKN - Token Ring 14 S100HYPR - Hyperchannel 15 S100FDDI - FDDI Ring 22 S100P2P - Point to Point (Claw) 24 S100LOOP - Loopback
0A	01	S100OSTA	Binary	SNLNOSTA	Operational status x'00' S100ODWN - Interface Down x'01' S100OUP - Interface Up x'02' S100OPOL - Interface is Polling x'03' S100ODED - Interface Appears Dead
0B	01	S100ASTA	Binary	SNLNASTA	Administrative status x'00' S100ADWN - Administratively down x'01' S100AUP - Administratively up
0C	04	S100SPED	Binary	SNLNSPED	Media Speed
10	04	S100NMTU	Binary	SNLNNMTU	Maximum Transmission Unit Size for network
14	02	S100HWLN	Binary	SNLNHWLN	Hardware Address Length
16	06	S100HWAD	Binary	SNLNHWAD	Hardware Address

Offset	Length	Label	Format	Source	Description
1C	08	S100TIME	Binary	SNLNTIME	Time of Last status change
24	04	S100INBT	Binary	SNLNINBT	Count of inbound bytes
28	04	S100INUC	Binary	SNLNINUC	Count of inbound unicast packets
2C	04	S100INNU	Binary	SNLNINNU	Count of inbound non-unicast packets
30	04	S100INDS	Binary	SNLNINDS	Count of inbound packets discarded
34	04	S100INER	Binary	SNLNINER	Count of inbound error packets
38	04	S100INUN	Binary	SNLNINUN	Count of unknown protocol packets
3C	04	S100OUBY	Binary	SNLNOUBY	Count of outbound bytes
40	04	S100OUUC	Binary	SNLNOUUC	Count of outbound unicast packets
44	04	S100OUNU	Binary	SNLNOUNU	Count of outbound non-unicast packets
48	04	S100OUDS	Binary	SNLNOUDS	Count of outbound packets discarded
4C	04	S100OUER	Binary	SNLNOUER	Count of outbound error packets
50	04	S100OUPQ	Binary	SNLNOUPQ	Count of outbound packets queued
	54	S100DLEN	Binary		Length of subtype 100 record data portion

### Virtual Storage Statistics (Subtype 80) Data Section

The SMF record subtype 80 (VMSTATS) is written periodically (governed by the INTERVAL parameter of the SMF statement in the IJTCFGxx configuration member) to record virtual storage manager statistics for subpools. It contains no Task or User Identification sections.

Offset	Length	Label	Format	Source	Description
00	04	S80SPOFF	Binary	Internal	Offset to subpool descriptor section
04	04	S80SPNUM	Binary	Internal	Number of subpool descriptor entries (256)
08	04	S80IPOFF	Binary	Internal	Offset to IFS pool descriptor section
0C	04	S80IPNUM	Binary	Internal	Number of IFS pool descriptor entries
10	04	S80PVT	Binary	VSMREGN	Address of start of private area region (<16m)
14	04	S80PVTSZ	Binary	VSMREGN	Size of private area region (<16m)
18	04	S80EPVT	Binary	VSMREGN	Address of start of extended private area region
1C	04	S80EPVTS	Binary	VSMREGN	Size of extended private area region
20	04	S80ALLO	Binary	VSMLIST	Total allocated in all private area subpools (<16m)

Offset	Length	Label	Format	Source	Description
24	04	S80FREE	Binary	VSMLIST	Total free in all private area subpools (<16m)
28	04	S80USED	Binary	VSMLIST	Total used in all private area subpools (<16m)
2C	04	S80EALLO	Binary	VSMLIST	Total allocated in all private area subpools in extended region
30	04	S80EFREE	Binary	VSMLIST	Total free in all private area subpools in extended region
34	04	S80EUSED	Binary	VSMLIST	Total used in all private area subpools in extended region
38	04	S80TALLO	Binary	VSMLIST	Total allocated in all private area subpools
3C	04	S80TFREE	Binary	VSMLIST	Total free in all private area subpools
40	04	S80TUSED	Binary	VSMLIST	Total used in all private area subpools
44	04	S80RGSZ	Binary	VSMLIST	Total region size (<16m)
48	04	S80RGUN	Binary	VSMLIST	Total unallocated space (<16m)
4C	04	S80RGUS	Binary	VSMLIST	Total region used (<16m)
50	04	S80ERGSZ	Binary	VSMLIST	Total region size (>16m)
54	04	S80ERGUN	Binary	VSMLIST	Total unallocated space (>16m)
58	04	S80ERGUS	Binary	VSMLIST	Total region used (>16m)
5C	04	S80TRGSZ	Binary	VSMLIST	Total region size
60	04	S80TRGUN	Binary	VSMLIST	Total unallocated space
64	04	S80TRGUS	Binary	VSMLIST	Total region used
68	2868	S80SPLS	Binary	VSMLIST	Subpool descriptor area (described below)
2860	Varies	S80IFSPL	Binary	IPHD	IFS Pool descriptor section
	2800	S80LEN	Binary		Minimum length of subtype 80 record

The following table is a detailed description of the S80SPLS area described in the previous table, [Virtual Storage Statistics \(Subtype 80\) Data Section](#). This area is repeated 256 times (once for each subpool):

### Detailed Description of S80SPLS Area

Offset	Length	Label	Format	Source	Description
00	02	S80SPID	EBCDIC	VSMLIST	Subpool id
			C		
02	02				Reserved
04	04	S80SPALO	Binary	VSMLIST	Total allocated in subpool (<16m)
08	04	S80SPFRE	Binary	VSMLIST	Total free in subpool (<16m)
0C	04	S80SPUSE	Binary	VSMLIST	Total used in subpool (<16m)
10	04	S80SPEAL	Binary	VSMLIST	Total allocated in subpool in extended region
14	04	S80SPEFR	Binary	VSMLIST	Total free in subpool in extended region
18	04	S80SPEUS	Binary	VSMLIST	Total used in subpool in extended region
1C	04	S80SPTAL	Binary	VSMLIST	Total allocated in subpool
20	04	S80SPTFR	Binary	VSMLIST	Total free in subpool
24	04	S80SPTUS	Binary	VSMLIST	Total used in subpool
	28	S80SPLN			Length of subpool descriptor section

The following table is a detailed description of the IFS Pool descriptor section (S80IFSPL) described in the [Virtual Storage Statistics \(Subtype 80\) Data Section](#).

### Detailed Description of S80IFSPL Area

Offset	Length	Label	Format	Source	Description
00	04	S80IPID	EBCDIC	IPHD	IFS pool ID
04	01	S80IPSP	Binary	IPHD	Subpool
05	03	S80IPLN	Binary	IPHD	Length of elements
08	01	S80IPFLG	Binary	IPHD	Flags: X'80' S80IPFSP Requests may be suspended X'40' S80IPFFX Permanently fixed storage X'10' S80IPFHD Headers separate X'08' S80IPFEX Format exit exists
09	03		Binary		Reserved
0C	04	S80IPTTL	Binary	IPHD	Total elements in pool
10	04	S80IPFRE	Binary	IPHD	Count of free entries

Offset	Length	Label	Format	Source	Description
14	04	S80IPFLW	Binary	IPHD	Low-water mark of free entries
18	04	S80IPHWM	Binary	IPHD	Highest count of entries
1C	04	S80IPREQ	Binary	IPHD	Number of requests from pool
20	04	S80IPEXP	Binary	IPHD	Times expanded
24	04	S80IPCNT	Binary	IPHD	Times contracted
28	04	S80IPWCT	Binary	IPHD	Number of callers who waited
2C	04	S80IPERR	Binary	IPHD	Number of callers in error
	30	S80IPLN	Binary		Length of IFS pool descriptor section

## SMF Report Writer Program

The SMF Report Writer Program, T00SMFRW, generates several types of reports for the various SMF records written by Unicenter TCPaccess.

To execute the Report Writer, use the following JCL:

```
//SMFRW    EXEC  PGM=T00SMFRW
//STEPLIB DD   DISP=SHR,DSN=TRGIND X.LINK
//SMFIN    DD   DISP=SHR,DSN=SMFDATA
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD   *
/*
```

Where:

- SMFIN DD Defines the file holding the SMF records. This file can be created from the system SMF data sets using the IBM IFASMFDP utility.
- SYSPRINT DD Defines the file where the report will be written.
- SYSIN DD Defines the file that holds the Report Writer commands.

## Commands

The following commands are recognized by the report writer:

RECORDTYPE=*record\_type*

Defines the SMF record type used by Unicenter TCPaccess.

Valid types are 128-255.

Default: 130.

DATE=[ *date* | ( *start\_date-end\_date* ) ]

Defines the date or range of dates for which SMF records will be selected for the report.

Format: *yyyyddd*, where

*yyyy* = year

*ddd* = Julian day of year

Default: All dates.

TIME=[ *time* | ( *start\_time-end\_time* ) ]

Defines the time or range of times for which SMF records will be selected for the report.

Format:*hh:mm:ss.th*, where:

*hh* = the hour

*mm* = minutes

*ss* = seconds

*th* = tenths and hundredths of a second.

Any precision not specified is assumed to be zero.

For instance, 12:10 = 12:10:00.00.

Default: All times

**REPORT=report\_type**

Defines the report type to be written. Several REPORT commands can be included to combine reports. The report types are described in the following table.

**LOCAL=ip\_address:port**

Limits the report to records matching the local IP address and port specified.

**REMOTE=ip\_address:port**

Limits the report to records matching the remote IP address and port specified.

**JOBNAME=jobname**

Limits the report by the jobname of the Unicenter TCPaccess job.

The report types in the following table can be specified using the REPORT= command.

<b>Report Type</b>	<b>Description</b>	<b>Record Subtypes</b>
VSMSTATS	VSM statistics (totals)	80
VSMPOOLS	VSM Statistics by subpool	80
IFSPOOLS	IFS pool activity	80
DRIVER	Device driver statistics	100
PROTOCOL	Protocol-layer activity	110-124
TCP	TCP protocol activity	110-115,122-124
UDP	UDP protocol activity	110,117-119,111
RAW	RAW protocol activity	110,120-122
OE	OE Endpoint close	150
IUCV	IUCV Endpoint close	151
API	API Endpoint close	152
FTP	FTP activity	20-22
TELNET	Telnet activity	23
DUMP	Unformatted dump of records	All
ALL	All reports (except DUMP)	All

**Note:**

- The DUMP report should not be combined with another report type
- The LOCAL and REMOTE commands apply to the protocol reports, the endpoint close reports, the FTP, and the Telnet reports

For FTP, the IP addresses and ports are applied to the control connection for data set modification records (21), and to the data connection for data transfer and EOVS (20,22).

The IP address or port can be specified alone.

Example

```
LOCAL=138.42.220.120:13  
REMOTE=138.42.220.85  
LOCAL=:23
```

# Diagnostic Procedures

---

This chapter describes the problem determination and reporting procedures for Unicenter TCPAccess Communications Server.

This chapter discusses the following topics:

- [Problem Determination](#) – Describes problem diagnostic procedures and how to use the diagnostic commands
- [Problem Reporting](#) – Describes the documentation needed when reporting problems to customer support
- [Troubleshooting GateD](#) – Describes how to diagnose problems with GateD
- [Troubleshooting the CDLC Driver](#) – Describes procedures to use with the CDLC driver
- [Troubleshooting IUCV Sockets](#) – Describes procedures specific to IUCV sockets

## Problem Determination

If Unicenter TCPAccess does not respond properly, diagnostic action is required to restore proper service. This section provides a background in performing diagnostic action and gathering documentation for problem resolution.

Although there can be many different types of software failures, the majority of them fit into some general categories.

Whenever a problem is suspected, a good first step in problem determination is to check the T01LOG and JES message log (or console log) for any messages that may indicate the nature of the problem. You can use the IFS LOGGING command to enable more diagnostic messages, such as debugging messages. For more information, see [LOGGING](#).

## Failures That Produce Dumps

In the event of an abend, a system dump (SVC dump) is produced. This is usually accompanied by message T00IF050E in the message log.

- Save the dump/logs
- Complete Unicenter TCPaccess job output
- Forward to Customer Support

For certain abends, the dump is suppressed – for instance, if an external application passes an invalid buffer address. In this case, an abend message may appear in the log without an accompanying dump. If this happens frequently, the application should be identified and corrected.

## Failures Due To Network Problems

If the problem is reproducible, use the realtime network trace facility, TCPEEP, to diagnose the problem. For intermittent problems, use the TRACE CT command to initiate a non-wrapping trace of the network interface(s) involved. For more information, see [TCPEEP](#) and [TRACE](#).

## Failure of Unicenter TCPaccess to Respond

Certain problems can lead to a failure of Unicenter TCPaccess, or a major application, to respond. Try to identify the failing component.

## Failure of Major Applications

Determine whether the TCP stack is not responding, or whether the problem involves only an application (such as Telnet or FTP). Try to ping the Unicenter TCPaccess host from a workstation on the local area network. This tells you whether the network interface is alive and if the IP layer is responding. If there is no response to the ping, check the status lights on the network device, and note what they indicate.

Using a simple application such as the TSO PING command to send an outbound request will tell you whether the transport provider is active. You can use the TSO NETSTAT and SYSTAT commands or the operator console to query and cancel connections. For more information, see [PING](#) and [NETSTAT/SYSSTAT](#).

## Loops

Unicenter TCPAccess has loop detection code in most cases, to cancel a process that is looping, and take a diagnostic dump. If Unicenter TCPAccess is not responding, and using large amounts of CPU, it may be necessary to cancel the job. First, take a dump of the address space using the Unicenter TCPAccess SVCDUMP command, or, if that fails to respond, use the console DUMP command.

## Latch Contention

Unicenter TCPAccess uses a latch facility, called ILATCH, to serialize resource usage. If a latch or latches are not freed, other processes can be affected. Unicenter TCPAccess has automatic hung-latch detection code that normally issues message T00IF038W in this event. You can also use the ILATCH DISPLAY command to show any latches held over 60 seconds. Use the SVCDUMP command to take a diagnostic dump of Unicenter TCPAccess. Use the ILATCH FREE command to free any held latches, starting at the latch with the highest number, and working down. For more information, see ILATCH in the chapter “Unicenter TCPAccess Operation.”

## Storage shortages

If the amount of virtual storage used by Unicenter TCPAccess exceeds the threshold defined on the MAXSTGPCT parameter of the IFSPARM statement in the IJTFCGxx configuration member, it stops accepting new connections. This may be seen by TOPEN error messages in the T01LOG, accompanied by TPL dumps. Check the TPL return code at offset X'10' in the TPL to see whether it indicates a resource shortage condition. (See the Unicenter TCPAccess *Communications Server Unprefixed Messages and Codes* for a description of the TPL return codes). This condition may also be accompanied by message T00IF092W on the console.

Use the IFS POOL command and VSM command to display virtual storage allocation. Use the IFS SVCDUMP command to take a diagnostic dump of the Unicenter TCPAccess address space. In some cases, it may be possible to use NETSTAT to free up storage by canceling old or inactive sessions. For more information, see NETSTAT/SYSSTAT in the chapter “Diagnostic Commands.”

If the problem is chronic and involves a slow leak of virtual storage, use the SMF statement of the IJTFCGxx configuration member to turn on INTERVAL recording and record subtype 80. Be sure your system SMF parameters and exits (such as IEFU85) allow the recording of the Unicenter TCPaccess SMF record type. When the SMF data is collected, you can use the SMF Report Writer Program to view storage utilization and growth during the recording period. For more information, see SMF Report Writer Program in the chapter “Using the System Management Facility (SMF).”

**Note:** Use of the product ABEND-Aid is not recommended for Unicenter TCPaccess customers. Dumps formatted by this product do not include many of the control blocks required for problem analysis.

## Problem Reporting

When reporting a Unicenter TCPaccess problem to customer support, document as much information as you can to fully characterize the state of the system at the time of failure and any detectable failure symptoms. The exact documentation collected for a given problem varies, depending on the type of failure experienced.

In general, the following items should always be provided as initial documentation of a failure:

- The release and maintenance level of the product and function.  
**Note:** Optional products such as EPS and CPT have their own release levels independent of Unicenter TCPaccess.
- Always include job output.  
This includes:  
T01LOG  
SYSPRINT  
DNRLOG  
MAPLOG  
SNMLOG
- The console log, or the job log, if running in batch.
- The conditions leading up to the failure.
- Whether the failure is reproducible or a one-time occurrence.
- Whether this is a new failure or if the product was working until this happened (maintenance, more users, new application).

It is better to collect an abundance of initial documentation than wait to have Technical Support technicians request additional information later when the data may not be available (especially in the case of intermittent failures or hardware problems).

## ABENDs

In the event of an ABEND, collect the following information:

- The SVC dump of the failing address space.
- The job output of the Unicenter TCPaccess gateway at the time of failure, including the JES log, T01LOG, and any enabled traces. If the ABEND is in a Unicenter TCPaccess application (for example, Server Telnet or FTP), and you are running the application in an address space separate from the gateway, collect the job output from the application address space as well.
- A detailed description of the action immediately preceding the failure (if known) or conditions of the system (such as 4500 Telnet sessions).
- Whether the failure is reproducible or random. If reproducible, obtain a TCPEEP trace while recreating the failure. If random, you may be able to capture the trace data using the TRACE command to start a wrap-around trace. See [TCPEEP](#) and [TRACE](#) for more information.

## Incorrect Output—Telnet

For incorrect output through Telnet, collect the following information:

- A detailed description of the action immediately preceding the failure (if known) or conditions of the system (such as a large number Telnet sessions).
- The job output of the Unicenter TCPaccess gateway at the time of failure, including the JES log, T01LOG, and any enabled traces.
- Any client software involved, including the name of the vendor and the release number.
- Whether the failure is reproducible or random. If reproducible, obtain a TCPEEP trace and a VTAM buffer trace while recreating the failure.

## Incorrect Output—FTP

For incorrect output through FTP, collect the following information:

- A detailed description of the action immediately preceding the failure (if known) or conditions of the system (such as 100 terabyte transfer).
- The job output of the Unicenter TCPaccess gateway at the time of failure, including the JES log, T01LOG, and any enabled traces.
- A log or screen print of the failing transfer, including any messages.
- Any client software involved, including the name of the vendor and the release number, and the direction of transfer and file format; the contents only if requested.
- Whether the failure is reproducible or random. If reproducible, obtain a TCPEEP trace.
- You can also set parameters to collect information when running FTP:
  - For FTP2, specify TESTI
  - For FTP3, you can specify TRACE

See the *Unicenter TCPaccess Communications Server User Guide* for more information on these commands.

## Incorrect Output—DNR, EPS, and so forth

For incorrect output, collect the following information:

- A detailed description of the action immediately preceding the failure (if known) or conditions of the system.
- The job output of the Unicenter TCPaccess gateway at the time of failure, including the JES log, T01LOG, and any enabled traces.
- Any error logs and/or job log from the failing component.
- Whether the failure is reproducible or random. If reproducible, obtain a TCPEEP trace and a component-specific trace or debug log. Obtain a VTAM buffer trace only if requested.

## Hardware

For hardware problems, collect the following information:

- A detailed description of the action immediately preceding the failure (if known) or conditions of the system, and the MVS console log showing Unicenter TCPaccess messages.
- The job output of the Unicenter TCPaccess gateway at the time of failure, including the JES log, T01LOG, and any enabled traces.
- The model and microcode version of the network controller and the configuration in the channel path (length, devices in chain, presence of switches, and so forth).
- The front-panel display at the time of the failure; note whether the data displayed is static or changing.
- Note whether the failure is reproducible or random. If reproducible, obtain a TCPEEP trace and a GTF channel trace while recreating the failure. If random, notice whether resetting the controller and restarting Unicenter TCPaccess resolves it.

## Startup and Parameter Errors

For startup problems and parameter errors, collect the following information:

- The job output of the Unicenter TCPaccess gateway at the time of failure, including the JES log, T01LOG, and any enabled traces.
- A listing of the parameter and command members affected (especially any recent changes).
- Whether the failure is reproducible or random.

## TCP API Trace Procedures

Use the following steps to gather detailed documentation for API failures or C-sockets interface problems.

**CAUTION!** You are advised to gather the information all at one time since separate sessions have time stamps that do not match up between components:

1. Raise the size of the Unicenter TCPaccess internal IFS trace table. This keeps the IFS internal trace table from wrapping too quickly. In the Unicenter TCPaccess startup JCL there is a CMND parameter representing a member name in the SYSPROC DD library. You can set the IFS trace table to one MB by placing this command in the CMND member:

```
MODIFY TRACE ON SIZE(256)
```

To use a half megabyte or above the line storage for the IFS trace table, set SIZE(128) on the MODIFY command.

2. If a trace address space is not running for the Unicenter TCPaccess stack, start one now. See [TRACE](#) for more information on the TRACE command and the trace address space.
3. If Unicenter TCPaccess is not running, start it now.
4. Start a trace instance using [TCPEEP](#) or the [TRACE](#) command. Remember, it is always better to capture as much **relevant** data as possible the first time. In order to maximize the amount of relevant data captured, trace filters should be carefully chosen to avoid filling trace data sets with data not related to the problem session or application.

If the problem session is of short duration and is readily reproducible, it may be sufficient to use TCPEEP, or to write a wraparound trace to a relatively small data set, using the TRACE command. If the problem takes longer to develop, you will probably need to use the TRACE command, and write the output to a file that is sufficiently large to hold the data. Specify trace group TLI. If the problem involves network traffic, specify group NETIF. If possible, use filters (host names, port numbers, protocols) to limit the amount of data traced. Only trace the data if necessary.

5. Start the application and wait for the hang or failure to occur.
6. From the MVS console, issue the command:

```
F runtcp,SVC DUMP, JOB(jobname)
```

where *runtcp* is the name of the Unicenter TCPaccess stack, and *jobname* is the job name of the API application. This will take a single SVC dump of both address spaces.

7. Stop the trace or TCPEEP job

8. Send everything you've collected to Customer Support:
  - All the Unicenter TCPaccess job output including JES logs
  - The SVC dump of the Unicenter TCPaccess address space
  - All the job output from the failing application address space
  - The SVC dump of the failing application address space
  - The output from the TRACE or TCPEEP command

## Troubleshooting GateD

This section provides information on troubleshooting GateD. For more information on GateD commands, refer to the Unicenter TCPaccess *Users Guide*.

### GateD Logs

There are three logs associated with GateD:

GateD Log	Description
GTDLOG	Normal GateD messages are written to this log.
GTDERR	GateD error messages with a level greater than the syslog up to level are written to this log.
GTDTRC	GateD trace messages are written to this log.

The OPSFmon and ripquery tools are provided with Unicenter TCPaccess to help with GateD implementation. For more information, see OSPFMON and RIPQUERY in the chapter "Diagnostic Commands."

### Controlling GateD

ACTEST can be used to stop GateD.

To determine the *pta\_address*:

1. Issue the command **TASK GATED** and note the *pta\_address* that is returned.
2. Issue the command **POST *pta\_address* CLOSE**.

GateD stops after this command is issued, and restarts automatically, using the configuration member specified on the IP statement in member TCPCFGxx.

## Creating Formatted Dumps for GateD

GateD has a trace dump facility which formats many of the control blocks for use in debugging. You can PPOST the GateD task via ACTEST to signal it to perform special actions as described below.

Command	Description
PPOST gated-pta OUTPUT	Generate formatted dump.
PPOST gated-pta INPUT	Reinitialize GateD (that is, reread config).
PPOST gated-pta ATTN	Suspend or resume tracing (that is, toggle).
PPOST gated-pta USER	Perform immediate rescan of interfaces.
PPOST gated-pta CLOSE	Shutdown GateD (no change).

For more information about RIP and OSPF, refer to RFCs 1058 and 1247, respectively.

## Troubleshooting the CDLC Driver

Certain types of CDLC controllers, such as the 3745, are governed by NCP generation parameters and may be subject to VTAM commands dealing with PU activation and deactivation. If the CDLC controller has no NCP dependencies, no VTAM considerations are applicable.

Prior to Unicenter TCPaccess startup, the channel adapter PU dedicated for IP traffic should be in a PCTD2 state. After Unicenter TCPaccess initialization, the PU should go into an ACTIV state. When Unicenter TCPaccess is brought down, the following VTAM message appears and the PU returns to the PCTD2 state:

```
IST259I INOP RECEIVED FOR puname
```

If, upon starting Unicenter TCPaccess, you encounter this message:

```
T01LL106E Device dev_name: initialization sequence failed to complete (timeout).
Device shutdown initiated.
```

Use the following commands to reset the channel adapter PU that is dedicated to IP traffic:

```
v net, inact, id=puname
v net, act, id=puname
```

You should then receive the following message:

```
T01LL181I Device dev_name: interface if_name is fully operational.
```

**Note:** Unicenter TCPaccess need not be shut down to issue the vary commands.

Check the job output from the T01LOG and the MVS syslog output for CDLC driver ABENDs, device I/O errors, device allocation errors, and other such errors.

## Troubleshooting IUCV Sockets

The IUCV sockets replace the corresponding IBM TCP/IP IUCV transport address space and TCP/IP facilities and let existing IUCV-based applications execute transparently. You can use the debugging facilities that exist within those interfaces, as well as IUCV socket traces to diagnose problems that may appear when using this feature.

Potential problems fall into two broad groups:

- Startup of the IUCV sockets and interface to Unicenter TCPaccess
- Application-level problems seen during normal operations

### Diagnosing Application-Level Problems

Once the IUCV and Unicenter TCPaccess tasks are started, application programs can connect to the network via TCP/IP. At this point, problems are typically protocol related. There are two levels to consider:

- The IUCV transport
- The socket function calls

The socket libraries (IBM C socket library), and the macro socket and IUCV APIs, depend on the correct encapsulation of a socket-family call inside an IUCV parameter list, as well as a functional IUCV connection (path) between the application, the IUCV task, and the Unicenter TCPaccess task.

#### IUCV Transport

The application program using IUCV sockets uses two items of information to build a connection with Unicenter TCPaccess over IUCV (this is different than a socket connection, described in detail below).

- The IUCV subsystem name; by default, VMCF.
- The TCP/IP jobname; by default, TCPIP.

If your configuration matches these values, then connection should proceed without any error. However, the most commonly seen error code is in response to socket function calls. For example:

```
RC=1 on IUCV_CONNECT to tcpip, FD = -254, Path = nn, IPRCODE = 11
```

The IPRCODE of 11 indicates that the task name tcpip was not found or did not accept the connection. Verify that the step name of the Unicenter TCPAccess task matches the TCPJOBNAME of the socket library (the Macro API determines this name from a zappable constant in the Unicenter TCPAccess version of EZASOK03).

It is more difficult to diagnose an incorrect subsystem name because many applications expect that the VMCF subsystem is always present and these applications are not prepared to handle error cases.

These return codes are set by the IUCV sockets support routines when the subsystem is not found or not initialized:

- SNMPGPCN            4 (not found) or 8 (not valid)
- IUCVMULT            10103 (PC numbers not set by SNMPGPCN)
- IUCVSTRT/MAST Same as that set by IUCVMULT

**Note:** Actions by the IBM C socket library when the IUCV subsystem is not present or is not named VMCF are unpredictable.

## Socket Function Calls

Once the IUCV connection is made and a path exists to Unicenter TCPAccess, the actual socket function can proceed. The number and order of function calls depends on the application, but typically a socket() is allocated. For a server program, this is followed by a bind() and listen(); for a client program, it is followed by connect(). In this example, the function is a socket connect that takes place over an existing IUCV connection.

Problem determination at this level depends on tracking the function requested by the application, through the socket library, through IUCV, through Unicenter TCPAccess, and out to the network, then following the return status back through the same series of components to the application. The IUCV sockets supply some of the trace tools needed; the remainder exist either in the application, in the socket library, or on the network.

Applications may have some level of socket-call-level tracing that can be enabled via parameters or environment variables.

The IBM C socket library provides the SOCKDEBUG option in hlq.TCPIP.DATA to list the IUCV arguments and reply data to the SYSPRINT DD statement.

The Unicenter TCPAccess Macro API has a trace function that is enabled by zapping constants in the option module T02U80EZ. It then prints trace records to the designated DD statement. See [Macro API Diagnostic Facilities](#) for details of the trace.

Both the IUCV task (RUNIUCV) and the Unicenter TCPAccess task (RUNTCP) record trace data into an internal trace buffer when the TRACE is turned on and the TEST attribute is set to ON for the following task groups:

- IUCV IJT task group
- TCP TCP task group

This is accomplished via command statements in the STARTxx script executed at task startup. See Startxx Configuration in the chapter “Unicenter TCPAccess Operation” for detailed syntax of these statements. The samples provided with this feature are preconfigured with usable defaults.

Trace entries are written into the internal circular buffer once tracing has been enabled. They are formatted to sysout when the following operator command is entered:

- For IUCV, enter:  
`f runiucv,ijt snap all`
- For Unicenter TCPAccess, enter:  
`f runtcp,tcp snap all`

*runiucv*

IUCV address space name.

*runtcp*

Unicenter TCPAccess address space name.

The trace table being snapped exists in different task groups in the IUCV and TCP tasks. In addition to the trace table, selected control blocks are also formatted.

Trace entries contain details about conditions within the task that include control block addresses, register locations, and internal error indicators. Trace entry format is shown in the “IFS Internal Trace Entries” chapter of *Unicenter TCPAccess Communications Server Unprefixed Messages and Codes*. However, complete interpretation of the trace data is intended to be performed by Customer Support.

Unicenter TCPAccess can record the network message exchange via the TCPEEP tracing tool. For socket applications, select the TCP level of recording with the DATA option greater than 256.

When possible, collect traces at all points in the application-to-network path from a known, repeatable set of starting conditions. All traces should record the same event, with consistent time stamps. Accurate trace data is key to diagnosing protocol related errors.

## Diagnosing Other IUCV Failures

Other than the problems noted above, program checks (ABENDs) can occur or performance issues may be detected during otherwise normal operation.

In the case of ABENDs, it is critical to retain all the memory dump information, along with the sysout logs from *both* Unicenter TCPaccess and IUCV tasks. If the tasks remain functional, issue an **IJT SNAP ALL** or **TCP SNAP ALL** to the IUCV task and Unicenter TCPaccess task, respectively. In many cases, an SVC dump is taken automatically of all three address spaces (application, IUCV, Unicenter TCPaccess) and recorded into a SYS1.DUMPxx data set. This dump is intended to be analyzed via the IPCS tool, so formatted printing is not recommended. If possible, note the applications running at the time and any other related messages. If an application-task dump occurs, retain that also.

Performance problems are diagnosed by obtaining IUCV and Unicenter TCPaccess internal traces, which contain very accurate time stamps, along with a TCPEEP trace to monitor the data flow.

## Reporting IUCV Failures

When a failure occurs, collect documentation as appropriate (see the previous sections for details) and contact Customer Support. They may request that you send the items of documentation in machine-readable form (in other words, not on paper). More rapid diagnosis can be made if documentation can be analyzed with online tools.

Be prepared to describe the environment that existed at the time of the failure and if the failure is repeatable (and if so, under what circumstances) or if it is an isolated occurrence.

## Macro API Diagnostic Facilities

The configuration and diagnostic capabilities available with the Macro API runtime support module, EZASOK03, are described in this section.

The load module T02U80EZ (alias EZASOK03) must be in an APF authorized library. Several application defaults can be set or overridden by zapping the defaults CSECT IUCVCONS in module T02UIUCV.

Output from the Macro API Trace is written to the DD name chosen (default SYSPRINT) in the client job or address space.

This table describes the values of the CSECT IUCVCONS:

Option Name	Offset	Value	Description
TCPIPJOBNAME	8(x'08')	CL8' '	As delivered, EZASOK03 lets the application set this value with the INITAPI call.  If this field is zapped to some other value, the INITAPI value is overridden at run time.
DNRSSID	16(x'10')	CL4'ACSS'	This value is only used if the application uses the TYPE=GETHOSTBYADDR or TYPE=GETHOSTBYNAME functions.
VMCFSSID	20(x'14')	CL4'VMCF'	VMCF subsystem ID
TRACEFLAG	24(x'18')	X'00'	As delivered, tracing is turned off. X'80' = TRACING ON
TRACESIZE	26(x'1A')	H'25'	As delivered, the default trace table size is limited to 25 entries before wrapping.  This value is only used if the TRACEFLAG value is set to ON.
TRACEDD	28(x'1C')	C'SYSPRINT'	As delivered, the default trace DD statement name is SYSPRINT.  This value is only used if the TRACEFLAG value is set to ON.  If the OPEN fails for this DD statement, no tracing is attempted.

**Note:** If you are going to change the Unicenter TCPAccess jobname and so forth, use USERMOD MU1IUCV, found in the SAMP library. It is the preferred way to use SMP/E for zapping.

## Examples

Examples of the character formatted socket trace and the in-memory trace table are in the IUCVNOTE member of the SAMP library.

Example

This job is an example of how to zap application defaults/overrides:

```

/*
/* ZAP FOR SETTING EZASOK03 CONFIGURATION OVERRIDES/DEFAULTS
/*
//STEP EXEC PGM=AMASPZAP
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DSN=TCPICS.Vxxx.LOAD,DISP=SHR
//SYSIN DD *
NAME T02UIUCV IUCVCONS
REP 0008 E3C3D7C9D7404040 'TCPIP' - TCPIP ADDRESS SPACE NAME
REP 0010 C1C3E2C8 'ACSH' - DNR SUBSYSTEM ID
REP 0014 C9D5D3D2 'INLK' - VMCF SUBSYSTEM ID
REP 0018 80 TRACE ON
REP 001A 0032 50 INCORE TRACE TABLE ENTRIES
REP 001C E2E8E2E3D9C1C3C5 SYSTRACE - DD STATEMENT FOR
* CHAR TRACE
/*

```

The tracing output made available by zapping T02UIUCV comes in two parts:

- An in-memory trace table that records the parameter lists
- A character formatted socket trace

USERMOD

```

++ APAR (MU1IUCV)
/* TCP/IP CORRECTIVE MAINTENANCE */
++ VER (Z038)
FMID(TCP0xxx)
PRE(TPXXXXX)
/*
PROBLEM DESCRIPTION (PSR052177):
=====
* PROBLEM SUMMARY:
* ZAP IUCV CONSTANTS FORMODULES
*
* RECOMMENDATION: APPLY THE APAR TO CHANGE THE CONSTANTS.
*
* PROBLEM DESCRIPTION: CHANGE IUCV CONSTANTS
*
=====
* CJB *
THE FOLLOWING MODULES/MACROS ARE AFFECTED BY THIS PTF:
LIST BEGIN
T02UIUCV
LIST END
*/

```

```

++ZAP(T02UIUCV) DISTLIB(ATCPLoad) .
NAME T02UIUCV IUCVCONS
REP 0008 E3C3D7E5F4F1F040          TCPV410
REP 0010 C1C3E2E9                  ACSZ
REP 0014 C9E4C3E5                  IUCV
REP 0018 8000                      X'80' TRACE FLAG ON
REP 001A 0032                      50 INCORE TRACE TABLE ENTRIES
REP 001C E2E8E2E3D9C1C3C5        SYSTRACE - DD STATEMENT
//SMPCNTL DD *
SET BDY(GLOBAL) .
RECEIVE S(MU1IUCV) .
SET BDY(TCPTZN) .
APPLY S(MU1IUCV) .
//

```

### Character Formatted Socket Trace

There are two trace records written to the trace DD statement for each function being traced

- The first record, [character formatted socket trace](#), is written before the PC call, the call to IUCVMULT, or the dirsrv request (get...by... emulation)
- The second record, in-memory trace table record, is written after the function has completed.

Field #	Description
1	The subtask name supplied by the Macro API caller on the INITAPI call.
2	The socket function, IUCV function, or dirsrv request.
3	The IUCV path (if applicable).
4	The value in IPTRGCLS+2, typically the socket number (if applicable).
5	The return code. On successful calls this field often contains socket call result information. For example, on a successful read(), this field contains the number of bytes read. On unsuccessful socket calls, this field contains -1.

<b>Field #</b>	<b>Description</b>
6	If Field #5 contains -1, Field #6 contains the errno value (whose meaning can be found in the header library member TCPERRNO). On a successful INITIAL MESSAGE call, Field #6 contains the maximum socket value returned by TCP.
7	Field #7 in the character formatted socket trace corresponds to the trace entry type that gets generated and written to the in-memory table. The TYP5 record on an IUCVMINI SET function gives the address of the corresponding in-memory trace table. There is a separate in-memory trace table for every TCB (TASK=). The address on the TYP5 record can be used to find the correct in-memory table when a dump of the application region is taken.

# Diagnostic Commands

This chapter describes the diagnostic commands available in Unicenter TCPaccess.

It includes the following topics:

- [Troubleshooting Client and Server FTP3](#) – Describes guidelines for problem isolation with Unicenter TCPaccess FTP compatibility
- [ACTEST](#) – A customer support command to debug the Unicenter TCPaccess commutator
- [DNRGET](#) – A command to query the Domain Name Resolver (DNR)
- [NETSTAT/SYSSTAT](#) – The diagnostic command to monitor and manage Unicenter TCPaccess activity
- [OSPFMON](#) – A command to request information for routes known to OSPF
- [PING](#) – A command to determine if a host is active on the network
- [RIPQUERY](#) – A command to request all routes known by a RIP gateway
- [RPCINFO](#) – A command to query an RPC server to get information about program names and program numbers for a specified host
- [STROBE](#) – A command to measure the amount of CPU time used by a module
- [TCPEEP](#) – A command to invoke the Unicenter TCPaccess packet trace program and diagnose remote host communication problems
- [TRACE](#) – Trace command enhancements to collect TCP/IP data and display it on a terminal or send it to an external writer
- [TRACERT](#) – A command to print the route that packets take to get to a network host

## Troubleshooting Client and Server FTP3

This section describes problem isolation in the Unicenter TCPaccess FTP compatibility feature. Troubleshooting information is divided into server and client. The client is a three-party model; it therefore requires both server and client documentation.

### General Guidelines for Problem Isolation

When encountering FTP3 problems, start with the following problem resolution process.

1. Determine the level of maintenance that is running.
2. Enable the DEBUG or TRACE option and rerun the failing command.
3. Run a network trace.
4. Look at the T01LOG job output.

### Resolving Server FTP Problems

For incorrect input through FTP, collect the following:

- A description of the action preceding the failure and the condition of the system (such as 100 terabyte transfer)
- The job output of the Unicenter TCPaccess gateway at the time of failure, including:
  - The JES log
  - Any PSNAP data
  - T01LOG
  - Any enabled traces
- A log or screen print of the failing transfer, including any messages
- Any client software involved, including the name of the vendor, the release number, the direction of transfer, the file format; and the contents (if requested)
- Whether the failure is reproducible or random  
If reproducible, get a TCPEEP TCP trace.

## Resolving Client FTP Problems

For incorrect input through FTP, collect the following information:

- A description of the action preceding the failure and the condition of the system (such as 100 terabyte transfer).
- The job output of the Unicenter TCPaccess gateway at the time of failure, including:
  - The JES log
  - Any PSNAP data
  - T01LOG
  - Any enabled traces
- A log or screen print of the failing transfer, including any messages.
- Turn on the TRACE or DEBUG parameters. The recommended procedure is to run the command in batch mode and save the recorded trace output for problem resolution.
- Any client software involved, including the name of the vendor, the release number, the direction of transfer, the file format; and the contents (if requested).
- Whether the failure is reproducible or random. If reproducible, obtain a TCPEEP TCP trace.

## ACTEST

The Telnet command ACTEST invokes the ACTEST parameter and debugging interface to the Unicenter TCPaccess commutator. A generalized terminal I/O interface implements it. Typically, it is used only on advice of Customer Support.

To invoke ACTEST

- From Server Telnet, through an ATPUT/ATGET interface, select ACTEST at the Server Telnet command screen (port 23). It requests logon and calls PACCESS to verify privileges.
- From User Telnet (TSO), use VTAMTEST as the remote host name. The path requires a local logon. It is particularly useful when there is a problem with TCP/IP, since no network I/O is required.

The following list describes the ACTEST options:

ABEND	Abruptly terminates the entire commutator.
ACCOM <i>length</i>	Lists length (in bytes) of the main control block.

ALIST <i>address length</i>	Displays storage at address within unprotected areas of the address space for length bytes; tries to print the information as ASCII; specifies address and length in hex.
CATV <i>pta_1 pta_2</i>	Copies the application transfer vector (PTAATRV) from the PTA at address <i>pta_2</i> to the PTA at address <i>pta_1</i> .
COMMANDS	Lists all the commands known to ACTEST.
END	Terminates the ACTEST ptask normally.
EQUATE <i>symbol address length</i>	Defines a symbol to be used by LIST.
EXECUTE <i>slot</i>	Issues the commands saved in slot by the REMEMBER command.
FREEZE	Issues a PWAIT FREEZE and halts execution of the ACTEST ptask until external action is taken to unblock the ptask.
HELP <i>command</i>	Displays the syntax for command.
LIST <i>address length</i>	Displays storage at address within unprotected areas of the address space for length bytes. Tries to print the information as EBCDIC.
MODULES <i>mask</i>	Lists the loaded modules or ESDs that have names matching mask (* is a wildcard character).
PATTACH <i>module parms</i>	Creates a new control fork to pass control to the indicated module and to pass it the indicated parameter string in the form defined for the EXEC PARM field; each fork is identified by a control block called the Pseudo Task Area (PTA)
PDELETE <i>name</i>	Removes from storage the module loaded by PLOAD called name.
PDETACH <i>pta</i>	Destroys the PTA at address <i>pta</i> that was created by PATTACH. Explicitly forces PTA at a given address to exit.  Use this to install a new load module within the running Unicenter TCPaccess when the old module has been PLOADed by a permanent ptask. Issue PDETACH for the ptask to free the old module, and issue a new PATTACH to bring in the new module
PLOAD <i>name options</i>	Loads the program module name into main storage where it stays until you exit from ACTEST.  Possible options are: ESD, FETCH, FREE, HOLD, SYS, and USE
PPOST <i>ecb code</i>	Marks event ecb complete with the code <i>code</i> .

---

<i>PPOST pta event_list</i>	Marks <i>pta</i> as no longer blocked on certain events. Possible events are ATTN, CLOSE, CORE, INPUT, OPEN, OUTPUT, and USER.
<i>PSTATUS ANCESTOR pta</i>	Lists all ancestors of the PTA at address <i>pta</i> .
<i>PSTATUS DESCENDENT pta</i>	Lists all descendents of the PTA at address <i>pta</i> .
<i>PSTATUS PARENT pta</i>	Lists all parents of the PTA at address <i>pta</i> .
<i>PSTATUS PROCESS mask</i>	Lists all the PTAs that have names matching the mask (* is a wildcard character).
<i>PSTATUS SON pta</i>	Lists all sons of the PTA at address <i>pta</i> .
<i>PTALIST mask</i>	Lists all the PTAs that have names matching the mask along with the status and contents of the registers of the PTA (* is a wild card character).
	The command PTA I displays the status of all PTAs whose names start with I (for example, INPTASK, IMPIN, IMPOUT and so forth.).
	The PTA display formats the dispatchability and semaphore bits in the first word of the PTA and shows the registers at the time of the last P- service call (normally PWAIT).
<i>PWAIT integer</i>	Suspends the ACTEST ptask to let the commutator dispatch other ptasks (that is, this lets Unicenter TCPaccess execute for a time specified by integer). Here integer is in units of 0.01 seconds. To specify a decimal integer, follow the integer with a period.
<i>REMEMBER slot command_line</i>	Saves the <i>command_line</i> in slot for later execution. The text of <i>command_line</i> is saved and later given to the parse routines by the EXECUTE command.
	Saved commands can include nested EXECUTE commands. There are sixteen slots numbered 0 - 15.
<i>RESOURCE PTA_address mask</i>	Displays those resources of the PTA at the specified address that have names matching mask (* is a wildcard character in mask). If mask is not specified, all resources owned by the PTA display.
	Examples:
	The following example lists the PCOREd areas belonging to the PTA at address 56A00.
	RESOURCE 56A00 PCORE or RES 56A00 PC

---

The following example displays the AEXIT resource for the specified PTA. This includes all the TCP/IP and environmental control blocks.

```
RES 56A00 AEXIT or RES 56A00 AE
```

The following example displays all the resources owned by the specified PTA.

```
RES 56A00  
RUN transfer_initialization_parameters
```

Starts an application program. The module identified by transfer is loaded into core and treated as the Application Transfer Vector (ATRV).

The module identified by initialization is then forked by PATTACH (specifying the option that the new ptask is to have no parents) with the ATRV and given the parameters in EXEC PARM form.

SHOWREM <i>slot</i>	Displays the command line saved in <i>slot</i> by the REMEMBER command.
SPOOLDMP ON   OFF	Specifies whether ptask SPOOL#4 should take a dump at certain task termination events (open, close, dynamic allocation, and so forth).  <b>Note:</b> SPOOLDMP entered without any parameters displays the current setting.
SRUN <i>pta</i>	Makes the PTA at address <i>pta</i> ready to run.
START <i>pta</i>	Revives the PTA at address <i>pta</i> that was suspended with STOP.
STOP <i>pta</i>	Suspends the PTA at address <i>pta</i> from running.
STORE <i>data</i>	Replaces storage with data at the last address displayed with the LIST command.
SUBPOOLS	Displays allocated, used, and free virtual storage in all used subpools above and below the 16 MB <i>line</i> , including totals.
SVC99	Activates the extended error processing from SVC99. These extended messages are issued to the master console by the SVC99 error processing facility.
SYSDUMP ON   OFF	Specifies whether a system dump should be produced for commutator ABEND conditions, in addition to the formatted SNAP dump that is normally produced.  The system dump goes to the SYSUDUMP, SYSABEND, or SYSMDUMP data set.  Default: OFF.
TASKS mask	Alternate for the PSTATUS PROCESS mask command. Gives a one-line status display for each ptask whose name starts with the string mask.  <b>Note:</b> If mask is omitted, the status display lists all current ptasks

---

VLT ON   OFF	<p>VLT tracing. VLT OFF stops VLT tracing; VLT ON starts VLT tracing.</p> <p>Uses module AMDUSR to format the GTF records written by the VLT trace.</p> <p>Default: VLT OFF.</p>												
WHERE <i>address</i>	<p>Locates the given address within the current set of loaded modules, if possible, and prints its relative position.</p>												
XSEC <i>options</i> ON   OFF	<p>Activates or deactivates monitoring of selected security events. Those options can also be specified on the SECURITY statement in member IJTFCGxx.</p> <p><b>Note:</b> Global system security options can be turned on or off by specifying ON or OFF.</p> <table><tr><td>ACEE</td><td>Monitors ACEE activity.</td></tr><tr><td>ACSECP</td><td>Monitors entry to the security call module that performs all security checking within Unicenter TCPaccess.</td></tr><tr><td>COMMAND</td><td>Monitors command authorization calls. The COMMAND option aids in the debugging of COMMAND security problems within Unicenter TCPaccess</td></tr><tr><td>DATASET</td><td>Monitors data set authorization calls. The DATASET option aids in the debugging of DATASET security problems within Unicenter TCPaccess.</td></tr><tr><td>LOGOFF</td><td>Monitors departure from the system. The LOGOFF option aids in the debugging of sign-off security problems within Unicenter TCPaccess.</td></tr><tr><td>LOGON</td><td>Monitors attempts to gain entry to the system. The LOGON option aids in the debugging of signon security problems within Unicenter TCPaccess.</td></tr></table>	ACEE	Monitors ACEE activity.	ACSECP	Monitors entry to the security call module that performs all security checking within Unicenter TCPaccess.	COMMAND	Monitors command authorization calls. The COMMAND option aids in the debugging of COMMAND security problems within Unicenter TCPaccess	DATASET	Monitors data set authorization calls. The DATASET option aids in the debugging of DATASET security problems within Unicenter TCPaccess.	LOGOFF	Monitors departure from the system. The LOGOFF option aids in the debugging of sign-off security problems within Unicenter TCPaccess.	LOGON	Monitors attempts to gain entry to the system. The LOGON option aids in the debugging of signon security problems within Unicenter TCPaccess.
ACEE	Monitors ACEE activity.												
ACSECP	Monitors entry to the security call module that performs all security checking within Unicenter TCPaccess.												
COMMAND	Monitors command authorization calls. The COMMAND option aids in the debugging of COMMAND security problems within Unicenter TCPaccess												
DATASET	Monitors data set authorization calls. The DATASET option aids in the debugging of DATASET security problems within Unicenter TCPaccess.												
LOGOFF	Monitors departure from the system. The LOGOFF option aids in the debugging of sign-off security problems within Unicenter TCPaccess.												
LOGON	Monitors attempts to gain entry to the system. The LOGON option aids in the debugging of signon security problems within Unicenter TCPaccess.												

## DNRGET

The DNRGET TSO diagnostic command invokes the command processor that allows direct queries to be made to the Domain Name Resolver (DNR). DNRGET can issue all sixteen combinations of domain name requests supported by the DNR. The full range of options that the DNR can support can be specified.

Each reply returned by the DNR is formatted and displayed. The format of the returned data is dependent on the type and mode of the request. Any conditional completion codes or error codes are formatted and displayed. Along with these codes, a textual message explaining the code is displayed.

DNRGET executes as a TSO command processor. All messages that are written and done with TPUTs restricting its use to interactive TSO users only. In addition, the TSO profile option PROMPT must be set if DNRGET is to prompt for correct operands in the event of an error.

DNRGET cannot be invoked with the TSO CALL or TSOEXEC commands. Therefore, the LINK data set must be either in your system's link list or in the STEPLIB DD concatenation of your TSO JCL procedure.

If, after invocation, DNRGET waits too long for a reply from the DNR, press PA1 to interrupt its execution without causing any errors.

The syntax of the DNRGET command is as follows:

```
DNRGET request_type request_mode
      request [ Block|Noblock ] [ Copy|Nocopy ] [ Local|Global ] [ SIZE(count)]
              [ SYSid(subsystem_id) ] [ Time(seconds)]
```

**Note:** At a minimum, you must specify the letters that are capitalized of the request parameters.

*request\_type*

Specifies the type of Domain Name System information being requested.

Valid values are:

HOST	Resolves host names and Internet addresses.
HOSTINFO	Returns the CPU type and operating system of a host.
HOSTSERV	Returns which services a host provides.
NETWORK	Resolves network names and Internet network numbers.
NS	Returns list of authoritative name servers for domain.
PROTOCOL	Resolves transport protocol names and Internet protocol numbers.

---

	ROUTE	Returns mail routing information for a host.
	RPC	Resolves RPC program names and numbers.
	SERVICE	Resolves well known protocol services.
	SOA	Returns start of authority data for a domain
<i>request_mode</i>		Specifies the format of the request data
		Valid values are:
	BYALIAS	The request field is a host name ALIAS. This option returns a replacement name for the requested host name.
	BYNAME	The request field is a host name, network name, protocol name/service name or protocol name. BYNAME is valid with any request type.
	BYVALUE	The request field is either an Internet number in <i>nnn.nnn.nnn.nnn</i> format, a TCP/UDP service number, or a protocol number. BYVALUE is only valid with request types HOST, NETWORK, SERVICE, and PROTOCOL.
<i>request</i>		Specifies the data that the Domain Name Resolver used as input. Its format is dependent on the <i>request_type</i> and <i>request_mode</i> arguments.
		Valid values are:
	Block   Noblock	Block specifies that a response is not returned until either the request is completed or the time or retry limits are exceeded.
		Noblock specifies that a response is returned immediately. If the data to complete the request is not in cache, an error is returned immediately, but DNR continues to try to resolve the request in order to update the cache.
		This argument only applies if Global is specified.
		Default: Block.

Copy   Nocopy	<p>Copy specifies that if data to resolve a request is in the DNR cache it uses it to satisfy the request.</p> <p>Nocopy specifies that DNR ignores the data in the cache and queries domain name servers for data to resolve the request.</p> <p>This parameter only applies if Global is specified.</p> <p>Default: Copy.</p>
Local   Global	<p>Local specifies that the DNR use only configuration data to resolve requests.</p> <p>Global specifies that the DNR query Domain Name Servers for the necessary data to resolve the request.</p> <p>Default: Global.</p>
Size(count)	<p>Specifies the number of entries to be returned by DNR when the result field has more than one item in the reply.</p>
count	<p>he maximum number of entries to be returned. 0 (zero) specifies that the maximum number of entries is determined by the return buffer size, which cannot be dynamically changed.</p> <p>Default: Zero.</p>
SYsid ( <i>subsysid</i> )	<p>sSpecifies the subsystem ID of the Unicenter TCPaccess job or started task in which the DNR is active. <i>subsystem_id</i> is the four-character MVS subsystem ID.</p> <p>Default: ACSS.</p>
Time( <i>seconds</i> )	<p>Specifies the maximum amount of time the DNR is allowed to resolve a request.</p>
<i>seconds</i>	<p>The maximum amount to process the request in seconds. 0 (zero) specifies that there is no time limit to resolve the request.</p> <p>Default: 30 seconds.</p>

## NETSTAT/SYSSTAT

The NETSTAT and SYSSTAT diagnostic commands invoke the user information and operator control interface for Unicenter TCPAccess. Use these diagnostic commands to monitor and manage Unicenter TCPAccess activity.

The TCP task group must be active. System security adjustments should be made if the new management commands require authorization

Information retrieval is from a running, operational gateway, so commands associated with the NETSTAT facility can originate outside the gateway address space.

Commands can originate from the following.

- The system console operator
- A TSO user
- A remote user via Telnet

NETSTAT commands that interrogate only are unrestricted commands. Management commands resulting in dynamic modification of internal control blocks or session behavior require authorization, unless the source is the system console operator.

**Note:** SYSSTAT commands entered under TSO are secure based on the TSO user ID.

## Command Syntax

The external syntax of commands varies slightly, depending on the origin. Operator commands start with some sequence that identifies the target address space.

For example, to request a NETSTAT connection display, use one of the following operator commands:

```
F jobname,NETSTAT CONN
```

or

```
%NETSTAT CONN
```

where % is the subsystem recognition character.

A remote Telnet user would use the following command:

```
NETSTAT CONN
```

A TSO version can include TSO-specific keywords as shown below:

```
NETSTAT CONN SSID( ABCD )
```

## Dotted Notation IP Address Wildcarding ( \* )

In most cases where a dotted notation IP address is accepted, wildcarding or omission adheres to the following syntax:

```
nnn < .nnn | * > < .nnn | * > < .nnn | * >
```

The usual dotted notation consists of four numeric components, each between 0-255, and separated by a period. For example:

```
111.112.113.4
```

If a component is omitted, right to left, it is replaced with 0, so 111.112.113 is interpreted as a specific IP address of 111.112.113.000, and 111.112 is interpreted as 111.112.000.000.

If a component is replaced with an asterisk (\*), it is interpreted as a non-specific wildcard.

So:

- 111.112.113.\* satisfies any IP address match starting with 111.112.113
- 111.112.\* satisfies any IP address starting with 111.112

Wildcarding must be applied to an entire component. In other words, 11\* is not valid. Both wildcarding and omission must be rightmost in the IP address string. That is, 111.\*.113.4 is not valid.

## TSO Command Common Parameters

The DDNAME( *name* ) and SSID( *name* ) parameters are common to all TSO commands.

```
NETSTAT CONN DDNAME( ddname ) SSID( subsysname )
```

DDNAME( *ddname* ) If the standard TSO output definition SYSTSPRT is not appropriate, an output file can be preallocated with an LRECL size of 120 bytes. Its DD name can then be used as a parameter to the TSO NETSTAT or SYSSTAT command.

This option is particularly useful if archiving is required, and browsing a disk file may be convenient.

If output is directed to a file specified by the DDNAME( *name* ) parameter, all message lines are date and time stamped.

SSID( *subsysname* ) Specifies the subsystem ID.

## NETSTAT Diagnostic Commands

This section describes the NETSTAT diagnostic commands.

### ARP

Displays the status of devices that responded to a request or are pending a request. The request status is retrieved from cache.

```
NETSTAT ARP[ IPNAME ] [ MEDIA( name ) ] [ MSGID | s ]
```

null Display all devices.

IPNAME Attempt to resolve the symbolic IP host name. If not resolvable, dotted notation displays.

MEDIA( *name* ) Limit the display to a specific media, as defined in TCPCFGxx.

MSGID | NOMSGID Controls the display of the nine-character message prefix.

- For Telnet users, NOMSGID is the default; use MSGID to turn on the message display
- For TSO users, MSGID is the default; use NOMSGID to omit the prefix from the display
- For operator requests, the prefix is always displayed; this parameter has no effect

**Note:** CLAW, CDLC, and HYPERCHANNEL do not provide any ARP information to Unicenter TCPAccess. ARP data is received only from controllers running CETI or 3172.

## HELP

Displays a summary of the NETSTAT and SYSSTAT commands.

```
NETSTAT HELP [ command ] [ MSGID | NOMSGID ]
```

null

Display a list of all the NETSTAT commands.

*command*

Display a specific command, description and syntax.

MSGID | NOMSGID

Controls the display of the nine-character message prefix.

- For Telnet users, NOMSGID is the default; use MSGID to turn on the message display
- For TSO users, MSGID (default) use NOMSGID to omit the prefix from the display
- For operator requests, the prefix is always displayed; this parameter has no effect

## CNFG

Display configuration information in one of the following formats.

```
NETSTAT CNFG [ MSGID ]  
NETSTAT CNFG LNI [CETI | CLAW] [ MSGID ]  
NETSTAT CNFG TCP | UDP | RAW | ALL [ MSGID ]  
NETSTAT CNFG BINDSEC [ MSGID ]
```

null

Displays the basic network in an outline form.

MSGID

Controls the display of the nine-character message prefix.

- For Telnet user, the default is no message. Use MSGID to turn on the message display
- For TSO users, MSGID is the default; using no message omits the prefix from the display
- For operator requests, the prefix is always displayed; this parameter has no effect

**LNI [CETI | CLAW]** Displays configuration information from the TCPCFGxx statement.

The LNI selection displays Local Network Interface information, and TPI displays Transport Provider Information variables.

If the CETI option is selected for a CETI device, then four additional messages are displayed.

If the CLAW option is selected for a CLAW device, then three additional messages are displayed.

**TCP | UDP | RAW** Protocol identifiers. Use TCP, UDP, and RAW to limit the display to specific protocols, otherwise all protocols are displayed.

**BINDSEC** Displays bind security configuration information.

## CONN

Gives connection summary information for specified connection.

```
NETSTAT CONN
  [ COUNT ] [ FULL ] [ MSGID | NOMSGID ] [ SEP ] [ TCP | UDP | RAW ]
  [ LOCPORT( nnn...nnn... ) ] [ REMPORT( nnn...nnn... ) ]
  [ LOCHOST( ip_addr ) ] [ REMHOST( ip_addr ) ]
  [ JOB( name ) ] [ PLU( name ) ] [ SLU( name ) ] [ SESSION( nnn ) ]
```

**COUNT** Specifies that a one-line connection count be displayed.

**FULL** Specifies that the full output be displayed, with no filtering ( approximately 23 lines per connection).

**MSGID | NOMSGID** Controls the display of the nine-character message prefix.

- For Telnet users, NOMSGID is the default; use MSGID to turn on the message display
- For TSO users, MSGID is the default; use NOMSGID to omit the prefix from the display
- For operator requests, the prefix is always displayed; this parameter has no effect

**SEP** Specifies that a separator, a single line with an asterisk (\*) be placed between connections.

This is useful when many connections are displayed

**TCP | UDP | RAW** Protocol identifiers. Use TCP, UDP, and RAW to limit the display to specific protocols, otherwise all protocols are displayed.

LOCPORT( *nnn ...nnn* ) Displays one or more sessions, using a local port number (*nnn*). You may specify individual ports, but not a range of ports

REMPORT( *nnn ...nnn* ) Displays one or more sessions, using a remote port number (*nnn*). You may specify individual ports, but not a range of ports.

LOCHOST( *ip\_addr* ) Displays a connection to a local host. *ipaddr* is the IP address of the host

**Note:** The IP address must be entered in dotted notation.

REMHOST( *ip\_addr* ) Display a connection to a remote host. *ipaddr* is the IP address of the host.

**Note:** The IP address must be entered in dotted notation.

JOB( *name* ) Specifies that the output be filtered for the job name given.

PLU( *name* ) Specifies that the output be filtered for the Primary LU given.

SLU( *name* ) Specifies that the output be filtered for the Secondary LU given.

SESSION( *nnn* ) Display a session. *nnn* represents a specific session number.

## RESOLVE

The RESOLVE command accepts either an IP address in dotted notation, or an IP name character string, and attempt to resolve one format into the other. The resolution is dependent on the DNR configuration.

NETSTAT RESOLVE *IPaddr* | *IPname* MSGID | NOMSGID

IPADDR | IPNAME Specifies either the IP address (in dotted decimal notation) or the IP name.

MSGID | NOMSGID Controls the display of the nine-character message prefix:

- For Telnet users, NOMSGID is the default; use MSGID to turn on the message display
- For TSO users, MSGID is the default; use NOMSGID to omit the prefix from the display
- For operator requests, the prefix is always displayed; this parameter has no effect

## ROUTE

Displays the routing table entries.

```
NETSTAT ROUTE [ IPNAME ] [ MASK ] [ MSGID | NOMSGID ]
```

null

Displays a list of all the NETSTAT commands.

IPNAME

Attempts to resolve the symbolic IP host name. If not resolvable, dotted notation will display.

MASK

Specify this option to display the subnet mask for route entries.

MSGID | NOMSGID

Controls the display of the nine-character message prefix.

- For Telnet users, NOMSGID is the default; use MSGID to turn on the message display
- For TSO users, MSGID is the default; use NOMSGID to omit the prefix from the display
- For operator requests, the prefix is always displayed; this parameter has no effect

## RTM

```
NETSTAT RTM
```

Displays a continuous activity report on all active hardware interfaces

NETSTAT RTM can be used by a Telnet user to display a continuous activity report on all active hardware interfaces. The frequency is determined by the value in seconds, and defaults to nine if not entered. Permissible values are between 1 and 32767.

Once activated, the RTM command will continue to display a time-stamped connection counter, and LNI counts and rates. The display itself appears after the specified interval in seconds, but the rate is based on a time difference between now and the last sampling. If multiple RTM commands are in effect, the rates will not always be consistently smooth.

To terminate the RTM command, press Enter.

**TELNET**

Displays the status of Telnet sessions.

```
NETSTAT TELNET [ SESSION (nnn) ] [ REMHOST (ip_addr) ]  
               [ APPL (name) ] [ PLU (name) ] [ SLU (name) ]  
               [ LOCPORT (nnn...nnn) ] [ REMPORT(nnn...nnn) ]  
               [ FULL ] [ DETAIL ] [ MSGID ] [ NOMSGID ]
```

**SESSION (*nnn*)** Number of the session for which to display the status information

If this parameter is not specified, status information for all sessions is displayed.

The value specified must be expressed as a numeric quantity.

**REMHOST (*ip\_addr*)** Remote host IP address of the sessions for which to display the status information. If this parameter is not specified, status information for all sessions is displayed.

The value specified must be expressed as a numeric quantity using dotted decimal notation.

**APPL (*name*)** VTAM application name of the sessions for which to display the status information. If this parameter is not specified, status information for all sessions is displayed.

The value specified must be from one to eight bytes in length and may be expressed as a quoted character string ('ABC') or a non-quoted character string where a separator or delimiter indicates the end. The name specified can include wildcard characters with the character % denoting a single mask character and the character \* denoting zero or more mask characters.

**PLU (*name*)** Primary LU name of the sessions for which to display the status information. If this parameter is not specified, status information for all sessions is displayed.

The value specified must be from one to eight bytes in length and can be expressed as a quoted character string ('ABC') or a non-quoted character string where a separator or delimiter indicates the end. The name specified may include wildcard characters with the character % denoting a single mask character and the character \* denoting zero or more mask characters.

**SLU (*name*)** Secondary LU name of the sessions for which to display the status information. If this parameter is not specified, status information for all sessions is displayed.

The value specified must be from one to eight bytes in length and may be expressed as a quoted character string ('ABC') or a non-quoted character string where a separator or delimiter indicates the end. The name specified can include wildcard characters with the character % denoting a single mask character and the character \* denoting zero or more mask characters.

LOCPORT ( <i>nnn...nnn</i> )	<p>Local port number of the sessions for which to display the status information. If this parameter is not specified, status information for all sessions is displayed.</p> <p>The value specified must be expressed as a numeric quantity.</p>
REMPORT ( <i>nnn...nnn</i> )	<p>Remote port number of the sessions for which to display the status information.</p> <p><b>Note:</b> If this parameter is not specified, status information for all sessions is displayed.</p> <p>The value specified must be expressed as a numeric quantity.</p>
FULL	<p>Indicates that detailed information for each of the selected sessions is to be displayed. An alternate form of FULL is DETAIL.</p>
DETAIL	<p>Indicates that detailed information for each of the selected sessions is to be displayed. An alternate form of DETAIL is FULL.</p>
MSGID	<p>Indicates that nine-character message prefixes are displayed.</p> <p>For Telnet users, NOMSGID is the default and MSGID must be specified to enable the display of the message prefixes.</p> <p>For TSO users, MSGID is the default and NOMSGID must be specified to suppress the display of the message prefixes.</p> <p>For operator requests, the message prefixes are always displayed and this parameter has no effect</p>
NOMSGID	<p>Indicates that nine-character message prefixes are not displayed.</p> <p>For Telnet users, NOMSGID is the default and MSGID must be specified to enable the display of the message prefixes.</p> <p>For TSO users, MSGID is the default and NOMSGID must be specified to suppress the display of the message prefixes.</p> <p>For operator requests, the message prefixes are always displayed and this parameter has no effect</p>

## USER

Displays information about connected users.

```
NETSTAT USER [ SESSION( nnn ) ] [ IPNAME ] [ MSGID | NOMSGID ]
```

or

```
NETSTAT USER [ REMHOST( ip_addr ) ] [ LOCPORT( nnn ) ] [ IPNAME ]  
[ MSGID | NOMSGID ]
```

null	Lists all users.
SESSION( <i>nnn</i> )	Limits the display to a specified session.
REMHOST( <i>ip_addr</i> )	Limits the display to the entries associated with a specific host. The <i>ip_addr</i> can be specified as either dotted notation, or a symbolic string, and an ending asterisk (*) can be used as a wildcard indicator. The REMHOST( <i>ip_addr</i> ) and LOCPORT( <i>nnn</i> ) options can be used together or separately.
LOCPORT( <i>nnn</i> )	Limits the display to the entries associated with a specific port.
IPNAME	Attempts to resolve and display actual hostname instead of IP address.
MSGID   NOMSGID	Controls the display of the nine-character message prefix. <ul style="list-style-type: none"><li>■ For Telnet users, NOMSGID is the default; use MSGID to turn on the message display</li><li>■ For TSO users, MSGID is the default; use NOMSGID to omit the prefix from the display</li><li>■ For operator requests, the prefix is always displayed; this parameter has no effect</li></ul>

## XCF

Displays the status of XCF members.

```
NETSTAT XCF [ MSGID | NOMSGID ]
```

MSGID   NOMSGID	Controls the display of the nine-character message prefix. <ul style="list-style-type: none"><li>■ For Telnet users, NOMSGID is the default; use MSGID to turn on the message display</li><li>■ For TSO users, MSGID is the default; use NOMSGID to omit the prefix from the display</li><li>■ For operator requests, the prefix is always displayed; this parameter has no effect</li></ul>
-----------------	--

The following messages result from the NETSTAT XCF command:

T01NT110I    T01NT111I    T01NT112I    T01NT113I

## SYSSTAT Diagnostic Commands

SYSSTAT diagnostic commands include all of the NETSTAT commands, plus the commands CANC and CHANGE. They are described below.

### CANC

Cancels a specified session.

```
SYSSTAT CANC [ SESSION( nnn ) ] [ SSID( subsysname ) ] [ MSGID | NOMSGID ]
```

or

```
SYSSTAT CANC [ TCP | UDP | RAW ] REMHOST( ip_addr ) | LOCPORT( nnn )
[ MSGID | NOMSGID ]
```

SESSION( <i>nnn</i> )	Cancels a specific session number ( <i>nnn</i> ).
SSID( <i>subsysname</i> )	Specifies the subsystem ID.
TCP   UDP   RAW	If cancellation is not by specific session number, one of these protocol identifiers must be specified.
LOCPORT( <i>nnn</i> )	Cancels one or more sessions, using a local port number.  <i>nnn</i> represents a specific port number.
REMHOST( <i>ip_addr</i> )	Cancels a connection to a remote host.  <i>ip_addr</i> is the IP address of the host. The IP address must be entered in dotted notation.
MSGID   NOMSGID	Controls the display of the nine-character message prefix. <ul style="list-style-type: none"> <li>■ For Telnet users, NOMSGID is the default; use MSGID to turn on the message display</li> <li>■ For TSO users, MSGID is the default; use NOMSGID to omit the prefix from the display</li> <li>■ For operator requests, the prefix is always displayed; this parameter has no effect</li> </ul>

**WARNING!** Cancellation by a specific session number only cancels a single session. The other option, cancellation by protocol/IP/port, can terminate multiple sessions.

## CHANGE

Changes TCP variables.

```
SYSSTAT CHANGE [ CWIND( nnn ) SESSION( nnn ) ] [ THRS( nnn ) SESSION( nnn ) ]  
                [ RTT( nnn ) SESSION( nnn ) ] [ DEBUG( ON | OFF ) ] [ SESSION( nnn ) ]  
                [ MSGID | NOMSGID ]
```

CWIND( nnn )  
SESSION( nnn )

Changes the congestion window size for a specified session.

**Note:** Both variables are required.

THRS( nnn ) SESSION(  
nnn )

Changes the slow start threshold for a specified session.

**Note:** Both variables are required.

RTT( nnn ) SESSION(  
nnn )

Changes the round trip start time for a specified session.

**Note:** Both variables are required.

DEBUG( ON | OFF )

Used in TCP connections to produce debugging messages to the T01LOG data set.

In order to use this feature, the TCP task group must have the TEST flag set on. An operator command, or startup script, can set this flag, using the command /SET TEST ON TGB(TCP). Similarly the flag can be set off using the command /SET TEST OFF TGB(TCP). Once this flag is on, the individual DEBUG flags of each connection are queried.

Using this method to diagnose TCP problems is not recommended, as large volumes of messages may be produced. These messages are directed to the T01LOG data set. The messages produced are in the range of T01TC9xx.

MSGID | NOMSGID

Controls the display of the nine-character message prefix.

- For Telnet users, NOMSGID is the default; use MSGID to turn on the message display
- For TSO users, MSGID is the default; use NOMSGID to omit the prefix from the display
- For operator requests, the prefix is always displayed; this parameter has no effect

## OSPFMON

The OSPFMON (`ospf_monitor`) TSO diagnostic command can request information for routes known to OSPF.

**Note:** OSPFMON executes as a TSO command processor. All messages that are written are done with TPUTs, restricting its use to interactive TSO users only.

OSPFMON is intended for use as a tool for debugging gateways, not for network management. SNMP is the preferred network management protocol.

GateD responds only to OSPFMON requests issued on the same subnet on which GateD is running.

```
OSPFMON -x subsysid filename
```

*subsysid*

Four character MVS subsystem ID of the Unicenter TCPaccess job or started task.

Default: ACSS

*filename*

Name of a file that has a list of gateways.

### Example

The following is a sample of a list of gateways:

```
138.42.181.50 gateway1.company.com [password]  
138.42.171.64 gateway2.company.com  
138.42.181.200 gateway3.company.com  
138.42.171.220 gateway4.company.com  
138.42.181.3 gateway5.company.com  
138.42.224.4 gateway6.company.com
```

The optional *password* is specified by the `monitorauthkey` sub-parameter of the `ospf` parameter of the GateD configuration file.

## Local Commands

Enter **OSPFMON** for the desired subsystem ID.

The following commands are supported:

d	Shows configured destinations.
h	Shows history.
x	Exits.
@ <i>remote_command</i>	Uses last destination.
@ <i>dest_index remote_command</i>	Uses configured destination index.
F <i>filename</i>	Writes monitor information to filename.
S	Monitors information to stdout.

## Remote Commands

**a** *area\_id type ls\_id  
adv\_rtr*

Display link state advertisement. *Area\_id* is the OSPF area for which the query is directed. *adv\_rtr* is the router-id of the router which originated this link state advertisement. *Type* specifies the type of advertisement to request and should be specified as follows:

1. Request the router links advertisements. They describe the collected states of the router's interfaces. For this type of request, the *ls\_id* field should be set to the originating router's Router ID.
2. Request the network links advertisements. They describe the set of routers attached to the network. For this type of request, the *ls\_id* field should be set to the IP interface address of the network's Designated Router.
3. Request the summary link advertisements describing routes to networks. They describe inter-area routes, and enable the condensing of routing information at area borders. For this type of request, the *ls\_id* field should be set to the destination network's IP address.
4. Request the summary link advertisements describing routes to AS boundary routers. They describe inter-area routes, and enable the condensing of routing information at area borders. For this type of request, the *ls\_id* field should be set to the Router ID of the described AS boundary router.
5. Request the AS external link advertisements. They describe routes to destinations external to the Autonomous System. For this type of request, the *ls\_id* field should be set to the destination network's IP address.

- 
- c** Display cumulative log. This log includes input/output statistics for monitor request, hello, data base description, link-state request, link-state update, and link-state ack packets. Area statistics are provided which describe the total number of routing neighbors and number of active OSPF interfaces. Routing table statistics are summarized and reported as the number of intra-area routes, inter-area routes, and AS external data base entries.
- e** Display cumulative errors. This log reports the various error conditions that can occur between OSPF routing neighbors and shows the number of occurrences for each.
- h** Displays the next hop list. This is a list of valid next hops mostly derived from the SPF calculation.
- I** [*retrans*] Display the link-state database (except for ASE's). This table describes the routers and networks making up the AS. If *retrans* is non-zero, the retransmit list of neighbors held by this lsd structure will be printed.
- A** [*retrans*] Display the AS external data base entries. This table reports the advertising router, forwarding address, age, length, sequence number, type, and metric for each AS external route. If *retrans* is non-zero, the retransmit list of neighbors held by this lsd structure will be printed.
- o** [*which*] Display the OSPF routing table. This table reports the AS border routes, area border routes, summary AS border routes, networks, summary networks and AS external networks currently managed via OSPF. If *which* is omitted, all of the above will be listed. If specified, the value of *which* (between 1 and 63) specifies that only certain tables should be displayed. The appropriate value is determined by adding up the values for the desired tables from the following list:
- 1 Routes to AS border routers in this area.
  - 2 Routes to area border routers for this area.
  - 4 Summary routes to AS border routers in other areas.
  - 8 Routes to networks in this area.
  - 16 Summary routes to networks in other areas.
  - 32 AS routes to non-OSPF networks.
- I** Display all interfaces. This report shows all interfaces configured for OSPF. Information reported includes the area, interface IP address, interface type, interface state, cost, priority, and the IP address of the DR and BDR for the network.
- N** Display all OSPF routing neighbors. Information reported includes the area, local interface address, router ID, neighbor IP address, state, and mode.

## PING

Use the PING TSO diagnostic command to determine if a host is active on the network. PING sends an ICMP ECHO\_REQUEST packet to network hosts to elicit an ICMP ECHO\_RESPONSE from the specified host or network gateway. If the host responds, PING replies that the host is alive and then exits. Otherwise, after the timeout expires, PING replies that there was no answer from the host.

If a *count* is not specified, PING continues to try until it is stopped.

**Note:** PING requires the SAS/C Transient Library modules that are supplied in the LOAD/SASLOAD data set. This library is required in either your LOGON procedure or BATCH job STEPLIB DD.

Source for PING and many BSD socket applications can be obtained via anonymous FTP from Internet host gatekeeper.dec.com.

If, after invocation, PING waits too long for a reply, press PA1 to interrupt its execution (this may produce a dump of system related errors).

```
PING [ -dnqv ] [ -c count ] [ -i wait ] [ -l preload ] [ -p pattern ] [ -s  
packetsize ] [ -x subsysid ]  
      host
```

- d Sets the SO\_DEBUG socket option.
- n Shows network addresses as numbers (PING normally displays addresses as host names).
- q Runs PING in quiet mode.
- v Verbose output. Lists any ICMP packets received, other than ICMP\_RESPONSE.
- c *count* Specifies the number of requests to send.
- i *wait* Specifies the interval between successive transmissions.  
  
Default: One second.
- l *preload* Specifies the count of pings initially sent.
- p *pattern* Fills the buffer with *pattern*.
- s *packetsize* Specifies the packet size to send.  
  
Default: 64 bytes.

*-x subsysid* Specifies the subsystem ID of the Unicenter TCPaccess job or started task; *subsysid* is the four-character MVS subsystem ID.

Default: ACSS.

*host* Specifies the host name or IP address.

## RIPQUERY

The RIPQUERY TSO diagnostic command can request all routes known by a RIP gateway by sending a RIP request or POLL command. The routing information in any routing packets returned is displayed numerically and symbolically.

**Note:** The RIPQUERY tool executes as a TSO command processor. All messages are written with TPUTs, which restricts its use to interactive TSO users.

The RIPQUERY tool is for debugging gateways, not for network management. SNMP is the preferred network management protocol.

By default, RIPQUERY uses the RIP POLL command for this version of GateD. The RIP POLL command is preferable to the RIP REQUEST command as it is not subject to Split Horizon and/or Poisoned Reverse. See the RIP RFC for more information.

```
RIPQUERY -x subsysid [ -a password ] [ -d ] [ -n ] [ -p ] [ -r ] [ -v ] [ -1 ]
[ 2 ] [ -w time] routers ...
```

*-x subsysid* Four-character MVS subsystem ID of the Unicenter TCPaccess job or started task.

Default: ACSS.

*-a password* Authentication password to use for queries. If specified, an authentication type of SIMPLE is used. Otherwise, the default is an authentication type of NONE.

Authentication fields in incoming packets are displayed but not validated.

*-n* Prevents the address of the responding host from being looked up to find the symbolic name.

*-p* The RIP POLL command requests information from the routing table. This is the default for some versions of GateD.

If there is no response to RIP POLL, try the RIP REQUEST command. GateD responds to a POLL command with all the routes learned via RIP.

- r** The RIP REQUEST command requests information from the gateway's routing table. All gateways should support RIP REQUEST.
- If there is no response to the RIP REQUEST command, try the RIP POLL command. GateD responds to a REQUEST command with all routes currently announced on the specified interface.
- For systems based on BSD 4.3 Reno or earlier, responses to RIP REQUESTs contain information about the interface used to send the reply. To get information about a particular host, run RIPQUERY on that host.
- v** Displays version information about RIPQUERY before querying the gateways.
- 1** Sends the query as a version one packet.
- 2** Sends the query as a version two packet (default).
- w *time*** Specifies the time in seconds to wait for the initial response from a gateway.
- Default: Five seconds.
- routers*** Router name list.

## RPCINFO

The RCPINFO command queries an RPC server to obtain information about program names and program numbers for the specified host.

The RPCINFO command is supplied with Unicenter TCPaccess as a TSO command processor for use on the MVS system. Its usage is similar to the rpcinfo command on UNIX systems. No parameters are positional or case sensitive.

See *Unicenter TCPaccess Communications Server Unprefixed Messages and Codes* for information about RPCINFO error messages.

```
RPCINFO [ PROTO ( protocol ) ] [ HOST( host_name ) ] [ PROG( prog_num ) ]
        [ VERS( vers_num ) ] [ PORT( port ) ] [ SYSID( subsystem_id ) ]
```

PROTO( <i>protocol</i> )	Specifies the protocol to use, either UDP or TCP.
HOST( <i>host_name</i> )	Specifies the name of the host where the service is resident.
PROG( <i>prog_num</i> )	Specifies the RPC program number in decimal.
VERS( <i>vers_num</i> )	Specifies the version number of the RPC program.
PORT( <i>port_num</i> )	Specifies the UDP or TCP port where the selected service is registered.
SYSID( <i>subsystem_id</i> )	Specifies the subsystem ID of Unicenter TCPaccess.

Default: ACSS.

## Examples

Example 1 Use this RPCINFO command to dump the portmapper registration tables:

```
RPCINFO PROTO( protocol ) HOST( host_name ) PROG( prog_num )
        VERS( vers_num ) ] [ PORT( port_num ) ]
        SYSID( subsystem_id ) ]
```

Use this command to ping an RPC service:

```
RPCINFO HOST( host_name ) [ SYSID( subsystem_id ) ]
```

Use this command to delete an RPC service registration with the LOCAL portmapper:

```
RPCINFO DELETE PROG( prog_num ) VERS( vers_num ) SYSID( subsystem_id )
```

## Example 2

The following is a sample command and output for the RPC HOST command to list the services registered on MVS host zeus:

```
RPCINFO HOST(ZEUS)
RPC105I Thu Feb 07 10:47:10 1991 number vers. protocol port name
RPC106I Thu Feb 07 10:47:10 1991 100000 2 TCP 111 PORTMAPPER
RPC106I Thu Feb 07 10:47:10 1991 100000 2 UDP 111 PORTMAPPER
RPC106I Thu Feb 07 10:47:10 1991 100059 1 UDP 4098
RPC106I Thu Feb 07 10:47:10 1991 100044 1 UDP 4099
RPC106I Thu Feb 07 10:47:10 1991 100005 1 UDP 4100 MOUNTD
RPC106I Thu Feb 07 10:47:10 1991 150001 1 UDP 4101
RPC106I Thu Feb 07 10:47:10 1991 100003 2 UDP 2049 NFS
```

## STROBE

STROBE is a diagnostic tool to measure the amount of CPU time, by module. The tool is first activated using a STROBE ON command, time sampling is started, and at each time slice, the address portion of the interrupted PSW is saved. When the desired amount of samples is reached, or when the run is forced with the STROBE OFF command, module names are identified by matching the captured address list against module residence addresses.

The subsequent report is sorted by popularity, in decreasing order, and makes a distinction between TCB mode vs. SRB mode hits.

**Note:** The report is produced when the STROBE process ends (because the sample limit was reached, or because a STROBE OFF command is issued) and is sent to the T01LOG as diagnostic-level a message.

Syntax for the STROBE command:

```
STROBE ON|OFF|STATUS <SAMPLES(nnn) DETAIL(name)>
```

ON	Activates the sampling run. Sampling runs until the SAMPLES count is reached, or stopped with the OFF operand.
OFF	Used to terminate the run when desired.
STATUS	Used at anytime to sample progress.
SAMPLES( <i>nnn</i> )	Specifies the number of samples to capture. The upper limit is 1000000. May be aliased as SIZE.  Default: 500000.
DETAIL( <i>name</i> )	Used to focus on a specific module for more detailed information.

## TCPEEP

TCPEEP is a TSO command that invokes the Unicenter TCPAccess packet trace program to diagnose remote host communication problems. The TCPEEP realtime trace consists of selected network packet traffic to and from a local host. The TCPEEP command recognizes LNI level traffic and most of the IP-based higher-level protocols.

TCPEEP creates a NO WRAP Component Trace Instance and displays the output on a TSO terminal or directs it to a dynamically allocated SYSOUT data set. Optionally, it can stop any component trace instance, or modify an existing component trace instance or view an existing Component Trace Instance.

**Note:** TCPEEP runs only when the Unicenter TCPAccess and TRACE address spaces are active.

## User Interface

TCPEEP is run as a TSO command, either online or in batch.

The JCL to run TCPEEP as a batch job is in SAMP member TCPEEP. Unicenter TCPAccess TRACE must be up and running before submitting a batch job for TCPEEP.

The following is a sample JCL for running TCPEEP in batch.

```
//TCPEEP JOB (TCPEEP), 'TCPEEP', CLASS=A, MSGCLASS=X
/*
/* Sample JCL to run TCPEEP in batch.
/*
/* Update 'trgindx' to reflect your library naming convention.
/*
/* Note: The TCPAccess and TRACE address spaces must be
/*       running.
/*
//TCPEEP EXEC PGM=IKJEFT01, DYNAMNBR=50, REGION=4M
//STEPLIB DD DISP=SHR, DSN=trgindx.LINK
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*, LRECL=132, BLKSIZE=132, RECFM=FB
//SYSIN DD DUMMY
/*
//SYSTSIN DD *
TCPEEP TRCSSID(ACTR) GROUPS(NETIF) +
      BUFFTME(5)
/*
```

To stop the TCPEEP batch job, issue the MVS **STOP** command (for example, **P jobname**).

## Trace Data Collected

The trace facility collects the following type of trace data:

- From an application
- TCP/IP address space

Simultaneous tracing of various trace types can be fed into a single trace output.

**Note:** If data is collected via the external writer to an external data set, you must use the MVS TRACE command. See [External Writer](#) for more information.

## Viewing Trace Data

TCPEEP collects realtime data and by default writes to SYSTSPRT. Optionally, it can write to a dynamically allocated SYSOUT data set.

## Trace Operation

Each occurrence of the TCPEEP command varies according to the type of environment.

To stop a trace, press the terminal ATTENTION key and enter **H** at the prompt (null entry allows trace to continue).

**WARNING!** *TCPEEP should be installed in a protected library only. It can be used to display all network traffic through Unicenter TCPaccess, including user IDs and passwords.*

## TCPEEP Syntax

This section describes the TCPEEP syntax and describes its parameters.

```
TCPEEP [ ASID( asid,... ) ]
        [ BUFFERS ( size, number ) ]
        [ BUFTIME ( time_out ) ]
        [ DATASIZE( record_size ) ]
        [ DNRSSID ( ssid ) ]
        [ FORMAT( format_options ) ]
        [ FULL | SUMMARY ]
        [ GROUPS( ( group [, 'filter' ] ... ) ]
        [ HALT ]
        [ INSTANCE ( instance_ID ) ]
        [ JOBNAME ( jobname,... ) ]
        [ NOHEADER ]
        [ PEEK ( limit ) ]
        [ SYSOUT ( class ) ]
        [ TRACE SIZE ( num_records ) ]
        [ TRCSSID ( ssid ) ]
```

ASID ( *asid* ,... )

Specifies the address space identifiers (ASIDs) of address spaces used as a filter for tracing. Events in the ASIDs are recorded by the component trace.

The parameter contains a list of 0 to 16 hexadecimal ASIDs separated by commas.

An empty ASID list, ASID=(), turns off filtering by address spaces. In the ASID parameter, list all address spaces to trace. Address spaces for previous traces are not traced unless listed.

Default: None.

BUFFERS  
( *size*,*number* )

Specifies the size of the trace buffers in kilobytes or the number of buffers.

*size*                    A value between 64 and 1024 (Default: 256).

*num*                     A value between 2 and 128 (Default: four).

BUFFERS                (Optional). Can only be specified when creating a new trace instance. If specified when modifying a trace instance, it is ignored.

Range: *Size*: 64 - 1024  
       *Number*: 2 - 128.

Default: (256, 4).

**BUFFTIME** ( *time\_out* ) Specifies the buffer time out interval in seconds. At the end of each interval, if the current buffer contains data but is not full, a buffer flush operation is initiated.

BUFFTIME is optional and can only be specified when creating a trace instance. If specified when modifying a trace instance, it is ignored.

Use this parameter to force a buffer switch so you do not have to wait for the entire buffer to fill to see trace data.

Range: 0 – 99999.

Default: 10.

**DATASIZE** ( *record\_size* )

Specifies the maximum size of a trace record in kilobytes. Trace records that exceed the specified value are truncated.

DATASIZE – (Optional). Can only be specified when creating a trace instance. If specified when modifying a trace instance, it is ignored.

If the specified maximum size exceeds the largest supported trace record size (64 KB less control headers), then the specification has no effect.

Default: None.

**DNRSSID** ( *ssid* ) Specifies the subsystem ID of the address space to use for name resolution by the GROUPS filtering process.

Specification is ignored if GROUPS with filter parameters is not specified.

Default: ACSS.

**FORMAT** ( *format\_options* )

The format of the collected data.

Formatted data can be specified as follows:

APCB	Specifies the APCB should be formatted.
APCBHEX	Dump the APCB in hex.
ARP	Selects ARP level trace.
ARPHex	Selects ARP level trace but dumps the ARP header in hex.

---

ASCII( <i>len</i> )	Same as DATA( <i>len</i> ) but uses an ASCII only translate table.
DATA( <i>len</i> )	Displays the actual data transmitted.  <i>len</i> – (Optional). The default is to display all the data. For example, FORMAT( DATA ). Specifying FORMAT( DATA ( 16 ) ) displays at most 16 (X'10') bytes of data. This uses a mixed ASCII and EBCDIC translate table.
DEVICE	Displays device/interface name.
EBCDIC( <i>len</i> )	Same as DATA( <i>len</i> ) but uses an EBCDIC only translate table.
EZADATA   EDATA   EZAEBDIC   EZAASCII or EZADATA ( <i>nn</i> )   EDATA ( <i>nn</i> )   EZAEBDIC ( <i>nn</i> )   EZAASCII ( <i>nn</i> )	Displays details about EZASOCKET / EZACICAL socket API parameters and optionally API data for Unicenter Netmaster Socket Management and CPT endpoints.  EZADATA   EDATA uses a mixed ASCII and EBCDIC translate table  EZAEBDIC   EEBDIC uses an EBCDIC-only translate table  EZAASCII   EASCII uses an ASCII-only translate table  Range: 0 - 65535  Default: 65535
IP	Selects IP level trace.
IPHEX	Selects IP level trace but dumps the IP header in hex.
ICMP	Selects ICMP level trace.
ICMPHEX	Selects ICMP level trace but dumps the ICMP header in hex.
LNI	Selects the LNI trace (all LNI traffic is traced).

OEDATA | OEEBCDIC | OEASCII

or

OEDATA(*nn*) | OEEBCDIC(*nn*) | OEASCII(*nn*)

Displays details about Open Edition (OE)/Unix System Services (USS) socket API parameters and optionally API data

- *nn* is between 0 and 65535 (default is 65535)
- OEDATA uses a mixed ASCII and EBCDIC translate table
- OEEBCDIC uses an EBCDIC-only translate table
- OEASCII uses an ASCII-only translate table

TCP	Selects TCP level trace.
TCPHEX	Selects TCP level trace but dumps the TCP header in hex.
DEVICE	Displays device/interface name.
EBCDIC( <i>len</i> )	Same as DATA( <i>len</i> ) but uses an EBCDIC only translate table.
IP	Selects IP level trace.
IPHEX	Selects IP level trace but dumps the IP header in hex.
ICMP	Selects ICMP level trace.
ICMPHEX	Selects ICMP level trace but dumps the ICMP header in hex.
LNI	Selects the LNI trace (all LNI traffic is traced).

TLIDATA | TLIEBCDIC | TLIASCII

or

TLIDATA(*nn*) | TLIEBCDIC(*nn*) | TLIASCII(*nn*)

Displays details about Transport Layer Interface (TLI) socket API parameters and optionally API data

- *nn* is between 0 and 65535
- TLIDATA uses a mixed ASCII and EBCDIC translate table
- TLIEBCDIC uses an EBCDIC-only translate table
- TLIASCII uses an ASCII-only translate table

TNDATA | TNEBCDIC | TNASCII

or

TNDATA(*nn*) | TNEBCDIC(*nn*) | TNASCII(*nn*)

Displays details about TN3270 parameters and optionally data

- *nn* is between 0 and 65535 displays data captured for the trace group TELNET
- TNDATA uses an ASCII translate table for NVT mode and EBCDIC otherwise
- TNEBCDIC uses an EBCDIC-only translate table
- TNASCII uses an ASCII-only translate table

**Note:** TNDATA is recommended, since it translates data according to the session's current negotiated telnet protocol.

TPL	Specifies the TPL should be formatted.
TPLHEX	Dump the TPL in hex.
UDP	Selects UDP level trace.
UDPHEX	Selects UDP level trace but dumps the UDP header in hex.

**Format Options**

NETIF: ARP, ARPHex, DATA(*nn*) | EBCDIC(*nn*) | ASCII(*nn*), DEVICE, IP, IPHEX, LNI, TCP, TCPHEX, UDP, and UDPHEX.

Defaults: ARP, IP, ICMP, LNI, TCP, UDP

TLI: APCB, APCBHEX, TPL, and TPLHEX.

Defaults: APCB and TPL.

IUCV Displays details about (HPNS) High Performance Native Sockets or Inter-User Communications Vehicle(IUCV) socket API parameters and optionally API data

Use one of the following options to format the data collected using MAXIUCVDATA:

- IUCVDATA (alias IDATA) – Uses a mixed ASCII and EBCDIC translate table
- IUCVEBCDIC (alias IEBCDIC) – Uses an EBCDIC-only translate table
- IUCVASCII (alias IASCII) – Uses an ASCII-only translate table

If specified as is, then all of the data is displayed. Each takes an optional *nn* data amount; for example, IDATA (32) displays up to 32 bytes of data.

FULL | SUMMARY Show all or a summary of the data.

Default: SUMMARY.

GROUPS( (*group* [, '*filter*' ] ) ... )

Trace group or groups for which data is collected and optionally a filter parameter for each group.

Use the GROUPS parameter to limit the amount of data collected. Used with INSTANCE to modify an existing trace instance, either adds a new group to the trace instance or replaces an existing group for the trace instance. Once added, a group cannot be removed from the active trace instance.

---

You can specify a maximum of four trace groups.

*group*

Selects the type of data to collect: EZA, IUCV, NETIF, TLI, OE, RTM, or TELNET.

- The EZA is a collection of trace points in Socket Management for processing EZASOCKET API activity with the following filter options:

HOST(host,...,host) – Up to 16 IP HOST addresses (names)

PORT(port,...,port) – Up to 16 port numbers (names)

USER(jobname,...,jobname) – Takes 1 to 16 *jobnames* that refer to jobs using the EZA function

UASID(asid,...,asid) – Takes 1 to 16 *asids* that refer to jobs using the EZA function

MAXEZADATA(*nnnn*) | MEZADATA(*nnnn*) | MEDATA(*nnnn*) – Where *nnnn* is a positive integer less than or equal to 65535

**Note:** By default, no user data is traced, so MAXEZADATA causes the data to be collected.

- The IUCV keyword traces both IUCV and HPNS applications. IUCV can be specified with the following *filter* options:

HOST(*host,...,host*) – Up to 16 IP HOST addresses (names)

PORT(*port,...,port*) – Up to 16 port numbers (names)

USER(*jobname,...,jobname*) – Takes 1 to 16 *jobnames* that refer to jobs using the TLI function

Specify IUCVAPP for IUCV or HPNSAPP for HPNS.

UASID(*asid,...,asid*) – Takes 1 to 16 *asids* that refer to jobs using the TLI function

MAXIUCVDATA(*nnnn*) – Where *nnnn* is a positive integer less than or equal to 65535 (alias MIDATA)

**Note:** By default, no user data is traced, so MAXIUCVDATA causes the data to be collected

- NETIF – A network interface that includes the following filter options:

HOST(*host,...,host*) – Up to 16 IP HOST addresses (names)

PORT(*port,...,port*) – Up to 16 port numbers (names)

PROTOCOL – A single protocol (for example, UDP)

TYPE(*type*) for NETIF can be one of the following:

- TYPE(IP) – if datagram is an IP packet, then keep it
- TYPE(ARP) – if datagram is an ARP packet, then keep it
- TYPE(ALL) – keep all datagrams (same as not specifying the filter)

MAXDATA(*nn*) – Alias MDATA. *nn* is between 1 and 65535. By default, the first 256 bytes of each packet are captured (to include the IP, TCP, and so on headers).

---

**Note:** Any value between 1 and 255 is increased to 256. There are no defaults for filters; everything is collected by default.

- OE (UNIX System Services)

HOST(*host*,...,*host*)

PORT(*port*,...,*port*)

USER(*jobname*,...,*jobname*)

UASID(*asid*,...,*asid*)

MAXOEDATA(*nn*). *nn* is between 1 and 65535. By default, no user data is captured.

INODE(*inode1*,...,*inode16*) can be specified.

**Note:** This requires you to know the inode in advance. It is intended more for post processing captured data (not for TCPEEP) under IPCS.

- TELNET – A collection of trace points in the TN3270E server to trace activity to/from the client and the VTAM application.

HOST(*host*,...,*host*) – Up to 16 IP HOST addresses (names).

PORT(*port*,...,*port*) – Up to 16 port numbers (names).

LU(*luname*,...,*luname*) – Up to 16 LU names. These are either VTAM Applids or SLU names. This filter is restrictive in that, if specified, no tracing should be collected on a connection until either an SLU or APPLID is determined.

MAXTNDATA(*nn*). *nn* is between 1 and 65535. By default, no data is captured. (Alias is MTNDATA).

Refer to Useful TCPEEP Commands for usage of the GROUPS parameter.

- RTM— A collection of trace points in TelnetRTM that traces RTM activity. The following filter options are supported:
  - UASID(*asid*,...,*asid*)— Takes 1 to 16 *asids*, which refer to jobs using TelnetRTM (Unicenter NetSpy or a Unicenter TCPaccess Telnet server).
  - USER(*jobname*,...,*jobname*)— Takes 1 to 16 jobnames, which refer to jobs using TelnetRTM (Unicenter NetSpy or a Unicenter TCPaccess Telnet server).

Default: NETIF.

- TLI— A collection of trace points in the stack for processing TLI activity with the following filter options:

HOST(*host*,...,*host*)— Up to 16 IP HOST addresses (names)

PORT(*port*,...,*port*)— Up to 16 port numbers (names)

UASID(*asid*,...,*asid*)— Takes one to 16 *asids*, which refer to jobs using the TLI function

USER(*jobname*,...,*jobname*)— Takes 1 to 16 *jobnames* that refer to jobs using the TLI function

MAXTLIDATA(*nn*)— Alias MTLIDATA or MTDATA. *nn* is between 1 and 65535.

By default, no user data is captured.

EPID(*epid1*,...,*epid16*) may be specified. As this requires one to know the *epid* in advance, this is intended more for post-processing captured data (not for TCPEEP) under IPCS.

---

HALT	Stops a component trace instance.
INSTANCE ( <i>instance_id</i> )	Select a trace instance to display by specifying the <i>instance_id</i> returned from TCPEEP or the MVS TRACE CT command.
JOBNAME ( <i>jobname ,....</i> )	Names of jobs used as filters for tracing. Events in these jobs are recorded by the component trace.
NOHEADER	Do not display the header information from the trace entry.  This option helps limit the output. Without it, one line prints describing the trace entry even if no other information about that entry displays.
PEEK ( <i>limit</i> )	Number of trace records to view.  Use to view an existing trace. PEEK or PEEK(0) implies no limit; PEEK( <i>n</i> ) traces only <i>n</i> trace records.  Range: Zero - no limit.  Default: Zero.
SYSOUT ( <i>class</i> )	Sends output to a dynamically allocated SYSOUT data set.  By default, the output for TCPEEP writes to SYSTSPRT.  Default: X.
TRACESIZE ( <i>num_records</i> )	Maximum number of trace records to be recorded. If not specified, there is no limit to the number of records recorded.  TRACESIZE is optional and can only be specified when creating a trace instance. If specified when modifying a trace instance, it is ignored.
TRCSSID ( <i>ssid</i> )	ID of the trace address space.  Default: ACTR.

## Useful TCPEEP Commands

This section lists the keywords for each function listed.

Creating a NOWRAP Trace

Use the following command to initiate a NO WRAP trace instance:

```
TCPEEP TRCSSID( trcsubsysid ) DNRSSID( dnrsubsysid ) GROUPS( group )      +
      JOBNAME( jobname ) ASID( asid ) DATASIZE( size ) TRACESIZE( trcsize )  +
      BUFFERS( bufname ) BUFTIME( buftime ) FORMAT( format_options )      +
      SYSOUT( class )
```

Stopping Trace

Use this command to stop any trace instance:

```
TCPEEP INSTANCE( inst ) HALT TRCSSID( trcsubsysid )
```

Modify Existing Trace

Use this command to modify an existing trace instance:

```
TCPEEP INSTANCE( inst ) TRCSSID( trcsubsysid ) DNRSSID( dnrsubsysid )      +
      GROUPS ( group ) JOBNAME( jobname ) ASID( asid )      PEEK          +
      FORMAT( format_options ) SYSOUT( class )
```

Regardless of how the trace is started view its LIFO for the specified number of records (PEEK or PEEK(0) implies no limit).

View Existing Trace

Use this command to view an existing trace instance:

```
TCPEEP INSTANCE( inst ) TRCSSID( trcsubsysid PEEK          +
      FORMAT( format_options ) SYSOUT
```

Regardless of how the trace is started view its LIFO for the specified number of records (PEEK or PEEK(0) implies no limit).

Specifying GROUPS

This example shows how to specify the GROUPS for TCPEEP.

```
TCPEEP GROUPS ( ( NETIF, 'HOST ( host1, host2 ), PORT ( ECHO ) ' ),      +
      ( NETIF, 'HOST ( host3 ) ' ) )
```

The following example allows you to specify NETIF without additional filtering.

```
TCPEEP GROUPS ( NETIF )
```

To specify NETIF and PROTOCOL(UDP):

```
TCPEEP GROUPS ( ( NETIF, 'PROTOCOL ( UDP )' ) )
```

The following specifies a NETIF and an IUCV group. You may use the IUCV keyword to specify an IUCV or HPNS application for tracing.

```
TCPEEP TRCSSID( ACTR ) FORMAT ( TCP, DATA (16), MIDATA (16) )
GROUPS (( NETIF, 'PROTOCOL (TCP), PORT (3722) ' ), ( IUCV, 'USER ( IUCVAPP ) ' ) )
```

**Note:** IUCVAPP can be replaced with HPNSAPP.

## TCPEEP Examples

This section shows the output from several TCPEEP commands.

### Default TCPEEP Command

This sample shows the output for a TCPEEP command with no filtering.

```

3172  00010100 03/26 12:47:13.246650 INPUT
3172 Control:Offset=0, Next=20, Data Len=20, Adapter=0
      CMD= Shutdown_ Flag= ICP-Init
-----
3172  00010200 03/26 12:47:13.247232 OUTPUT
FDDI:Offset=0, Next=73, Data Len=73, Adapter=2
      Destination=FF.FF.FF.FF.FF.FF, Source=52.34.56.78.9A.BC
      ProtType=0800 (IP)
IP:Src=192.120.80.9, Dst=9.12.120.110, TotLen=48, Prot=6, Id=38827
  Vers=4, Hdr Len=20, TOS=0, TTL=60, Offset=0(0), Cksum=68C0
TCP:12:47:13.247232 192.120.80.9:1234 --> 9.12.120.110:9
  Seq:1,9(8) Ack:2 Psh
  HdrLen=20, Win=2048(2050), Cksum=9BAC, Flags=08
-----
3172  00010100 03/26 12:47:13.247546 INPUT
FDDI:Offset=0, Next=61, Data Len=61, Adapter=2
      Destination=FF.FF.FF.FF.FF.FF, Source=52.34.56.78.9A.BC
      ProtType=0800 (IP)
IP:Src=192.120.80.8, Dst=9.12.120.110, TotLen=36, Prot=17, Id=38827
  Vers=4, Hdr Len=20, TOS=0, TTL=60, Offset=0(0), Cksum=68C0
UDP:Src Port=5678, Dst Port=9, Len=16, Cksum=F35A
-----
3172  00010100 03/26 12:47:13.247706 INPUT
FDDI:Offset=0, Next=61, Data Len=61, Adapter=2
      Destination=FF.FF.FF.FF.FF.FF, Source=52.34.56.78.9A.BC
      ProtType=0800 (IP)
IP:Src=192.120.80.8, Dst=9.12.120.110, TotLen=36, Prot=1, Id=20000
  Vers=4, Hdr Len=20, TOS=0, TTL=60, Offset=0(0), Cksum=68C0
ICMP:Echo Response, Cksum=5D52
  Id=04D2, Sequence Number=5678
-----
3172  00010100 03/26 12:47:13.248779 INPUT
Token Ring:Offset=0, Next=316, Data Len=316, Adapter=1
  Access Control=10, Frame Control=40
  Destination=FF.FF.FF.FF.FF.FF, Source=92.34.56.78.9A.BC
  Control Byte 1=06, Control Byte 2=B0
  Segments:1234 5678
  Dsap=01, Ssap=02, Ctl=03
  ProtType=0800 (IP)
IP:Src=192.120.80.9, Dst=9.12.120.110, TotLen=281, Prot=6, Id=38827
  Vers=4, Hdr Len=20, TOS=0, TTL=60, Offset=0(0), Cksum=68C0
TCP:12:47:13.248779 9.12.120.110:9 <-- 192.120.80.9:1234
  Ack:2 Seq:1,242(241) Urg Psh
  HdrLen=20, Win=2048(2050), UrgPtr=2(3), Cksum=9BAC, Flags=28
-----

```

## LNI Level Trace

This example shows the LNI level trace output.

```
3172      00010100 03/26 13:00:53.774527 INPUT
3172 Control:Offset=0, Next=20, Data Len=20, Adapter=0
      CMD= Shutdown_ Flag= ICP-Init
-----
3172      00010200 03/26 13:00:53.774881 OUTPUT
FDDI:Offset=0, Next=73, Data Len=73, Adapter=2
      Destination=FF.FF.FF.FF.FF.FF, Source=52.34.56.78.9A.BC
      ProtType=0800 (IP)
-----
3172      00010100 03/26 13:00:53.775035 INPUT
FDDI:Offset=0, Next=61, Data Len=61, Adapter=2
      Destination=FF.FF.FF.FF.FF.FF, Source=52.34.56.78.9A.BC_
      ProtType=0800 (IP)
-----
3172      00010100 03/26 13:00:53.775967 INPUT
Token Ring:Offset=0, Next=316, Data Len=316, Adapter=1
      Access Control=10, Frame Control=40
      Destination=FF.FF.FF.FF.FF.FF, Source=92.34.56.78.9A.BC_
      Control Byte 1=06, Control Byte 2=B0
      Segments:1234 5678
      Dsap=01, Ssap=02, Ctl=03
      ProtType=0800 (IP)
-----
CETI      00010103 03/26 13:00:53.778563 INPUT
CETI Ethernet:Flags=00, Data Len=50
      Destination=FF.FF.FF.FF.FF.FF, Source=52.34.56.78.9A.BC
      ProtType=0800 (IP)
-----
```

## DATA Output

This sample shows the output for viewing the data.

```

3172    00010200 03/26 13:03:50.826067 OUTPUT
FDDI:Offset=0, Next=73, Data Len=73, Adapter=2
  Destination=FF.FF.FF.FF.FF.FF, Source=52.34.56.78.9A.BC
  ProtType=0800 (IP)
IP:Src=192.120.80.9, Dst=9.12.120.110, TotLen=48, Prot=6, Id=38827
  Vers=4, Hdr Len=20, TOS=0, TTL=60, Offset=0(0), Cksum=68C0
TCP:13:03:50.826067 192.120.80.9:1234 --> 9.12.120.110:9
  Seq:1,9(8) Ack:2 Psh
  HdrLen=20, Win=2048(2050), Cksum=9BAC, Flags=08
DATA +0000 E3C3D740 C4C1E3C1 *TCP@DATA *
-----
3172    00010100 03/26 13:03:50.826222 INPUT
FDDI:Offset=0, Next=61, Data Len=61, Adapter=2
  Destination=FF.FF.FF.FF.FF.FF, Source=52.34.56.78.9A.BC
  ProtType=0800 (IP)
IP:Src=192.120.80.8, Dst=9.12.120.110, TotLen=36, Prot=17, Id=38827
  Vers=4, Hdr Len=20, TOS=0, TTL=60, Offset=0(0), Cksum=68C0
UDP:Src Port=5678, Dst Port=9, Len=16, Cksum=F35A
DATA +0000 55445020 64617461 *UDP data *
-----
3172    00010100 03/26 13:03:50.827670 INPUT
Token Ring:Offset=0, Next=316, Data Len=316, Adapter=1
  Access Control=10, Frame Control=40
  Destination=FF.FF.FF.FF.FF.FF, Source=92.34.56.78.9A.BC
  Control Byte 1=06, Control Byte 2=B0
  Segments:1234 5678
  Dsap=01, Ssap=02, Ctl=03
  ProtType=0800 (IP)
IP:Src=192.120.80.9, Dst=9.12.120.110, TotLen=281, Prot=6, Id=38827
  Vers=4, Hdr Len=20, TOS=0, TTL=60, Offset=0(0), Cksum=68C0
TCP:13:03:50.827670 9.12.120.110:9 <-- 192.120.80.9:1234
  Ack:2 Seq:1,242(241) Urg Psh
  HdrLen=20, Win=2048(2050), UrgPtr=2(3), Cksum=9BAC, Flags=28
DATA +0000 54686973 20697320 41534349 49203031 *This is ASCII 01*
      +0010 00000000 00000000 00000000 00000000 *.....*
      +0020 00000000 00000000 00000000 00000000 *.....*
      +0030 00000000 00000000 00000000 00000000 *.....*
      +0040 00000000 00000000 00000000 00000000 *.....*
      +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 *.....*
      +00A0 00000000 00000000 00000000 00000000 *.....*
      +00B0 00000000 00000000 00000000 00000000 *.....*
      +00C0 00000000 00000000 00000000 00000000 *.....*
      +00D0 00000000 00000000 00000000 00000000 *.....*
      +00E0 00E38889 A24089A2 40C5C2C3 C4C9C340 *.This@is@EBCDIC@*
      +00F0 F0 *0 *
-----

```

## ICMP Format

This sample shows the output formatted for ICMP filtering.

```
3172      00010100 03/26 13:06:16.977131 INPUT
ICMP:Destination Unreachable,_ Cksum=5D52
      Code=1 (Hos
      )
      IP:Source=192.120.99.4, Dest=9.12.99.5, Proto=6, Tot Len=200
      Vers=4, Hdr Len=36, TOS=0, TTL=60, Id=1234, Offset=0, Cksum=68C0
-----
3172_____ 00010100 03/26 13:06:16.978076 INPUT
ICMP:Source Quench, Cksum=5D52
      IP:Source=192.120.99.4, Dest=9.12.99.5, Proto=6, Tot Len=200
      Vers=4, Hdr Len=36, TOS=0, TTL=60, Id=1234, Offset=0, Cksum=68C0
-----
```

## TRACE

Unicenter TCPaccess contains enhancements to the IBM MVS Component Trace facility to provide a method of collecting TCP/IP data and displaying it on a terminal or sending it to an external writer. Additional JCL is required to enable the Component Trace enhancements. To use Component Trace as a TSO command, see [TCPEEP](#).

## MVS Component Trace

MVS Component Trace is a diagnostic aid used to trace the action of certain system components and third party components that define themselves to Component Trace. The TRACE operator command is used to start, stop and control the component trace. For more information on the MVS TRACE command, see the IBM MVS System Commands.

## Trace Address Space

Component Trace is defined in its own address space and collects trace data for trace points defined in other address spaces. Likewise, the other address spaces must identify the Component Trace address space for the data collection. Multiple occurrences of Component Trace can be active at the same time, each with a unique subsystem ID.

For each address space, IJTFCGxx in the PARM member defines the subsystem ID. The definition is specified using the TRACENAME keyword on the IFSPARM statement as follows:

```
IFSPARM PROMPT VMCFNAME( VMCF ) TRACENAME( ACTR ) NOPROMPT
```

In the example, the trace data in the address space is directed to the Component Trace address space with a ACTR subsystem ID.

**Note:** If the Trace Address Space is brought up after the TCP stack, there may be a delay of two minutes before events are traced.

## Exit

MVS Component Trace requires an exit to communicate with the tracing component. The T03PTRSS exit must reside in LPALIB or the Link List. It is distributed in the LINK library.

## External Writer

The collected trace data is written to DASD or TAPE using an external writer. A suitable External Writer Cataloged Procedure for use with Component Trace may already be defined on your system. For more information on defining an External Writer used with Component Trace, see the IBM publication *MVS Diagnosis: Tools and Service Aids*.

## Component Trace JCL

This JCL allows you to run a Component Trace Address Space.

**WARNING!** Be sure to specify *P=T03* (shown in bold below) on the TRACE statement to specify the Component Trace Address space. If you specify T01 you will bring up a second Unicenter TCPAccess stack.

```
//RUNTRACE JOB
//*
//*   SAMPLE JCL PROCEDURE TO RUN TCPACCESS'S TRACE ADDRESS SPACE
//*   NOTE:  THIS ADDRESS SPACE SHOULD NOT BE TERMINATED
//*
//*   EDIT THE TRGINDX, SSN, SOUT, CMND SYMBOLIC
//*   PARAMETERS
//*
//*   VERIFY THAT THE JOB CARD AND NAMING CONVENTIONS MEET
//*   YOUR SITE'S JCL REQUIREMENTS, THEN SUBMIT THIS JOB.
//*
//ICSTRACE PROC TRGINDX='TRGINDX', TARGET LIBRARIES DSN INDEX
//          SSN=ACTR,           DFLT SUBSYSTEM NAME
//          SOUT='*',           CHOOSE A HOLD NONPURGE SYSOUT CLASS
//          CMND=STARTTR,       DFLT STARTUP COMMAND SCRIPT NAME
//          CNFG=00             IJTFCGXX SUFFIX
//*
//TRACE     EXEC PGM=IFSSTART,REGION=6144K,TIME=1440,
// PARM='IFSINIT,U=&SSN,P=T03,SO=&SOUT,CM=&CMND,CF=&CNFG'
//*
//STEPLIB DD DISP=SHR,DSN=&TRGINDX..LOAD
//          DD DISP=SHR,DSN=&TRGINDX..SASLINK
//*
//* WARNING: THE LOAD DATA SET MUST NEVER BE ADDED TO THE LINK LIST.
//*          IOS/390'S ELEMENT NAMES ARE NOT UNIQUE AND MAY
//*          AFFECT THE OPERATIONS OF OTHER SOFTWARE.
//*          THE LIBRARY SHOULD ALWAYS BE REFERENCED THROUGH
//*          A STEPLIB OR JOBLIB STATEMENT.
```

```

/**
/**      CONFIGURATION DATA SETS
/**
//SYSPARM DD DISP=SHR,DSN=&TRGINDX..PARM
//SYSPROC DD DISP=SHR,DSN=&TRGINDX..PARM
//SYSHELP DD DISP=SHR,DSN=&TRGINDX..HELP
/**
/**      LOG DATA SETS
/**
//T01LOG DD SYSOUT=&SOUT
//SYSPRINT DD SYSOUT=&SOUT
/**
/**      DUMP DATA SETS
/**
//SYSUDUMP DD SYSOUT=&SOUT
/**
/**      MISC DATA SETS
/**
//ABNLIGNR DD DUMMY          /* DISABLE ABEND-AID PROCESSING */
/**

```

## TRACE Command

The TRACE command starts, stops, modifies, or displays the status of a system trace, master trace, or component trace.

The TRACE command is a standard MVS operator command.

```

TRACE    [ CT { [ , ON | OFF ] [ , COMP=name ] [ , PARM=mem ] } ]
          [ CT { [ , WTRSTART=mem_name[, WRAP | NOWRAP ] ] } ]
          [ CT { , WTRSTOP=job_name } ] ]

```

- ON** Turns on tracing for a component if the component trace is currently off. If the component trace is on and can be changed, this changes the trace options.
- OFF** Turns off tracing for the component. If the component is connected to an external writer, the trace is implicitly disconnected from the writer.
- COMP=*name*** Identifies the component trace with the subsystem ID for the trace address space. This is required for each TRACE command.
- PARM=*mem*** Identifies a member of SYS1.PARM or a data set in the system parmlib concatenation containing the parameters used for tracing. Using a parmlib member enables the operator to initiate the trace, change it, or stop it without a message prompting for parameters.

Parameters specified on the TRACE command override the options specified in the parmlib member. The parameters are described in the section [TRACE Command Reply](#).

---

WTRSTART= *mem\_name*

Identifies the member containing the JCL to invoke an external writer and opens the data sets used by the external writer. The member must be a SYS1.PROCLIB cataloged procedure or a job.

After starting the external writer, use the WTR parameter to connect the component trace to the external writer.

WRAP | NOWRAP

NOWRAP instructs the system to stop writing data to a data set when the data set is full. With the WRAP parameter, when the data set or group of data sets is full, new data overwrites the oldest data at the start of the data set or the start of the first data set.

If the WTRSTART parameter on the TRACE CT command specifies NOWRAP, the system uses the primary and secondary extents of the data set or sets. If the WTRSTART parameter specifies WRAP or omits the parameter, the system uses only the primary extent or extents.

WTRSTOP=*job\_name*

Disconnects the external writer from the component trace and closes the data sets used by the external writer.

*jobname* is the member name if the source JCL is a procedure, or a job name if defined on a JOB statement within the source JCL.

Before stopping the external writer, turn the component trace off with TRACE CT,OFF or disconnect the external writer with WTR=DISCONNECT.

## TRACE Command Reply

In response to a TRACE CT,ON command without the PARM parameter, the system prompts you to specify the component trace options. Use the REPLY command to respond.

```
R id[ ,ASID=( nnnn [ ,nnnn ]... ) ]  
  [ ,CONT | ,END ]  
  [ ,JOBNAME=( name [ ,name ]... ) ]  
  [ ,OPTIONS=( option [ ,option ]... ) ]  
  [ ,WTR={ mem_name | DISCONNECT } ]
```

*id* Use the same identification number (0-9999) from the message to identify the reply.

ASID=( nnnn [ , nnnn ] ... )

Specifies the address space identifiers, ASIDs of address spaces used as a filter for tracing. Events in the ASIDs are recorded by the component trace.

The parameter contains a list of 0 to 16 hexadecimal ASIDs separated by commas. An empty ASID list, ASID=(), turns off filtering by address spaces.

In the ASID parameter, list all address spaces to be traced. Address spaces for previous traces are not traced unless listed.

CONT or END

The CONT parameter continues the reply on another line. The system issues another reply message. You can then continue the reply and repeat any parameters on the continuation line, except END. Repeated parameters are strung together, they do not overlay each other. The END parameter identifies the end of the REPLY.

CONT or END must be the last parameter on the input line.

JOBNAME=( name [ , name ] ... )

Names of jobs used as filters for tracing. Events in these jobs are recorded by the component trace.

The parameter contains a list of 0 to 16 job names separated by commas.

An empty job list, JOBNAME=(), turns off filtering by jobs.

In the JOBNAME parameter, list all jobs to be traced. Jobs specified for previous traces are not traced unless listed.

OPTIONS=( option [ , option ] ... )

Specifies the component trace options described in [TRACE Command Reply Options](#).

WTR= *mem\_name* | DISCONNECT

The *membername* identifies the member containing the source JCL that invokes the external writer. The member must be a SYS1.PROCLIB cataloged procedure or a job. The membername in the WTR parameter must match the membername in the TRACE CT,WTRSTART command.

WTR=DISCONNECT disconnects the external writer. The component continues tracing and placing the trace records in the address-space buffer, but stops passing trace records to the external writer.

You must also specify a TRACE CT,WTRSTART or TRACE CT,WTRSTOP command to start or stop the writer.

## TRACE Command Reply Options

Use the Options parameter in response to the Reply prompt.

```
OPTIONS= [ BUFFERS( size,num ) ] [ BUFTIME( time ) ] [ DATASIZE( dsize ) ]
          [ DNRSSID( ssid ) ] [ GROUPS( ( group, 'filter' ) ... ) ] [ HALT ]
          [ INSTANCE( inst ) ] [ STATUS ] [ TRACESIZE( tsize ) ] [ WRAP ]
          [ NOWRAP ]
```

**BUFFERS** ( *size,num* ) Size of the trace buffers in kilobytes or the number of buffers.

*size* A value between 64 and 1024.

Default: 256.

*num* A value between 2 and 128.

Default: Four.

BUFFERS can only be specified for a new trace instance. BUFFERS is optional and should only be specified when creating a trace instance.

**Note:** If specified when modifying a trace instance, it is ignored.

**BUFTIME**( *time* ) Buffer timeout interval in seconds. At the end of each interval, if the current buffer contains data but is not full, a buffer flush operation is initiated.

BUFTIME is optional and is considered only when creating a trace instance.

Use this parameter to force a buffer switch so you do not have to wait for the entire buffer to fill to see trace data.

Range: 0 - 99999.

Default: 10.

**DATASIZE**( *dsize* ) Specifies the maximum size of a trace record in kilobytes. Trace records that exceed the specified value are truncated.

DATASIZE is optional and can only be specified when creating a trace instance. If specified when modifying a trace instance, it is ignored. If the specified maximum size exceeds the largest supported trace record size (64 KB less control headers), then the specification has no effect.

Default: Any size record is recorded (up to the maximum IBM limit of 64 KB less control headers.)

**DNRSSID**( *ssid* ) Specifies the subsystem ID of the address space to use for name resolution by the GROUPS filtering process.

Specification is ignored if GROUPS with filter parameters is not specified.

Default: ACSS.

**GROUPS**( ( *group* [ , '*filter*' ] ) ... )

Trace group or groups for which data is collected and optionally a filter parameter for each group.

Use the GROUPS parameter to limit the amount of data collected. Used with INSTANCE to modify an existing trace instance, either adds a new group to the trace instance or replaces an existing group for the trace instance. Once added, a group cannot be removed from the active trace instance.

You can specify a maximum of four trace groups.

*group* Selects the type of data to collect: EZA, IUCV, NETIF, TLI, OE or TELNET.

- The EZA is a collection of trace points in Socket Management for processing EZASOCKET API activity with the following filter options:

HOST(*host*,...,*host*) – Up to 16 IP HOST addresses (names)

PORT(*port*,...,*port*) – Up to 16 port numbers (names)

USER(*jobname*,...,*jobname*) – Takes 1 to 16 *jobnames* that refer to jobs using the EZA function

UASID(*asid*,...,*asid*) – Takes 1 to 16 *asids* that refer to jobs using the EZA function

MAXEZADATA(*nnnn*) | MEZADATA(*nnnn*) | MEDATA(*nnnn*) – Where *nnnn* is a positive integer less than or equal to 65535

**Note:** By default, no user data is traced, so MAXEZADATA causes the data to be collected

- The IUCV keyword traces both IUCV and HPNS applications. IUCV can be specified with the following *filter* options:

HOST(*host,...,host*) – Up to 16 IP HOST addresses (names)

PORT(*port,...,port*) – Up to 16 port numbers (names)

USER(*jobname,...,jobname*) – Takes 1 to 16 *jobnames* that refer to jobs using the TLI function

Specify IUCVAPP for IUCV or HPNSAPP for HPNS.

UASID(*asid,...,asid*) – Takes 1 to 16 *asids* that refer to jobs using the TLI function

MAXIUCVDATA(*nnnn*) – Where *nnnn* is a positive integer less than or equal to 65535 (alias MIDATA)

**Note:** By default, no user data is traced, so MAXIUCVDATA causes the data to be collected

- NETIF is a network interface that includes the following filter options:

HOST(*host,...,host*) – Up to 16 IP HOST addresses (names)

PORT(*port,...,port*) – Up to 16 port numbers (names)

PROTOCOL – A single protocol (for example, UDP)

TYPE(*type*) for NETIF can be one of the following:

- TYPE(IP) – If datagram is an IP packet, then keep it
- TYPE(ARP) – If datagram is an ARP packet, then keep it

- TYPE(ALL) – Keep all datagrams (same as not specifying the filter)

MAXDATA(*nn*) – Alias MDATA. *nn* is between 1 and 65535. By default, the first 256 bytes of each packet are captured (to include the IP, TCP, and so on headers).

**Note:** Any value between 1 and 255 is increased to 256. There are no defaults for filters; everything is collected by default.

- OE (UNIX System Services)

HOST(*host,...,host*)

PORT(*port,...,port*)

USER(*jobname,...,jobname*)

UASID(*asid,...,asid*)

MAXOEDATA(*nn*). *nn* is between 1 and 65535. By default, no user data is captured.

INODE(*inode1,...,inode16*) can be specified.

**Note:** This requires you to know the inode in advance. It is intended more for post processing captured data (not for TCPEEP) under IPCS.

- TELNET is a collection of trace points in the TN3270E server to trace activity to/from the client and the VTAM application.

HOST(*host,...,host*) – Up to 16 IP HOST addresses names).

PORT(*port,...,port*) – Up to 16 port numbers (names).

LU(*luname,...,luname*) – Up to 16 LU names. These are either VTAM Applids or SLU names. This filter is restrictive in that, if specified, no tracing should be collected on a connection until either an SLU or APPLID is determined.

MAXTNDATA(*nn*). *nn* is between 1 and 65535. By default, no data is captured. (Alias is MTNDATA).

Refer to Useful TCPEEP Commands for usage of the GROUPS parameter.

- RTM— A collection of trace points in TelnetRTM that traces RTM activity. The following filter options are supported:
  - UASID(*asid*,...,*asid*)— Takes 1 to 16 *asids*, which refer to jobs using TelnetRTM (Unicenter NetSpy or a Unicenter TCPaccess Telnet server).
  - USER(*jobname*,...,*jobname*) — Takes 1 to 16 jobnames, which refer to jobs using TelnetRTM (Unicenter NetSpy or a Unicenter TCPaccess Telnet server).

Default: NETIF.

- TLI— A collection of trace points in the stack for processing TLI activity with the following filter options:

HOST(*host*,...,*host*)— Up to 16 IP HOST addresses (names)

PORT(*port*,...,*port*)— Up to 16 port numbers (names)

UASID(*asid*,...,*asid*)— Takes one to 16 *asids*, which refer to jobs using the TLI function

USER(*jobname*,...,*jobname*)— Takes 1 to 16 *jobnames* that refer to jobs using the TLI function

MAXTLIDATA(*nn*)— Alias MTLIDATA or MTDATA. *nn* is between 1 and 65535.

By default, no user data is captured.

EPID(*epid1*,...,*epid16*) may be specified. As this requires one to know the *epid* in advance, this is intended more for post-processing captured data (not for TCPEEP) under IPCS.

HALT	<p>Specifies that the trace instance identified by the INSTANCE keyword should be stopped.</p> <p>If HALT is specified, INSTANCE must be specified. If the INSTANCE is connected to an external writer, WTR=DISCONNECT must also be specified.</p>
INSTANCE ( <i>inst</i> )	<p>Modifies or stops a trace instance.</p> <p><i>inst</i> is the component trace address space ID. A new trace instance is created if <i>inst</i> is not specified.</p>
STATUS	<p>Displays all active trace instances.</p> <p><b>Note:</b> If STATUS is specified, any other keywords are ignored.</p>
TRACESIZE( <i>tsize</i> )	<p>Maximum number of trace records to record. If not specified, there is no limit to the number of records recorded.</p> <p>TRACESIZE is optional and can only be specified when creating a trace instance. If specified when modifying a trace instance, it is ignored.</p>
WRAP	<p>Create an in-memory WRAP trace.</p> <p>Not valid with WTR=<i>mem_name</i>, nor can the instance be modified later to specify WTR=<i>mem_name</i>. This is the default if NOWRAP is not specified and WTR=<i>mem_name</i> is omitted.</p>
NOWRAP	<p>Creates a NOWRAP trace. This is the default if WTR=<i>mem_name</i> is specified. The external writer writes the buffers when they fill up and after being written are reused.</p> <p>This option is for the Trace Address space and should not be confused with WRAP   NOWRAP on the TRACE CT command, which is for the IBM writer.</p>

## Comparing TCPEEP and MVS TRACE Syntax

The syntax for commands using TCPEEP differ from those used with the MVS TRACE facility.

OPTIONS To use MVS TRACE to specify NETIF and PROTOCOL(UDP), you need to specify:

```
OPTIONS = ( GROUPS ( ( NETIF, 'PROTOCOL ( UDP )' ) ) )
```

This is the same as the TSO TCPEEP command:

```
TCPEEP GROUPS ( ( NETIF, 'PROTOCOL ( UDP )' ) )
```

JOBNAME To specify JOBNAME using the MVS TRACE command, the syntax is:

```
JOBNAME = ( jobname1, ..., jobname16 )
```

When using TCPEEP the syntax is

```
TCPEEP JOBNAME ( jobname1, ..., jobname16 )
```

For more information about TCPEEP, see [TCPEEP](#).

## Trace Command Examples

The following examples demonstrate the control of the T03 Trace Facility using the MVS TRACE command. The examples assume a T03 Trace Facility is active and is using a Subsystem ID of ACTR.

Starting a Trace Instance

In this example a new trace instance is created. The instance is limited to tracing 10000 records for group ID NETIF from address space TCP51.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( GROUPS(NETIF ) TRACESIZE( 10000 ) ),JOBNAME=(TCP51),END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR909I Trace start successful Instance(01)
```

Modifying a Trace Instance

In this example an existing trace instance is modified. The trace instance is changed to include records for group ID API.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( INSTANCE( 1 ) GROUPS( TCP ) ),END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR911I Trace modify successful Instance(01)
```

Displaying Trace Status

In this example the status of the trace instances is displayed.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( STATUS ),END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR916I Instance(01) Active, records=23,745
T03TR916I Instance(02) Active, records=1,576
```

Stopping a Trace Instance

In this example an existing trace instance is stopped.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( INSTANCE( 1 ) HALT ),END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR910I Trace shutdown successful Instance(01)
```

Stopping All Trace Instances

In this example all existing trace instances are stopped.

```
TRACE CT,OFF,COMP=ACTR
T03TR910I Trace shutdown successful Instance(01)
T03TR910I Trace shutdown successful Instance(02)
```

Starting an External Writer

In this example an external writer is started. T03XWTR is the name of a predefined started task.

```
TRACE CT,WTRSTART=T03XWTR
```

Starting a Trace Instance and Connecting an External Writer

In this example a new trace instance is created and connected to an external writer. Only one instance can have an external writer connected.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( GROUPS( NETIF ) ),WTR=T03XWTR,END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR909I Trace start successful Instance(01)
```

Modifying a Trace Instance to Connect an External Writer

In this example an existing trace instance is connected to an external writer. Only one instance can have an external writer connected.

```
TRACE CT,ON,COMP=ACTR
R xx,OPTIONS=( INSTANCE( 1 ) ),WTR=T03XWTR,END
```

Where *xx* ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND. The resulting message is:

```
T03TR911I Trace modify successful Instance(01)
```

Modifying a Trace Instance to Disconnect an External Writer

In this example an existing trace instance is disconnected from an external writer.

```
TRACE CT,ON,COMP=ACTR  
R xx,OPTIONS=( INSTANCE( 1 ) ),WTR=DISCONNECT,END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR911I Trace modify successful Instance(01)
```

Stopping a Trace Instance and Disconnecting an External Writer

In this example an existing trace instance is to be stopped and disconnected from an external writer.

```
TRACE CT,ON,COMP=ACTR  
R xx,OPTIONS=( INSTANCE( 1 ) HALT ),WTR=DISCONNECT,END
```

Where *xx* ITT006A specifies the TRACE CT command operand. The resulting message is:

```
T03TR910I Trace shutdown successful Instance(01)
```

Stopping an External Writer

In this example an external writer is stopped. The writer must be disconnected from a trace instance before it can be stopped.

```
TRACE CT,WTRSTOP=T03XWTR
```

## Processing Trace Data

In order to process the data collected with the TRACE command, use the IPCS CTRACE command. It handles trace data that is in the MVS Component Trace Entry (CTE) format. By collecting trace data in this format, it is possible to use IPCS facilities to format the trace data. IPCS exit routines need to be written to locate trace records in dumps, filter trace records, and drive trace record formats.

## TRACERT

The TRACERT (trace route) TSO diagnostic command prints the route that packets take to get to a network host. TRACERT utilizes the IP protocol time-to-live field and attempts to elicit an ICMP TIME\_EXCEEDED response from each gateway along the path to some host. The only required parameter is the host name or IP address.

**Note:** To use the TRACERT command, you must PROTOCOL(RAW) defined in your TCPCFGxx member.

TRACERT requires the SAS/C Transient Library modules that are supplied in the LOAD/SASLOAD data set. This library is required in either your LOGON procedure or BATCH job STEPLIB DD.

```
TRACERT [ -drv ] [ data_size ] [ -g gateway ] [ -m max_ttl ] [ -p port# ]
        [ -q nqueries ] [ -s src_addr ] [ -n ] [ -t ts ]
        [ -w wait ] [ -x subsysid ] host
```

- d Turns on socket debugging (SO\_DEBUG).
- r Bypass normal routing tables and send directly to a host on an attached network. If the host is not on a directly attached network, an error is returned. Use to ping a local host through an interface that has no route through it (for example, after the interface was dropped by routed).
- v Specifies verbose output. Any ICMP packets that are received (other than TIME\_EXCEEDED and UNREACHABLE) are listed.
- data\_size* Specifies probe datagram length.  
Default: 38 bytes.
- g *gateway* Enables the IP LSRR (Loose Source Record Route) option in addition to the TTL tests.  
Useful for asking how the host at IP address *gateway* reaches a particular target.
- m *max\_ttl* Maximum time-to-live (number of hops) used in outgoing probe packets is *max\_ttl*.  
Default: 30 hops.
- p *port#* Base UDP port number used in probes is *port#*. TRACERT assumes that nothing is listening on UDP ports base to (base+nhops-1) at the destination host (so an ICMP message PORT\_UNREACHABLE is returned to terminate the route tracing).  
If something is listening on a port in the default range, you can use this option to pick an unused port range.  
Default: 33434.
- q *nqueries* Specifies the number of probes to sent.  
Default: Three.

- 
- s src\_addr* Specifies that *src\_addr* be used as the source address in outgoing probe packets. *src\_addr* must be specified as an IP number.  
On hosts with more than one IP address, this option can force the source address to be something other than the IP address of the interface the probe packet is sent on. If the IP address is not one of this machine's interface addresses, an error is returned and nothing is sent.
- n* Specifies that hop addresses be printed numerically rather than symbolically and numerically.  
This saves a nameserver address-to-name lookup for each gateway found on the path.
- t ts* Specifies the type of service in probe packets to be *ts*. This value must be a decimal integer in the range 0 to 255.  
Use this option to determine if different types of service result in different paths. Not all values of *ts* are legal or meaningful; see the IP specification for definitions. Useful values are *-t 16* (low delay) and *-t 8* (high throughput).  
Default: Zero.
- w wait* Sets the time to wait for a response to a probe to wait seconds.
- x subsysid* Specifies the subsystem ID of the Unicenter TCPaccess job or started task. *subsysid* is the four-character MVS subsystem ID.  
Default: ACSS
- host* Specifies the host name or IP address.  
**Note:** The *host* parameter should always be at the end of the statement.

## Usage Guidelines

To trace the route an IP packet travels to an internet host, TRACERT launches UDP probe packets with a small TTL, then listens for an ICMP time exceeded (TIME\_EXCEEDED) reply from a gateway. Probes are started with a TTL of one and are incremented by one until an ICMP port unreachable message (UNREACHABLE) is returned. This means the host was reached, the maximum number of hops was exceeded, or timeout was exceeded. Three probes are sent for each TTL setting.

## Example

The following example shows the output from this TRACERT command:

```
tracert -n -s 138.42.220.13 ibmnt.westend.com

tracert to IBMNT.WESTEND.COM. (194.231.119.65) from 138.42.220.13, 30 hops
max, 40 byte packets
 1 138.42.220.18 19 ms 18 ms 18 ms
 2 138.42.200.7 24 ms 22 ms 19 ms
 3 204.70.101.77 165 ms 163 ms 164 ms
 4 204.70.101.77 172 ms 161 ms 144 ms
 5 204.70.98.17 145 ms 148 ms 147 ms
 6 166.48.44.1 160 ms 162 ms 163 ms
 7 204.70.4.9 171 ms 174 ms 195 ms
 8 206.157.77.114 191 ms 194 ms 184 ms
 9 137.39.21.74 166 ms 181 ms 207 ms
10 137.39.69.69 207 ms 193 ms 203 ms
11 137.39.11.4 219 ms 228 ms 254 ms
12 137.39.156.246 395 ms 418 ms 345 ms
13 194.77.0.6 372 ms 381 ms 323 ms
14 194.77.0.78 346 ms 356 ms 396 ms
15 194.77.26.32 623 ms 510 ms 570 ms
16 194.231.119.65 577 ms * 522 ms
```

A line is displayed showing the TTL, gateway address, and round trip time of each probe. If answers come from several gateways, the address of each system is displayed. If no response is received in the three second timeout period, an asterisk (\*) is displayed for that probe.

The following list describes some possible notations:

---

!	An exclamation point following the time indicates the TTL is less than or equal to one.
H	Host unreachable received.
N	Network unreachable received.
P	Protocol unreachable received.

---

**Note:** TRACERT is useful for network testing, measurement, and management and should be used primarily for fault isolation. It can impose a load on the network and should not be used during normal operations or from automated scripts.

Source for the UNIX program traceroute and many BSD socket applications can be obtained via anonymous FTP from Internet host gatekeeper.dec.com.

# Index

## A

---

abbreviation, verb, 2-17

ABEND, collecting information for Customer Support, 4-5

### ACTEST

command description, 5-3  
usage with GateD, 4-9

### ALL

GTF statement parameter, 2-33  
parameter of the MODULE statement, 2-40  
SNAP statement parameter, 2-20

API trace procedures, 4-8

### APP

REFRESH statement parameter, 2-29  
task group identifier, 2-16

### ARM command

DEREGISTER parameter, 2-28  
ENABLE parameter, 2-28  
LISTEN parameter, 2-28  
NOLISTEN parameter, 2-28  
READY parameter, 2-28  
REGISTER parameter, 2-28  
START parameter, 2-28  
STOP parameter, 2-28  
WAITPRED parameter, 2-28

ARM, dynamic configuration command, 2-28

ARP, NETSTAT diagnostic command, 5-13

ATTR, POOL statement parameter, 2-44

## B

---

batch job, TCPEEP, 5-31

BINDSEC, REFRESH statement parameter, 2-29

## C

---

CACHE, DUMP statement parameter, 2-30

CANC, SYSSTAT diagnostic command, 5-21

CB, GTF statement parameter, 2-33

CHANGE, SYSSTAT diagnostic command, 5-22

character formatted socket trace, 4-17

### CLASS

LOGGING statement parameter, 2-37  
SNAP statement parameter, 2-20

CLEAR, P statement parameter, 2-42

### CNFG

NETSTAT diagnostic command, 5-14  
parameter of the UPDATE statement, 2-27  
START statement parameter, 2-46

command scripts, 2-53

FLUSH command statement, 2-54  
LIST command statement, 2-54  
NOFLUSH command statement, 2-54  
NOLIST command statement, 2-54

### commands

ARM dynamic configuration, 2-28  
ARP, NETSTAT diagnostic command, 5-13  
CANC, SYSSTAT diagnostic command, 5-21  
CHANGE, SYSSTAT diagnostic command, 5-22  
CNFG, NETSTAT diagnostic command, 5-14  
comments, adding, 2-19  
CONN, NETSTAT diagnostic command, 5-15  
DELETE dynamic configuration, 2-23

---

- delimiters, 2-19
- DNRGET, TSO diagnostic command, 5-8
- DUMP, DNR system command, 2-30
- dynamic configuration commands, 2-21
- dynamic configuration, TCPCFGUP member, 2-23
- format, 2-15
- GTF, IJT system command, 2-33
- HELP, NETSTAT diagnostic command, 5-14
- ILATCH, 2-8
- keyword operands, 2-18
- keywords, abbreviating, 2-19
- NETSTAT, TSO Telnet diagnostic command, 5-13
- objects, 2-17
- operands, 2-18
- operator, STOP, 2-3, 2-7
- OSPFMON, TSO diagnostic command, 5-23
- P CLEAR system command, 2-6
- PING, TSO diagnostic command, 5-26
- pool information, 2-42
- PPOST, GateD command, 4-10
- PURGE, DNR system command, 2-31
- REFRESH, system command, 2-29
- RIPQUERY, TSO diagnostic command, 5-27, 5-29
- ROUTE, NETSTAT diagnostic command, 5-17
- RTM, NETSTAT diagnostic command, 5-17
- starting task groups, 2-46
- STATUS, general task group command, 2-21
- SYSSTAT, TSO Telnet diagnostic commands, 5-21
- system, 2-15
- task group, 2-20
  - execution options, 2-45
  - information, 2-49
  - termination, 2-47
- TCPEEP, TSO Telnet diagnostic command, 5-31
- trace table status, 2-49
- TRACE, TSO diagnostic command, 5-48
- TRACERT, TSO diagnostic command, 5-62
- TSO
  - CONVXL8, 2-8
  - LOADXL8, 2-8
- TSO, Telnet diagnostic commands, 5-11
- Unix System Services commands, 2-32
- UPDATE dynamic configuration, 2-27
- USER, NETSTAT diagnostic command, 5-20
- verbs, 2-17
  - DISPLAY, 2-17
  - MODIFY, 2-17
  - VARY, 2-17
- XCF, NETSTAT diagnostic command, 5-20
- COMMANDS, HELP statement parameter, 2-34
- comments, adding, 2-19

- common data section table, SMF
  - FTP, 3-6
- Component Trace, 5-50
- configuration commands
  - ARM dynamic configuration, 2-28
  - DELETE dynamic configuration, 2-23
  - dynamic
    - overview, 2-21
    - TCPCFGUP member, 2-23
  - UPDATE dynamic configuration, 2-27
- configuration file, STARTxx member, 2-14
- CONN, NETSTAT diagnostic command, 5-15
- CONTENTION, ILATCH, 2-36
- CONVXL8 command, 2-50
- CONVXL8 command, 2-8, 2-50
- CSECT IUCVCONS values, 4-15
- customization, startup JCL, 2-11

---

## D

- DECLEN, MEM statement parameter, 2-41
- DELETE, dynamic configuration command, 2-23
- delimiters, 2-19
- DEMO, SET statement parameter, 2-45
- DEREGISTER
  - parameter of the ARM command, 2-28
- DEST, LOGGING statement parameter, 2-37
- detach MVS tasks, 2-5
- diagnostic commands
  - DNRGE, TSO diagnostic command, 5-8
  - dotted notation IP address wildcarding, 5-12
  - TSO
    - GTF, 2-33
    - HELP, 2-34
    - IFS, 2-34
    - ILATCH, 2-35
    - LOGGING, 2-37
    - MEM, 2-40, 2-41
    - MODULE, 2-40
    - P, 2-42
    - POOL, 2-42
    - SET, 2-45
    - SRC, 2-46

---

- START, 2-46
- STC, 2-47
- STOP, 2-47
- SVCDUMP, 2-48
- TASK, 2-49
- Telnet commands, 5-11
- TIME, 2-49
- TRACE, 2-49
- VSM, 2-50

diagnostic procedures, 4-1

DISABLE parameter of the ARM command, 2-28

#### DISPLAY

- command verb, 2-17
- GTF statement parameter, 2-33
- ILATCH statement parameter, 2-35

#### DNR

- DUMP system command, 2-30
- PURGE system command, 2-31
- queries, 5-8
- task group identifier, 2-16

DNRGE, TSO diagnostic command, 5-8

DUMP, DNR system command, 2-30

dynamic configuration commands

- overview, 2-21
- TCPCFGUP member, 2-23

## E

---

EI, GTF statement parameter, 2-33

ENABLE parameter of the ARM command, 2-28

ENTRIES, ILATCH statement parameter, 2-37

example, tracert output, 5-64

execution options, 2-45

## F

---

#### failures

- due to network problems, 4-2
- IUCV reporting, 4-14
- latch facility, 4-3
- loop detection, 4-3
- of major applications, 4-2
- storage shortages, 4-3
- that produce SVC dumps, 4-2

FREE, ILATCH statement parameter, 2-35

FREE=CLOSE, with logspin, 2-13

FTP Server, troubleshooting, 5-2

## G

---

#### GateD

- logs, 4-9
  - GTDERR, 4-9
  - GTDLOG, 4-9
  - GTDTRC, 4-9
- PPOST command, 4-10
- stopping with ACTEST, 4-9
- troubleshooting, 4-9

GENERAL, HELP statement parameter, 2-34

GREETING, REFRESH statement parameter, 2-29

GTDERR, GateD log, 4-9

GTDLOG, GateD log, 4-9

GTDTRC, GateD log, 4-9

#### GTF

- command script, 2-53
- IJT system command, 2-33
- SET statement parameter, 2-45
- tracing, 2-33

## H

---

hardware problems, 4-7

#### HELP

- IJT system command, 2-34
- NETSTAT diagnostic command, 5-14

HEXLEN, MEM statement, 2-41

## I

---

#### IFS

- IJT system command, 2-34
- MVS statement parameter, 2-41

IGNORE, UPDATE statement parameter, 2-27

IJT system commands

- GTF, 2-33
- HELP, 2-34

---

- IFS, 2-34
- ILATCH, 2-35
- LOGGING, 2-37
- MEM, 2-40, 2-41
- MODULE, 2-40
- P, 2-42
- POOL, 2-42
- SET, 2-45
- SRC, 2-46
- START, 2-46
- STCK, 2-47
- STOP, 2-47
- SVCDUMP, 2-48
- TASK, 2-49
- TIME, 2-49
- TRACE, 2-49
- VSM, 2-50

IJT task group identifier, 2-16

#### ILATCH

- IJT system command, 2-8
- IJT system command, 2-35

internal tracing, 2-49

Interval record tables

- subtypes 80 and 100, 3-23

#### IUCV

- application level problems, 4-11
- diagnosing performance problems, 4-14
- network message via TCPEEP trace tool, 4-13
- normal operation problems, 4-14
- reporting failures, 4-14
- socket function call problems, 4-12
- socket trace tables, 4-17
- trace function, 4-13
- transport problems, 4-11
- troubleshooting, 4-11

## J

---

#### JCL

- customization startup JCL, 2-11
- descriptions, 2-9
- distribution tape description, 2-9
- startup sample, 2-10

JESCT, MVS statement parameter, 2-41

## K

---

KEEP, ILATCH statement parameter, 2-37

keyword operands

- abbreviating, 2-19
- overview, 2-18

## L

---

latch lock utility, ILATCH, 2-8

LATCH, ILATCH statement parameter, 2-35

LISTEN parameter of the ARM command, 2-28

LNKLST, MVS statement parameter, 2-41

LOADXL8 command, 2-8, 2-52

LOGGING statement, implementing logspin utility, 2-13

LOGGING, IJT system command, 2-37

logspin

- implementing with LOGGING statement, 2-13
- T01LOG utility, 2-13

LUPARM, REFRESH statement parameter, 2-29

## M

---

maintenance status, 2-21

MAP task group identifier, 2-16

MEM, IJT system command, 2-40, 2-41

MEMBER

- START statement parameter, 2-46
- UPDATE statement parameter, 2-27

MOD

- GTF statement parameter, 2-33
- MEM statement parameter, 2-40

MODIFY

- command verb, 2-17
- GTF statement parameter, 2-33

MODULE, IJT system command, 2-40

MSG, ILATCH statement parameter, 2-36, 2-37

---

## MVS

- environment information, 2-41
- SMF record header table, 3-2
- SNAP statement parameter, 2-20

---

## N

### NAME

- DEVICE statement parameter, 2-25
- LNI statement parameter, 2-26

NAMESERVER, DUMP statement parameter, 2-30

### NETSTAT

- ARP diagnostic command, 5-13
- CNFG diagnostic command, 5-14
- CONN diagnostic command, 5-15
- HELP diagnostic command, 5-14
- ROUTE diagnostic command, 5-17
- RTM diagnostic command, 5-17
- TELNET, 5-18
- TSO Telnet diagnostic command, 5-13
- USER diagnostic command, 5-20

network message, recording via TCPEEP tracing tool for IUCV, 4-13

NOLISTEN parameter of the ARM command, 2-28

NOW, LOGGING statement parameter, 2-38

---

## O

### OFF

- GTF statement parameter, 2-33
- SET statement parameter, 2-45
- TRACE statement parameter, 2-49

### ON

- GTF statement parameter, 2-33
- SET statement parameter, 2-45
- TRACE statement parameter, 2-49

orderly shutdown, 2-47

ospf\_monitor, 5-23

### OSPFMON

- TSO diagnostic command, 5-23
- usage with GateD, 4-9

---

## P

P CLEAR system command, 2-6

P command, IJT system command, 2-42

### performance

- improving with logspin feature, 2-13
- problems, IUCV, diagnosing, 4-14

PING, TSO diagnostic command, 5-26

### POOL

- command options default values, 2-44
- HELP statement parameter, 2-34
- IJT system command, 2-42

pool, definition of, 2-42

positional operands, 2-18

PPOST command for GateD, 4-10

PRINT, LOGGING statement parameter, 2-38

### problems

- documentation requirements, 4-4
- hardware, 4-7
- incorrect FTP input, 4-6
- incorrect output through DNR EPS, etc., 4-6
- incorrect Telnet output, 4-5
- IUCV
  - application level, 4-11
  - during normal operation, 4-14
  - performance, diagnosing, 4-14
  - transport, 4-11
- socket function calls for IUCV, 4-12
- startup and parameter errors, 4-7

PURGE, DNR system command, 2-31

---

## R

R,TRACE command reply, 5-52

READY parameter of the ARM command, 2-28

recycling the TCP/IP stack, 2-7

REFRESH,system command, 2-29

REGISTER parameter of the ARM command, 2-28

### report writer

- commands, 3-28
- program, JCL, 3-27

restarting task groups, 2-7

---

ripquery, GateD usage, 4-9

RIPQUERY, TSO diagnostic command, 5-27

ROUTCDE, LOGGING statement parameter, 2-38

ROUTE, NETSTAT diagnostic command, 5-17

RPCINFO, TSO diagnostic command, 5-29

## S

---

S0C1 termination at shutdown, 2-7

S0C4 termination at shutdown, 2-7

scripts, command, 2-53

SCVT, MVS statement parameter, 2-41

SET, IJT system command, 2-45

shutdown

ABENDs, 2-7

fast, 2-4

orderly, 2-47

slow, 2-3

with P CLEAR, 2-6

SIZE, START statement parameter, 2-49

SMCA, MVS statement parameter, 2-41

SMF

Assembler API endpoint close subtype 152 table, 3-18

descriptor section table, 3-2

driver statistics subtype 100 data section table, 3-23

IUCV endpoint close subtype 151 table, 3-15

multi-volume FTP data section table, 3-11

MVS record header table, 3-2

OpenEdition endpoint close subtype 150 table, 3-14

OpenEdition subtype 150 user identification section table, 3-13

product section table, 3-3

report writer commands, 3-28

report writer program, 3-27

specification of, 3-1

standard record header samples, 3-2

subtype 110 - 123 protocol layer user identification section table, 3-20

subtype 110-123 tables, 3-19

subtypes 150 - 152 endpoint close tables, 3-13

task identification section table, 3-4

Telnet subtype 23 data section table, 3-11

user identification section table, 3-4

virtual storage statistics subtype 80 data section table, 3-24

SMF FTP

subtype 20/22 data section table, 3-7

subtype 21 data section table, 3-10

SNAP, general task group command, 2-20

SNM task group identifier, 2-16

socket trace tables for IUCV, 4-17

SPIN, LOGGING statement parameter, 2-39

SPOOL usage with T01LOG logspin utility, 2-13

SRC

IJT system command, 2-46

overview, 2-16

SSCT, MVS statement parameter, 2-41

START

DEVICE statement parameter, 2-25

IJT system command, 2-46

LNI statement parameter, 2-26

parameter of the ARM command, 2-28

starting

STARTxx command script, 2-53

the TCP/IP stack, 2-3

startup, 2-3

startup errors, 4-7

STARTxx member

command script, 2-53

customization, 2-14

PARM library, 2-14

STATIC, DUMP statement parameter, 2-31

STATUS, general task group command, 2-21

STCK, IJT system command, 2-47

STOP

DEVICE statement parameter, 2-25

IFS operator command, 2-7

IJT system command, 2-47

LNI statement parameter, 2-26

operator command, 2-3

parameter of the ARM command, 2-28

stop request

detach MVS tasks, 2-5

fast shutdown, 2-4

slow shutdown, 2-3

---

subsystem recognition character. *SRC*

subtype 20/22 data section table, SMF FTP, 3-7

SVCDUMP, IJT system command, 2-48

SYSOUT  
 files, T01LOG, 2-13  
 free SYSOUT data set, 2-13

SYSSTAT  
 CANC diagnostic command, 5-21  
 CHANGE diagnostic command, 5-22  
 TSO Telnet diagnostic command, 5-21

system commands, 2-15  
 DUMP, DNR task group, 2-30  
 GTF, IJT task group, 2-33  
 HELP, IJT task group, 2-34  
 IFS, IJT task group, 2-34  
 ILATCH, IJT task group, 2-35  
 LOGGING, IJT task group, 2-37  
 MEM, IJT task group, 2-40, 2-41  
 MODULE, IJT task group, 2-40  
 P, IJT task group, 2-42  
 POOL, IJT task group, 2-42  
 PURGE, DNR task group, 2-31  
 REFRESH, APP task group, 2-29  
 SET, IJT task group, 2-45  
 SNAP, general task group command, 2-20  
 SRC, IJT task group, 2-46  
 START, IJT task group, 2-46  
 STATUS, general task group command, 2-21  
 STCK, IJT task group, 2-47  
 STOP of IJT task group, 2-47  
 SVCDUMP of IJT task group, 2-48  
 TASK of IJT task group, 2-49  
 TIME of IJT task group, 2-49  
 TRACE of IJT task group, 2-49  
 VSM, IJT task group, 2-50

**T**

---

T01LOG, logspin utility, 2-13

tables  
 character formatted socket trace, 4-17  
 CSECT IUCVCONS values, 4-15  
 IUCV endpoint close subtype 151, 3-15  
 JCL descriptions, 2-9  
 multi-volume FTP data section, 3-11  
 POOL command options default values, 2-44

SMF Assembler API endpoint close subtype 152, 3-18  
 SMF descriptor section, 3-2  
 SMF driver statistics subtype 100 data section, 3-23  
 SMF FTP  
 common data section, 3-6  
 subtype 20/22 data section, 3-7  
 subtype 21 data section, 3-10  
 SMF MVS record header, 3-2  
 SMF OpenEdition endpoint close subtype 150, 3-14  
 SMF OpenEdition subtype 150 user identification section, 3-13  
 SMF product section, 3-3  
 SMF subtype 110 - 123 protocol layer user identification section, 3-20  
 SMF subtype 119 0 123, 3-19  
 SMF task identification section, 3-4  
 SMF Telnet subtype 23 data section, 3-11  
 SMF user identification section, 3-4  
 SMF virtual storage statistics subtype 80 data section, 3-24

TASK  
 IJT system command, 2-49  
 REFRESH statement parameter, 2-29

task groups  
 commands, 2-20  
 execution options, 2-45  
 graceful termination, 2-2  
 identifier  
 APP, 2-16  
 DNR, 2-16  
 IJT, 2-16  
 MAP, 2-16  
 SNM, 2-16  
 TCP, 2-16  
 TGI, 2-16  
 information, 2-49  
 maintenance status, 2-21  
 restarting, 2-7  
 starting, 2-46  
 termination, 2-42, 2-47

TCP REFRESH statement parameter, 2-29

TCP task group identifier, 2-16

TCP/IP stack  
 fast shutdown, 2-4  
 in drain mode, 2-3  
 introduction, 1-1  
 orderly shutdown, 2-2

---

slow shutdown, 2-3  
starting, 2-12  
stopping, 2-12

TCPEEP  
batch job, 5-31  
FORMAT, 5-34  
Format Options, 5-38  
GROUPS  
    IUCV or HPNS, 5-44  
    OE, 5-41, 5-56  
TELNET, 5-41, 5-56  
TN3270E, 5-41, 5-56  
TNASCII, 5-37  
TNDATA, 5-37  
TNEBCDIC, 5-37  
TSO diagnostic command, 5-31

TELNET  
NETSTAT command, 5-18  
tracing, 5-41, 5-56

Telnet diagnostic commands, 5-11  
ACTEST, 5-3  
NETSTAT TSO diagnostic commands, 5-13  
SYSSTAT TSO diagnostic command, 5-21

TERM, UPDATE statement parameter, 2-27

TEST, SET statement parameter, 2-45

TG, SET statement parameter, 2-45

TGI. *See* task group identifier

TIME  
IJT system command, 2-49  
ILATCH statement parameter, 2-35

TN3270E, tracing, 5-41, 5-56

TNASCII, 5-37

TNDATA, 5-37

TNEBCDIC, 5-37

trace  
function, IUCV, 4-13  
GTF, 2-33  
internal, 2-49  
table status, 2-49

TRACE  
command, 5-50  
CT, 5-50  
IJT system command, 2-49  
ILATCH statement parameter, 2-36  
REPLY command, 5-52  
TSO diagnostic command, 5-48

TRACERT  
output example, 5-64  
TSO diagnostic command, 5-62

tracing API problems, 4-8

translate table conversion, 2-50, 2-52  
CONVXL8 command, 2-8  
LOADXL8 command, 2-8

TRESET, ILATCH statement parameter, 2-36

TROFF, ILATCH statement parameter, 2-36

TRON, ILATCH statement parameter, 2-36

troubleshooting  
FTP Server, 5-2  
GateD, 4-9  
IUCV, 4-11

TSO commands  
CONVXL8, 2-50  
LOADXL8, 2-52

TSO Telnet diagnostic commands, 5-11

---

## U

---

Unix System Services commands, 2-32

UPDATE, dynamic configuration command, 2-27

USER  
NETSTAT diagnostic command, 5-20

USSTAB, REFRESH statement parameter, 2-29

---

## V

---

VARY, command verb, 2-17

verb abbreviation, 2-17

VSM, IJT system command, 2-50

VTAM major node activation, 2-8

---

**W**

---

WAITPRED parameter of the ARM command, 2-28

WTO, LOGGING statement parameter, 2-39

**X**

---

XCF, NETSTAT diagnostic command, 5-20







Computer Associates™