Unicenter

Management Services Command Reference

13th edition

P01- 012

Booklist

Computer Associates
The Software That Manages eBusiness

Edition	Publication Number	Publish Date
13th Edition	P01-012	October 2001

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.

© 2001 Computer Associates International, Inc.

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

Table of Contents

About This Manual xv

Notational Conventions xv
Icons xvii

Related Documentation xvii

Chapter 1 About Management Services 1-1

Management Services with NetMaster and SOLVE Products 1-1

Chapter 2 Management Services Commands 2-1

Summary Table..... 2-1

ABENDCMD 2-12

ACT 2-13

ACTLOGON 2-14

ALLOCATE (z/OS, OS/390, MSP, and VOS3)..... 2-15

ALLOCATE (VM/SP)..... 2-18

ALLOCATE (VSE/SP) 2-20

AOM GO 2-21

AOM NEWTAB..... 2-22

AOM PAUSE 2-24

AOM START 2-25
AT 2-28
ATF DEFINE 2-32
ATF LOAD 2-33
ATF UNLOAD 2-34
ATTACH 2-35
AUTOHOLD 2-37
CANCEL 2-39
CLEAR 2-42
CLSDST 2-43
CNM 2-45
CNMTRACE 2-46
CONNECT 2-49
CS- 2-50
CS+ 2-51
D 2-52
DEBUG 2-54
DEBUG BREAKPOINT 2-58
DEBUG CALLTRC 2-61
DEBUG CLEAR 2-62
DEBUG DISPLAY 2-63
DEBUG HOLD 2-65
DEBUG LIST BREAKPOINT 2-66
DEBUG MODIFY 2-67
DEBUG RESUME 2-69
DEBUG SET 2-70
DEBUG SOURCE 2-71
DEBUG START 2-73
DEBUG STEP 2-76
DEBUG STOP 2-78
DEBUG TRACE 2-80
DEFALIAS 2-81

DEFCLASS 2-84
DEFLINK (INMC) 2-91
DEFLINK (APPC)..... 2-93
DEFLINK (NVC) 2-96
DEFLOGON..... 2-98
DEFMODE 2-101
DEFMSG 2-103
DEFOPSET..... 2-105
DEFTERM..... 2-107
DEFTRANS..... 2-109
DELALIAS..... 2-115
DELCLASS 2-116
DELLINK 2-117
DELLOGON..... 2-119
DELMODE..... 2-120
DELOPSET 2-121
DELTERM 2-122
DELTRANS 2-123
DETACH 2-124
DISCONN..... 2-125
DNR 2-126
DOMAIN..... 2-130
EDB 2-132
EDIT 2-135
END 2-136
ENTER 2-137
EQUATE 2-140
EVERY 2-143
EXEC..... 2-147
F 2-149
FLUSH..... 2-151
FORCE 2-152

FSPROC 2-153
FSTOP 2-154
FTS 2-155
FTSINIT 2-156
FTSMOD 2-158
GO 2-160
INACT 2-161
INTQUE 2-162
ISR 2-163
ISR (Options for PPO) 2-166
ISR (Options for AOM) 2-169
ISR (Options for NTS) 2-174
JOURNAL 2-176
K 2-177
LIBPATH 2-178
LIBPATH DEFINE 2-179
LIBPATH DELETE 2-181
LIBPATH REPLACE 2-182
LIBRARY 2-183
LIBRARY DEFINE 2-184
LIBRARY DELETE 2-185
LIBRARY REFRESH 2-186
LIBRARY REPLACE 2-187
LINK (APPC) 2-188
LINK (INMC) 2-191
LINK (NVC) 2-195
LIST 2-197
LOCK 2-199
LOG 2-200
LOGSWAP 2-201
LSRPOOL 2-202
LTITLE 2-205

LUTRACE 2-206
MAIDISC 2-207
MAIINT 2-208
MAILOGON 2-209
MAISEND 2-212
MAISESS 2-213
MAISESS BUILD 2-215
MAISESS CANCEL 2-216
MAISESS DELETE 2-217
MAISESS MODIFY 2-218
MAISESS START 2-220
MAXUSERS 2-221
MSG 2-222
NCLCHECK 2-224
NCLTEST 2-225
NCLTRACE 2-227
NDB ALTER 2-229
NDB CREATE 2-232
NDB FIELD 2-234
NDB PURGE 2-236
NDB RESET 2-237
NDB START 2-238
NDB STOP 2-241
NDB UNLOAD 2-243
NETM 2-245
NETMCNTL 2-246
NETSTAT (IBM) 2-248
NETSTAT (TCPaccess) 2-250
NOTIFY 2-251
NOTRACE 2-252
NPTAB 2-253
NRDRET 2-254

NSBRO 2-255
NSLOOKUP 2-258
NSPCONN DEFINE 2-259
NSPCONN DELETE 2-260
NSPCONN START 2-261
NSPCONN STOP 2-262
NTSDBMOD 2-263
NTSMOD 2-265
OBEYFILE 2-268
OCSID 2-269
OPNDST 2-271
OPSYS 2-273
ORDER 2-274
PAGE 2-275
PF 2-276
PING 2-279
PPO 2-280
PROFILE 2-282
PROFILE EDS 2-291
PURGE 2-294
RECONN 2-297
REPALIAS 2-298
REPCLASS 2-300
REPLINK 2-307
REPLOGON 2-313
REPLY 2-316
REPMODE 2-317
REPOPSET 2-319
REPTRANS 2-320
REQMS 2-325
RETURN 2-328
ROUTE 2-329

SAWARE 2-331
SCRIPT 2-333
SECUSER 2-334
SHOW ALLOC 2-335
SHOW AOMABEND 2-336
SHOW AOMSTAT 2-337
SHOW APPC 2-338
SHOW ATF
SHOW ATF TABLE
SHOW ATF ENCODING 2-340
SHOW BRO 2-341
SHOW BUFF 2-342
SHOW CNMTRACE 2-343
SHOW COMMANDS 2-345
SHOW COMP 2-346
SHOW CONSOLES 2-347
SHOW DEBUG 2-349
SHOW DEFALIAS 2-351
SHOW DEFCLASS 2-353
SHOW DEFLINK 2-357
SHOW DEFLOGON 2-358
SHOW DEFMODE 2-359
SHOW DEFMSG 2-360
SHOW DEFOPSET 2-361
SHOW DEFTERM 2-362
SHOW DEFTRANS 2-363
SHOW DNR 2-364
SHOW DOMAINS 2-365
SHOW EDB 2-366
SHOW EDBSTAT 2-367
SHOW EDBUSER 2-368
SHOW EDBUSTAT 2-369
SHOW EDS 2-370

SHOW EPS 2-371
SHOW EQUATES 2-372
SHOW EXEC 2-373
SHOW FTS 2-375
SHOW ISR 2-377
SHOW ISRSTATS 2-379
SHOW LIB 2-380
SHOW LICENSE 2-381
SHOW LINK 2-382
SHOW LOCKS 2-386
SHOW LOGS 2-388
SHOW LSR 2-389
SHOW LUTRACE 2-390
SHOW MAI 2-391
SHOW MAISTAT 2-393
SHOW MAP 2-396
SHOW MSGQ 2-397
SHOW NCL 2-398
SHOW NCLGLBL 2-400
SHOW NCLSTAT 2-402
SHOW NCLVARS 2-405
SHOW NDB 2-406
SHOW NDBUSER 2-407
SHOW NETMCNTL 2-408
SHOW NETSPY 2-409
SHOW NETSPY DEFINITIONS 2-410
SHOW NETSPY USERS 2-411
SHOW NPF 2-412
SHOW NPTAB 2-413
SHOW NRD 2-414
SHOW NTS 2-415
SHOW NTSDBMOD 2-418

SHOW NTSSTATS 2-419
SHOW NTSUSER 2-421
SHOW OCS 2-422
SHOW PANELS 2-424
SHOW PARM 2-426
SHOW PATH 2-427
SHOW PAUSE 2-429
SHOW PPIUSERS 2-431
SHOW PPOSTAT 2-432
SHOW REPLY 2-433
SHOW SCNT 2-434
SHOW SERVER 2-435
SHOW SESS 2-437
SHOW SKEEP 2-439
SHOW SNAMS 2-441
SHOW SOCKETS 2-444
SHOW SSIEPS 2-445
SHOW SSISTATS 2-446
SHOW SSITERMS 2-447
SHOW SSIUSERS 2-448
SHOW STRACE 2-449
SHOW SUBSYS 2-450
SHOW SYSCONS 2-451
SHOW SYSPARMS 2-452
SHOW TCPIP 2-453
SHOW TERM 2-454
SHOW TIMER 2-455
SHOW TIMEZONE 2-456
SHOW TNCTL 2-457
SHOW TSO 2-458
SHOW TSS 2-459
SHOW UDB 2-460

SHOW UDBUSER 2-463
SHOW USERACCT 2-464
SHOW USERS 2-465
SHOW VARTABLES 2-467
SHOW VMOP 2-468
SHOW VSAM 2-469
SHOW XMIT 2-472
SHOW XNFTRACE 2-474
SHUTDOWN 2-475
SIGNOFF 2-477
SIGNON 2-478
SPLIT 2-482
SPO 2-483
SSI 2-484
START 2-485
STATUS 2-488
STRACE 2-489
SUBMIT 2-492
SUBMIT (AOMP) 2-494
SUBSYS 2-495
SUSLOGON 2-500
SWAP 2-501
SYSCMD 2-502
SYSLOG 2-508
SYSMON 2-509
SYSPARMS 2-510
TCPIP MODIFY 2-511
TCPIP QUIESCE 2-512
TCPIP START 2-513
TCPIP STOP 2-517
TELNET 2-518
TERMINAL 2-521

TIME 2-523
TIMEZONE 2-524
TITLE 2-525
TNCMD 2-526
TNCTL 2-527
TNCTL ADDPORT
TNCTL DEFPORT
TNCTL REPPORT 2-528
TNCTL DELPORT 2-531
TNDISC 2-532
TNSEND 2-533
TRACE 2-534
TRACEROUTE 2-535
TRANSMIT 2-537
UDBCTL 2-538
UNALLOC 2-542
V 2-543
X 2-545
XLATE 2-546
XNF 2-547
XNFTRACE 2-548
$SPLC15U ACTION=SNMPGET 2-549
$SPLC15U ACTION=SNMPSET 2-550

About This Manual

This manual is the principal command reference document for users of the NetMaster and SOLVE suite of products. It consists of a detailed description of each of the commands which are used to drive the system.

Notational Conventions

Each command description begins on a separate page, and covers the following points, where applicable:

Function:	A short description of the function, or purpose of the command
Use:	General description of use
Operands:	Description of operands
Return Codes:	Return codes set by the verb, with an explanation
Examples:	Examples of code to illustrate use
Notes:	Special aspects of use and related information
See Also:	A reference to other related commands

The precise syntax for each command, that is, how to enter the command and its operands, is defined in the area within the box at the beginning of each command description (see example over page).

The full command name appears in capital letters at the left hand side of this area.

The default minimum command authority level required to execute the command, appears in brackets below the command. The command authority may appear as *n/n*, indicating that individual operands have a different command authority from the command itself. The operands that are applicable to the command appear to the right of the command name. An example is as follows:

```
RECONN        [ ID=nnnnn ]
              [REGION={ CANCEL | DISC } ]
(0)
```

The command description format conventions are:

- UPPERCASE Characters
  Commands or operands consisting of uppercase characters must be entered as shown but can be entered in upper or lower case. Command names can be truncated to the minimum string that makes them unique. For example, the RECONN command can be abbreviated to REC.

- *Italic* Characters
  Italic characters are variables that show the kind of information, rather than the exact information, that must be supplied. The actual entry replaces the italic description. The types of valid data is described in the operands description section following the syntax box for each command.

- Underscored Values
  An underscored value indicates the default optional value that is assumed for an operand if that operand is not specified. In the example, the underscoring of CANCEL indicates that CANCEL is the default value for the REGION operand.

- Braces { }
  Braces are used to indicate the available options for a required operand. One of the alternatives listed must be selected. Do not include the braces when entering the desired option.

- Square Brackets [ ]
  Operands in square brackets are optional. This includes any accompanying equal signs. In the example, the entry of the REGION=CANCEL or REGION=DISC as an operand on the RECONN command is entirely optional. Do not include the square brackets when entering the desired option.

- Or-sign |
  The or-sign is used to separate options. If a group of options is enclosed by square brackets, and the individual options are separated by or-signs, none of the options in the group has to be chosen. If none of these operands are entered, the default value is used. Default values are always underscored.

- Commas and Equal Signs
  Commas and equal signs must be entered as shown. When commas and equal signs appear within brackets, they are optional and are to be used only if the accompanying optional operand is used.

Icons

Icons are used to draw your attention to important information. The following icons are used in this manual:

Warning

This icon indicates a warning to alert you to consequences of certain actions.

Key Concept

This icon indicates conceptual information to help you understand the task you are performing.

Related Documentation

The *Management Services User's Guide* is a guide for users of Management Services. It covers all aspects of terminal operation, system characteristics and system commands.

The *Management Services Administrator Guide* contains information concerning the implementation and administration of management services and the description of certain restricted functions.

Information on the operation and use of other NetMaster, NetSpy, and SOLVE products can be found in the appropriate product manuals.

Management Services applications are written in a language called *Network Control Language*, or NCL. Technical information on how to create your own NCL procedures is provided in the *Network Control Language User's Guide*. The syntax of NCL verbs and built-in functions is provided in the *Network Control Language Reference*.

1

About Management Services

Management Services is the name given to the central core of functions and services that are provided with each NetMaster and SOLVE product to assist in the management and operation of your organization's information processing environment. It is a comprehensive, yet flexible, network and systems management system for IBM and compatible mainframe environments, under a range of operating systems.

Management Services with NetMaster and SOLVE Products

Management Services is used by a set of NetMaster and SOLVE products to solve business problems. Management Services is present with each purchase, including optional facilities which are individually licensable.

The following paragraphs provide a brief description of Management Services and other NetMaster and SOLVE products.

Management Services

Management Services works in conjunction with any number of enterprise platforms and provides a range of tools, general services, and facilities that support the NetMaster and SOLVE product range.

The optional features of Management Services are products that solve specific business problems. These services include heterogeneous connection and network management services and a comprehensive development and customization language.

Management Services and NetMaster and SOLVE Products

NetMaster and SOLVE products are applications designed to solve specific business problems in the key operations areas of today's enterprises. These applications call on Management Services and by the common use of these services provide a highly integrated range of solutions to customer needs.

2

Management Services Commands

This chapter describes the available Management Services commands.

Summary Table

The following table is a list of the Management Services commands with a brief description of their function.

The column headed **Feature** indicates if a specific NetMaster or SOLVE feature or component must be included in the system initialization parameters at system startup, before the command can be used. For information about system initialization parameters, see the JCL parameters appendix in the *Management Services Administrator Guide*, in particular the PROD parameter description.

Note

Automation Services (AS) is an internal Management Services component that is enabled if any of the following products are configured in the region: FT, NETSPY, OPSCICS, OPSOS, SNA, SNAAUTO, TCPIP.

The column headed **Auth** indicates the default minimum authority level required to execute each command. The authority level may appear as *n/n*, indicating that individual operands have a different command authority from the command itself.

Command	Description	Feature	Auth
ABENDCMD	Sets the command to be issued if the Management Services system terminates abnormally		9
ACT	Activates a VTAM network node		1
ACTLOGON	Activates previously suspended logon paths		3
ALLOCATE	Allocates a cataloged or sysout or VSAM dataset to Management Services		0/3
AOM GO	Restarts the local AOM operating system interface after a PAUSE	AS	2
AOM NEWTAB	Compiles and (optionally) loads a new AOM Screening Table	AS	2
AOM PAUSE	Suspends the local AOM operating system interface	AS	2
AOM START	Starts the local AOM operating system interface	AS	2
AOM STOP	Terminates the local AOM operating system interface	AS	2
AT	Issues a command(s) or a message AT the specified time of day		0/2
ATF DEFINE	Defines a new transition name.		2
ATF LOAD	Loads translation tables for a specific translation name into storage.		2
ATF UNLOAD	Unloads translation tables for a specific translation name.		2
ATTACH	Attach a local non-SNA terminal to SOLVE SSI		3
AUTOHOLD	Determines the OCS screen display line mode		0
CANCEL	Cancels a nominated user or user region or window		0/1
CLEAR	Clears the current logical screen window		0
CLSDST	Closes the session(s) with the specified terminal(s)		1/0
CNM0	Starts and stops the VTAM CNM interface	SNA	3
CNMTRACE	Defines which CNM records are to be traced	SNA	3
CONNECT	Connects a terminal to Management Services using XNF		1
CS-	Retrieves the command from the bottom of the command stack		0
CS+	Retrieves the command from the top of the command stack		0
D	Displays a specific VTAM resource	AS	0
DEBUG	Controls the execution and debugging of NCL processes	NCL/DEBUG	0/3
DEFALIAS	Defines an alias entry for the Alias Name Translation Facility of NEWS	SNA	4
DEFCLASS	Defines RTM, SAW, Session or Resource classes	SNA	4
DEFLINK	Maintains the Dynamic Link Table, or DLT, used by LU6.2 services		3
DEFLOGON	Defines the logon information for the EASINET and MAI features		4

Command	Description	Feature	Auth
DEFMODE	Defines an entry in the Mode Control Table used by LU6.2 services		4
DEFMSG	Defines delivery options for PPO messages	AS	4
DEFOPSET	Defines an entry in the OSCT table		4
DEFTERM	Defines terminal type (VOS3 only)		2
DEFTRANS	Defines an entry in the or TCT used by LU6.2 services		4
DELALIAS	Deletes an alias entry used by the Alias Name Translation Facility	SNA	4
DELCLASS	Deletes an NTS class definition	SNA	4
DELLINK	Deletes a dynamic link definition		3
DELLOGON	Deletes an entry form EASINET/MAI appl table		4
DELMODE	Deletes an APPC MCT table entry		4
DELOPSET	Deletes an APPC Option Set Control Table (OSCT) entry		4
DELTERM	Deletes terminal definition (VOS3 only)		2
DELTRANS	Deletes an APPC Transaction Control Table (TCT) entry		4
DETACH	Detaches a local non-SNA terminal from SOLVE SSI		3
DISCONN	Disconnects either the current or the nominated processing region		0/1
DNR	Defines or modifies parameters for the SOLVE Domain Name Resolver (DNR)		4
DOMAIN	Defines or deletes a domain		3
EDB	Defines, starts, stops and deletes an EDB connection	EDBS	4
EDIT	Invokes panel maintenance from an OCS window or NCL procedure		0
END	Terminates a paused NCL process without further processing		0/2
ENTER	Defines the ENTER key		0
EQUATE	Assigns a text value to a unique string		0
EVERY	Issues a command or series of commands at a specified time frequency		0/2
EXEC	Schedules an NCL process for serial execution		0
F	Issues a VTAM modify (F) command	AS	1
FLUSH	Terminates an NCL process without further processing		0/2
FORCE	Forcibly inactivates a VTAM network node		1
FSPROC	Executes an NCL procedure in OCS fullscreen mode		0
FSTOP	Forces an immediate shutdown of the system		3
FTS	Involves File Transmission Services	FT	0
FTSINIT	Modifies the status of FTS initiators	FT	0

Command	Description	Feature	Auth
FTSMOD	Modifies the status of incoming or outgoing FTS transmissions	FT	0
GO	Resumes processing for a paused NCL process		0/2
INACT	Inactivates a VTAM network node	AS	1
INTQUE	Passes data to an NCL procedure's &INTREAD statement		0/2
ISR	Changes the status of ISR (Inter System Routing) conversations		2
JOURNAL	Controls the Management Services journal dataset		2
K	Clears the current OCS window		0
LIBPATH	Controls library path definitions		4
LIBRARY	Controls library definitions		4
LINK	Defines or changes the status of the INMC/NVC/APPC links		3
LIST	Lists statements in a specific member of the procedure library		0
LOCK	Suspends current screen processing		0
LOG	Writes messages to the activity log		0
LOGSWAP	Swaps logging to the next available log		3
LSRPOOL	Sets attributes for the VSAM shared resource pool		2
LTITLE	Sets the title for the Management Services logo panel		4
LUTRACE	Traces session traffic to selected terminals		4
MAIDISC	Forces disconnection of an MAI-OC session		0
MAIINT	Interrupts an MAI-OC application		0
MAILOGON	Creates an MAI-OC session with another application		0
MAISEND	Sends data to an application connected by an MAI-OC session		0
MAISESS	Provides functions for the control of MAI/FS sessions	SNAACCESS	0/4
MAXUSERS	Displays and limits the maximum number of concurrent users		3
MSG	Sends a message to the specified OCS Operator(s)		0
NCLCHECK	Tests syntax for an NCL procedure without execution		0
NCLTEST	Sets, resets, or displays NCL test status for this window		0
NCLTRACE	Dynamically alters the trace status of an executing NCL process		0/2
NDB ALTER	Builds, rebuilds or validates the key indexes for an NDB, or a field		2
NDB CREATE	Initializes a new $NDB database		2
NDB FIELD	Allows a database field definition to be added, deleted, or updated		2
NDB PURGE	Frees up a locked or halted database		2
NDB RESET	Deletes all data from a $NDB database		2

Command	Description	Feature	Auth
NDB START	Starts a $NDB database		2
NDB STOP	Flags a $NDB database as stopping		2
NDB UNLOAD	Unloads a copy of a $NDB database		2
NETM	Passes a command to NETM for execution (VOS3 only)		2
NETMCNTL	Controls communication between Management Services and NETM (VOS3 only)		2
NETSTAT	Invokes the TCP/IP NETSTAT command		–
NOTIFY	Sends a message to TSO/TSS users or CMS users		0
NOTRACE	Terminates a VTAM network trace	AS	1
NPTAB	Changes the status of a user's NPF table		0/4
NRDRET	Restores all Non-Roll Delete messages for an OCS window		0
NSBRO	Creates and sends a general or specific broadcast		3
NSLOOKUP	Uses the name service to find the name or address of a host		–
NSPCONN DEFINE	Defines a NetSpy connection.		4
NSPCONN DELETE	Deletes a connection to a NetSpy system.		4
NSPCONN START	Starts a connection to a NetSpy system.		4
NSPCONN STOP	Stops a connection to a NetSpy system.		4
NTSDBMOD	Alters database records or session keep counts on the NTS database	SNA	4
NTSMOD	Alters session trace and logging parameters of active sessions	SNA	4
OBEYFILE	Invokes the IBM TCP/IP OBEYFILE command		–
OCSID	Sets or resets OCS window identifiers		0
OPNDST	Connects a specific terminal to Management Services		1/0
OPSYS	Passes a command to the operating system for execution		2
ORDER	Reorders your OCS window display from top to bottom		0
PAGE	Ensures that output messages start at the top of the OCS window		0
PF	Displays and sets terminal Function keys		0
PING	Sends echo requests to a remote host		–
PPO	Starts and stops the VTAM PPO interface	AS	3
PROFILE	Displays or modifies a user profile		0/2
PROFILE EDS	Enables or disables event notification		0
PURGE	Purges and deletes current timer-initiated commands or NCL locks		0/2

Command	Description	Feature	Auth
RECONN	Reconnects the current terminal session with a disconnected region		0
REPALIAS	Replaces an alias name entry used NEWS	SNA	4
REPCLASS	Replaces an existing SAW, RTM, Session or Resource class definition	SNA	4
REPLINK	Replaces or defines an DLT definition		3
REPLOGON	Replaces an existing application logon entry		4
REPLY	Issues a VTAM REPLY command	AS	1
REPMODE	Replaces or defines an APPC Mode Control Table (MCT) entry		4
REPOPSET	Replaces or defines an APPC Option Set Control Table (OSCT) entry		4
REPTRANS	Replaces or defines an APPC Transaction Control Table (TCT) entry		4
REQMS	Sends data across the CNM interface	SNA	2
RETURN	Exits from any mode or function and returns to the primary menu		0
ROUTE	Sends a command string to a remote system		0
SAWARE	Controls NTS activity	SNA	4
SCRIPT	Starts or flushes an MAI-FS script procedure	SNAACCESS	0
SECUSER	Issues a command to a VM service machine	AS	2
SHOW AIF	Displays AIF users		0
SHOW ALLOC	Displays datasets allocated by the ALLOC command		0
SHOW AOMABEND	Displays diagnostic information if AOM ABENDs	AS	0
SHOW AOMSTAT	Displays AOM statistics	AS	0
SHOW APPC	Displays the status of all or selected APPC conversations		0
SHOW ATF	Displays information about the ATF tables.		0
SHOW BRO	Displays the current general broadcast messages		0
SHOW BUFF	Displays the current system storage and buffer utilization		0
SHOW CNMTRACE	Shows active CNM trace requests	SNA	0
SHOW COMMANDS	Displays commands available to user or system		0
SHOW COMP	Displays 3270 datastream compression statistics		0
SHOW CONSOLES	Displays the consoles currently allocated for use by AOM	AS	0
SHOW DEBUG	Displays NCL debug sessions	NCL/DEBUG	0/2

Command	Description	Feature	Auth
SHOW DEFALIAS	Displays one or more DEFALIAS entries	SNA	0
SHOW DEFCLASS	Displays NTS class definitions	SNA	0
SHOW DEFLINK	Displays current dynamic link definitions		0
SHOW DEFLOGON	Displays the current entries in the DEFLOGON table		0
SHOW DEFMODE	Displays APPC Mode Control Table (MCT) entries		0
SHOW DEFMSG	Displays the current DEFMSG delivery options	AS	0
SHOW DEFOPSET	Displays APPC Option Set Control Table (OSCT) entries		0
SHOW DEFTERM	Displays terminal definitions (VOS3 only)		0
SHOW DEFTRANS	Displays APPC Transaction Control Table (TCT) entries		0
SHOW DNR	Displays parameters or statistics for the SOLVE Domain Name Resolver (DNR)		0
SHOW DOMAINS	Displays the domain information of connected systems		0
SHOW EDB	Displays information about external database connections	EDBS	0
SHOW EDBSTAT	Displays statistics about currently defined EDB connections	EDBS	0
SHOW EDBUSER	Displays information about EDB users	EDBS	0
SHOW EDBUSTAT	Displays statistics about EDB users	EDBS	0
SHOW EDS	Displays current Event Distribution Services PROFILE definitions		0
SHOW EPS	Displays a list of all endpoints known to this system		0
SHOW EQUATES	Displays the current EQUATE strings available to this user		0
SHOW EXEC	Displays the names of members in a procedure library (OS/VS only)		0
SHOW FTS	Displays the status of file transmissions	FT	0
SHOW ISR	Displays ISR status information		0
SHOW ISRSTATS	Displays ISR statistics		0
SHOW LIB	Displays libraries which have been defined to the system		0

Command	Description	Feature	Auth
SHOW LICENSE	Displays the active licensing details and expiry dates		0
SHOW LINK	Displays the status of INMC or APPC links		0
SHOW LOCKS	Displays the status of all locks held within the system		0
SHOW LOGS	Displays the current status of the system activity logs		0
SHOW LSR	Displays LSR status information		0
SHOW LUTRACE	Displays the status of all LUTRACE requests		0
SHOW MAI	Displays information about MAI sessions for this user or all users		0
SHOW MAISTAT	Displays information about the status of MAI subtasks		0
SHOW MAP	Displays information about defined Mapping Services maps		0
SHOW MSGQ	Displays message queue depths		0
SHOW NCL	Displays the status of active NCL processes		0/2
SHOW NCLGLBL	Displays the names of any defined NCL global variables		2
SHOW NCLSTAT	Displays the status of NCL procedures currently in storage		0
SHOW NCLVARS	Displays information on variables in use by NCL procedures		0/4
SHOW NDB	Displays information about currently active or halted NDBs		0
SHOW NDBUSER	Displays a list of all users currently signed on to $NDB databases		0
SHOW NETMCNTL	Displays the status of NETM connectivity (VOS3 only)		0
SHOW NETSPY	Displays the status of currently defined NetSpy connections.		0
SHOW NETSPY DEFINITIONS	Displays the owners of currently defined NetSpy connections.		0
SHOW NETSPY USERS	Displays the current users of the NetSpy interface.		0
SHOW NPF	Displays user’s Network Partitioning Facility (NPF) tables		0
SHOW NPTAB	Displays the status of NPF resource tables		0
SHOW NRD	Displays the current queue of Non-Roll Delete Messages		0
SHOW NTS	Displays NTS resource or session information	SNA	0
SHOW NTSDBMOD	Displays currently executing NTSDBMOD & SHOW SKEEP processes	SNA	0
SHOW NTSSTATS	Displays NTS statistics	SNA	0

Command	Description	Feature	Auth
SHOW NTSUSER	Displays NTS users	SNA	0
SHOW OCS	Displays information about OCS users		0
SHOW PANELS	Displays Management Services panel queue information		0
SHOW PARM	Displays Management Services startup parameters		0
SHOW PATH	Displays current panel library path definitions		0
SHOW PAUSE	Displays the current PAUSE or WAIT status		0/2
SHOW PPIUSERS	Displays PPI user statistics		0
SHOW PPOSTAT	Displays PPO/SPO statistics	AS	0
SHOW REPLY	Displays VTAM messages that require a reply		0
SHOW SCNT	Displays the current session count		0
SHOW SERVER	Displays the status of registered server processes		0
SHOW SESS	Displays the terminals that are in session		0
SHOW SKEEP	Displays the NTS session keep counts for historical sessions	SNA	0
SHOW SOCKETS	Displays information about the use of TCP/IP services		0
SHOW SSIEPS	Displays SSI EPS information		0
SHOW SNAMS	Displays SNA Management Services information	SNA	0
SHOW SSISTATS	Displays Subsystem Interface (SSI) statistics		0
SHOW SSITERMS	Displays a list of terminals attached to SOLVE SSI		0
SHOW SSIUSERS	Displays information about SSI users		0
SHOW STRACE	Displays NTS session trace activity	SNA	0
SHOW SUBSYS	Displays user subsystem status		0
SHOW SYSCONS	Displays a list of currently logged on operating system consoles		0
SHOW SYSPARMS	Displays the current SYSPARMS settings		0
SHOW TCPIP	Displays information about the use of TCP/IP services		0
SHOW TERM	Displays the terminals that are in session		0
SHOW TIMER	Displays current timer initiated commands		0
SHOW TIMEZONE	Displays current time zones and the system time zone offset		0
SHOW TNCTL	Displays a list of all defined TNCTL (TELNET) ports and their option settings		

Command	Description	Feature	Auth
SHOW TSO	Displays current TSO users		0
SHOW TSS	Displays current TSS users		0
SHOW UDB	Displays VSAM dataset information		0
SHOW UDBUSER	Displays NCL UDB user information		0
SHOW USERACCT	Displays status of user NCL cpu time accounting		0/4
SHOW USERS	Displays the current signed on users		0
SHOW VARTABLES	Displays information about VARTABLES		0
SHOW VMOP	Displays VMOPERATOR active sessions (VM only)		0
SHOW VSAM	Displays extended VSAM dataset information		0
SHOW XMIT	Displays the status of FTS Transmission Requests	FT	0
SHOW XNFTRACE	Displays the status of all XNFTRACE requests (VOS3 only)		0
SHUTDOWN	Commences or cancels an orderly shutdown of the system		3
SIGNOFF	Signs off from a remote system		0
SIGNON	Signs on to a remote system		0/2
SPLIT	Opens or adjusts size of an OCS window		0
SPO	Starts and stops the VTAM Secondary Program Operator interface	AS	3
SSI	Signs off or stops Subsystem Interface (SSI)		0/3
START	Starts executing an asynchronous NCL process		0
STATUS	Displays current general system status		0
STRACE	Starts and stops session trace activity	SNA	4
SUBMIT	Passes a command to a background environment for processing		0/2
SUBSYS	Defines, starts, stops and deletes subsystems		3
SUSLOGON	Suspends an entry or entries in the DEFLOGON table		3
SWAP	Swaps current logical windows		0
SYSCMD	Sends a command to the operating system	AS	2
SYSLOG	Issues VMOPERATOR log browse commands (VM only)	AS	2
SYSMON	Sends data to System Monitor in 3600/4700	SNA	1
SYSPARMS	Defines or changes system default values. This command is described in the *Management Services Administrator Guide*.		4/9
TCPIP MODIFY	Alters the current trace options setting		4
TCPIP QUIESCE	Stops TCP/IP services when all the sockets are closed		4

Command	Description	Feature	Auth
TCPIP START	Initiates TCP/IP services		4
TCPIP STOP	Terminates TCP/IP services		4
TELNET	Starts a Telnet connection		–
TERMINAL	Provides extended color and/or highlighting datastreams to terminals		0
TIME	Sets and displays the current date and time		0
TIMEZONE	Maintains time zone names and offsets		3
TITLE	Sets the title to be displayed on the top line of the OCS window		4
TNCMD	Sends a Telnet command on a connection to a remote host		–
TNCTL ADDPORT	Defines a TELNET port		
TNCTL DEFPORT	Defines a TELNET port		
TNCTL DELPORT	Deletes a TELNET port		
TNCTL REPPORT	Replaces a TELNET port		
TNDISC	Disconnects a Telnet connection		–
TNSEND	Sends data on a Telnet connection		–
TRACE	Initiates a VTAM network trace		1
TRACEROUTE	Traces a route taken by TCP/IP packets to a remote host		–
TRANSMIT	Requests a dataset transmission	FT	0
UDBCTL	Controls the status of User DataBases (UDBs)		3/4
UNALLOC	Unallocates a closed dataset		0/3
V	Issues a VTAM VARY command	AS	1
X	Exits from OCS screen mode		0
XLATE	Tests alias name translation	SNA	1
XNF	Removes all XNF connectivity (VOS3 only)		3
XNFTRACE	Initiates or terminates tracing of XNF connectivity (VOS3 only)		4
$SPLC15U	Displays and modifies SNMP objects		

ABENDCMD

Function: Sets the operating system command to be issued if the MS region terminates abnormally.

ABENDCMD [*string*] (9)

Use: Sets a command which is issued to the operating system if the MS region terminates abnormally. The current ABENDCMD can be shown by issuing the STATUS command.

Operands: ***string***

The operating system command to be submitted. If no string is supplied, then the value is reset and no command is issued.

Examples:

```
ABENDCMD START SOLVPROD
```

Notes: ABENDCMD availability is dependent on operating system support.
Under certain circumstances the ABENDCMD may not be issued.

See Also: The STATUS command.

The *SYSPARMS Operands* appendix of the *Management Services Administrator Guide* for details of abend codes.

ACT

Function: Activates a VTAM network node.

ACT *name* (1)

Use: ACT builds and issues a VTAM VARY command with the format:

V NET,ACT,ID=*name*

This command can be used to activate network resources.

Operands: ***name***

The VTAM name for the resource to be activated.

Examples:

```
ACT NCP001
ACT LU054
```

Notes: Alternatives to this command would be:

```
V NET,ACT,ID=NCP001
V NCP001,ACT
```

See Also: The V command.

ACTLOGON

Function: Activates previously suspended logon paths.

```
ACTLOGON   { INPUT=entryname | APPL=applname }
(3)
```

Use: This command can be used to reactivate previously suspended logon paths. This allows users to again log on to the selected application from EASINET and the MAI facility.

Operands:

INPUT=*entryname*

The entry name of a defined DEFLOGON entry which is to be activated. Defined entries can be displayed using the SHOW DEFLOGON command. Previously suspended entries are displayed in high intensity. Use this operand if a single entry is to be activated.

APPL=*applname*

Allow logons to the specified application. Multiple entries can exist that allow users to log on to an application in different ways. This operand causes activation of all such entries.

Examples:

```
ACTLOGON APPL=IMS
ACTLOGON INPUT=TSOA
```

See Also: The SUSLOGON, DEFLOGON, DELLOGON, REPLOGON, and SHOW DEFLOGON commands.

ALLOCATE (z/OS, OS/390, MSP, and VOS3)

Function: Allocates a cataloged or SYSOUT dataset to a region (z/OS, OS/390, MSP, and VOS3 only).

```
ALLOCATE    [ DD=ddname ]
            { DSN=dataset name | SYSOUT=class }
            [ DISP={ OLD | SHR | MOD } ]
            [ CONCAT=ddname ]
            [ PGM=program ]
            [ COPIES={ nnn | 1 } ]
            [ DESTID=destination ]
            [ USERID=userid ]
            [ FCB=fcbname ]
            [ UCS=ucsname ]
            [ FOLD={ N | Y } ]
            [ HOLD={ N | Y } ]
            [ LRECL={ nnn | 132 } ]
            [ PRTCNTL={ A | M } ]
            [ FORM=forms ]
            [ MOUNT={ N | Y } ]
            [ FREE={ UNAL | CLOSE } ]
(0/3)
```

Use: MS regions support the dynamic allocation of cataloged or SYSOUT datasets through the ALLOCATE command (z/OS, OS/390, MSP, and VOS3). The format of the ALLOCATE command is shown above although the keywords and their operands can appear in any order.

A command authority level of at least 3 is required when allocation of other than non-INTRDR SYSOUT datasets is requested.

Operands: **DD=*ddname***

The DD operand can be used to provide a specific *ddname* to which the dataset specified is to be allocated. If no *ddname* is entered, then the system generates one and a message is returned indicating the name chosen.

DSN=*datasetname* | SYSOUT=*class*

The DSN= operand specifies the fully qualified name for the cataloged dataset to be allocated. This name should *not* be entered in quotes. This operand is required unless a sysout file is being allocated (see SYSOUT operand below).

The SYSOUT= operand specifies that a SYSOUT dataset is to be allocated and determines the JES2 or JES3 output class. This operand is mutually exclusive with DSN and DISP. All following keywords are related to SYSOUT.

ALLOCATE (z/OS, OS/390, MSP, and VOS3)

DISP={ OLD | DISP=SHR | DISP=MOD }
Specifies the allocated dataset's disposition. If not entered, defaults to OLD.

CONCAT=*ddname*
If this dataset is to be concatenated with other datasets, specifies the DD name of a previously allocated dataset which is also the DD name by which the group of concatenated datasets will be referenced. The order of concatenation is determined by the order in which the ALLOCATE commands are entered. See example below.

PGM=*program*
Provides the name for a SYSOUT processing program. This can be INTRDR if data written to this dataset is to be directed to the internal reader.

COPIES={ *nnn* | 1 }
The number of copies for SYSOUT datasets, in the range 1 to 255.

DESTID=*destination*
The JES2 or JES3 remote destination name for SYSOUT datasets.

USERID=*userid* (not supported on VOS3)
The TSO or VM user ID at that node at which the dataset will be delivered. This operand is valid only when DESTID= is also used.

FCB=*fcbname*
The name of an FCB to be used when processing the SYSOUT dataset.

UCS=*ucsname*
The UCS name on z/OS, OS/390, or MSP, or the character set table name on VOS3, to be associated with a SYSOUT dataset.

FOLD={ N | Y } (not supported on VOS3)
Specifies whether or not the FOLD option applies to this SYSOUT dataset.

HOLD={ N | Y }
Specifies whether the SYSOUT data is to be held or released upon dataset deallocation.

LRECL={ *nnn* | 132 }
Specifies the logical record length of the SYSOUT file being allocated. The LRECL *excludes* the print control character and logically sets the right margin of the file being printed. Options such as centering and justification of text apply accordingly. Range is 1 to 256.

PRTCNTL={ A | M }
Specifies the type of print control characters used in the file being printed. A indicates ASA control characters, M indicates machine code control characters. This option sets the RECFM value of the dataset.

FORM=*forms*
The forms name to be used when processing the SYSOUT dataset.

MOUNT={ Y | N }

Determines whether the allocation will request that the volume containing the dataset be mounted if it is currently not mounted (for example, tape volumes).

FREE={ UNAL | CLOSE }

Specifies, for sysout datasets only, whether the dataset is to be automatically freed at close. The default is to free only at unallocate (for example, the UNALLOC command). FREE=CLOSE will cause the dataset to be freed when closed, for example, by UDBCTL CLOSE or LOGSWAP.

Examples:

```
ALLOCATE DSN=USER1.DATA DD=FILE1 DISP=SHR MOUNT=Y
ALLOCATE DSN=NETWORK.CARDS
ALLOCATE DD=USRPRINT SYSOUT=A COPIES=5
ALLOCATE SYSOUT=X HOLD=YES
ALLOCATE DD=A DSN=MYDSN.NO1 DISP=SHR
ALLOCATE DSN=MYDSN.NO2 DISP=SHR CONCAT=A
ALLOCATE DSN=MYDSN.NO3 DISP=SHR
ALLOCATE DD=JCL SYSOUT=A PGM=INTRDR
ALLOCATE DD=USRPRINT SYSOUT=A LRECL=100 PRTCNTL=A
```

Notes:

Following a dataset's allocation, it can be used in the same manner as any other user datasets defined in Management Services execution JCL.

Allocation of SYSOUT datasets is allowed by users with a command authority level of 0. For non-SYSOUT datasets an authority of 3 is required.

Those datasets which have been dynamically allocated through the ALLOCATE command can be displayed using the SHOW ALLOC command.

Once closed, any dataset can be released by Management Services by using the UNALLOC command.

When datasets are concatenated using the CONCAT operand they can only be referenced using the prime DD name by which the group of datasets is known. If the prime DD name is subsequently freed using the UNALLOC command the entire concatenated group is no longer associated or recognized as a group and each DD name can be independently referenced. A separate UNALLOC command is required for each DD name to be unallocated. To change the order of a concatenated group of datasets they must be unallocated and reallocated in the desired sequence.

- If PRTCNTL=A is specified, the &FILE ADD statement can include the PRTCNTL operand to specify the control characters required (new line, new page and so on).
- If PRTCNTL=M is specified on the ALLOCATE, &FILE ADD cannot specify the print control characters, which default to 'skip one line'.

ALLOCATE command usage can be controlled by the dataset access authorization exit, which can be implemented to allow or reject attempts to allocate datasets. This exit is described in the *Management Services Administrator Guide*.

See Also:

The SHOW ALLOC, UNALLOC, and UDBCTL commands.

ALLOCATE (VM/SP)

Function: Allocates a VSAM dataset to an MS region (VM/SP only).

```
ALLOCATE     [ DD=dlblname ]
               DSN=datasetname
             [ MODE=filemode ]
             [ CAT=dlblname ]
(3)
```

Use: In VM/SP, MS regions support the dynamic allocation of VSAM datasets through the ALLOCATE command. The keywords and their operands can appear in any order.

Operands:

DD=*dlblname*

The DD operand can be used to provide a specific DLBL name to which the dataset specified is to be allocated. If the DD name is omitted, the system generates one and a message is returned indicating the DD name chosen.

DSN=*datasetname*

Specifies the full name for the VSAM dataset to be allocated. This name should *not* be entered in quotes. This operand is required.

MODE=*filemode*

Specifies the filemode of the minidisk for the VSAM dataset that is to be allocated. If omitted, this operand defaults to the SYSPARMS VSAMMODE value (that is, defaults to V).

CAT=*dlblname*

Specifies the DD name for the VSAM catalog with which the dataset is associated. This *dlblname* can be specified on a DLBL command included in the GCS startup procedure for Management Services, or it can itself have already been dynamically allocated by an earlier ALLOCATE command.

If this operand is omitted, it will default to value of the SYSPARMS VSAMCAT operand (which, in turn, defaults to IJSYSCT, the system master catalog).

Examples:

```
ALLOCATE DSN=USER1.DATA DD=FILE1 MODE=V1
ALLOCATE DSN=NETWORK.CARDS MODE=A CAT=NMUSRCT
```

Notes: ALLOCATE cannot be used to create new VSAM datasets. The allocated dataset must exist and must be cataloged, otherwise an error occurs when an open is attempted.

Following a dataset's allocation , it can be used in the same manner as any other user datasets defined in the MS execution JCL.

Datasets which have been dynamically allocated through the ALLOCATE command can be displayed using the SHOW ALLOC command.

Once closed, any dataset can be released by using the UNALLOC command.

See Also: The SHOW ALLOC, UNALLOC, and UDBCTL command descriptions.

ALLOCATE (VSE/SP)

Function: Allocates a VSAM dataset to an MS region (VSE/SP only).

```
ALLOCATE      [ DD=dlblname ]
                DSN=datasetname
              [ CAT=dlblname ]
(3)
```

Use: In VSE/SP, MS regions support the dynamic allocation of VSAM datasets through the ALLOCATE command. The keywords and their operands can appear in any order.

Operands:

DD=*dlblname*

The DD operand can be used to provide a specific DLBL name to which the dataset specified is to be allocated. If the DD name is omitted, the system generates one and a message is returned indicating the DD name chosen.

DSN=*datasetname*

Specifies the full name for the VSAM dataset to be allocated. This name should not be entered in quotes. This operand is required.

CAT=*dlblname*

Specifies the DD name for the VSAM catalog with which the dataset is associated. This *dlblname* can be specified on a DLBL command included in the Management Services JCL, or it can itself have already been dynamically allocated by an earlier ALLOCATE command.

If this operand is omitted, it will default to value of the SYSPARMS VSAMCAT operand (which, in turn, defaults to IJSYSCT, the system master catalog).

Examples:

```
ALLOCATE DSN=USER1.DATA DD=FILE1
ALLOCATE DSN=NETWORK.CARDS CAT=NMUSRCT
```

Notes: ALLOCATE cannot be used to create new VSAM datasets. The allocated dataset must exist and must be cataloged, otherwise an error will occur when an open is attempted.

Once a dataset is allocated, it can be used in the same manner as any other user datasets defined in the Management Services JCL. If the VSAM datasets attached using ALLOCATE are to be used by NCL, a UDBCTL command must be issued following successful allocation to notify NCL of the availability of the dataset.

Datasets which have been dynamically allocated through the ALLOCATE command can be displayed using the SHOW ALLOC command. Once closed, any dataset can be released by Management Services by using the UNALLOC command.

See Also: The SHOW ALLOC, UNALLOC, and UDBCTL commands.

AOM GO

Function: Restarts the local AOM operating system interface after a PAUSE.

```
AOM  GO
(2)
```

Use: This command is used to restart AOM after a pause. AOM can be placed into a pause state when it is started by using the AOM START PAUSE command. If AOM is already started, then it can be placed into a pause state by using the AOM PAUSE command.

While AOM is paused, the local operating system interface is suspended.

Notes: AOM must be active for this command to be accepted.

A message is queued to AOMPROC informing it of the execution of the AOM GO command.

See Also: AOM START and AOM PAUSE commands.

AOM NEWTAB

Function: Compiles and (optionally) loads a new AOM Screening Table.

```
AOM        NEWTAB  [ TEST | LOAD ]
           [ LIST [={ LOG | TERM | * } ] ]
           [ TABLE=name ]
(0/2)
```

Use: Before AOM is started the SYSPARMS AOMSTMEM=*name* command can be used to specify the name of the member containing the AOM screening table that is to be loaded when AOM is started.

If no screening table has been specified by the time AOM is started then no message screening will take place. That is, all messages will be routed to Management Services and none will be deleted.

After AOM is started the AOM NEWTAB command can be used to dynamically load or test a new or modified screening table. The SYSPARMS AOMSTMEM=*name* command or the TABLE=*name* operand is used to specify the name of a new or modified screening table to be used.

Operands:

TEST

Indicates that the syntax of the specified AOM screening table is to be checked, but the table is not to be loaded. This operand is mutually exclusive to the LOAD operand. The authority level for this operand is less than for the LOAD operand (default is 0).

LOAD

The specified AOM screening table is to be loaded. This operand is mutually exclusive with the TEST operand. The load of a new screening table can fail if the existing table is busy.

This operand is accepted only if AOM is running or paused.

Note

If the compile fails with errors, then the existing table remains active. If a new table name was specified with the TABLE= parameter, then the SYSPARMS AOMSTMEM is still updated.

LIST [={ LOG | TERM | * }]

Specifies where to direct the output listing of the AOM screening table. If LIST=LOG is coded the listing is sent to the log. If LIST, LIST=TERM or LIST=* is coded the listing is sent to both the requesting user's terminal and the log.

If no LIST operand is specified, no listing is produced.

The LIST operand can be used at any time—even while AOM is stopped.

TABLE=*name*

Provides a means of specifying an AOM Screening Table other than through the SYSPARMS AOMSTMEM=*name* command.

If both the TEST and TABLE=*name* operands are specified, then a test compile of the screening table specified by TABLE=*name* is performed. The current setting of AOMSTMEM is not updated.

If both the LOAD and TABLE=*name* operands are specified, then a compile and load of the screening table specified by TABLE=*name* is performed and the current value of AOMSTMEM is updated.

Note

AOMSTMEM is updated even if the compile contains errors. Therefore it is recommended that a *test* compile be performed before a new screening table is loaded.

Examples:

```
AOM NEWTAB TEST LIST=TERM
AOM NEWTAB TEST TABLE=AOMSCR01
AOM NEWTAB LOAD
```

Notes:

If an AOM screening table is loaded and in use then it is possible that an attempt to load a new or modified table can fail if there is a large amount of message traffic. AOM makes ten attempts to load the new table before issuing a LOAD FAILED message.

See Also:

AOM START command.

AOM PAUSE

Function: Suspends the local AOM operating system interface.

```
AOM   PAUSE
(2)
```

Use: This command is used to suspend the local AOM operating system interface. AOM will be left active but no messages or commands (z/OS, OS/390, MSP, or VOS3) will be screened or delivered by the local screening table. ISR traffic and AOMPROC remain active.

Examples: `AOM PAUSE`

Notes: This command is only valid if AOM is started.

It should be noted that DOMs will not flow while AOM is paused.

Messages arriving across ISR links, and AOMPROC NCL procedures are not affected by this command. A message is queued to AOMPROC to indicate that an AOM PAUSE command was executed.

See Also: AOM START and AOM GO commands.

AOM START

Function: Starts the local AOM operating system interface.

```
AOM  START    [ LIST [={ LOG | TERM | * } ] ]
              [ PAUSE ]
              [ TABLE={ name | NO } ]
(2)
```

Use: This command is used to start the local AOM operating system interface, and load the AOM Screening Table.

Operands:

LIST [={ LOG | TERM | * }]

Specifies where to direct the output listing of the AOM Screening Table. If LIST=LOG is coded the listing is sent to the log. If LIST, LIST=TERM or LIST=* is coded the listing is sent to both the requesting user's terminal and the log.

PAUSE

Specifies that AOM is to be started and the AOM Screening Table loaded but the operating system Interface is not to be activated.

The AOM GO command can be used to activate the operating Interface and commence processing at some later stage.

TABLE={ *name* | NO }

Provides a means of specifying an AOM Screening Table other than through the SYSPARMS AOMSTMEM=*name* command.

The screening table specified by TABLE=*name* is compiled and loaded. Any value currently specified by SYSPARMS AOMSTMEM=*name* is overridden and updated with the new screening table name. The value of AOMSTMEM is updated even if the compile of the new screening table produces errors.

TABLE=NO indicates that no table is wanted at this time. This has the same effect setting SYSPARMS AOMSTMEM=NO prior to starting AOM.

Examples:

```
AOM START PAUSE LIST=*
AOM START LIST=LOG
AOM START LIST=TERM TABLE=AOMSCR01
```

AOM START

Notes: The dataset containing the AOM Screening Table is specified by SYSPARMS AOMSTDDN=*ddname*. The actual AOM Screening Table to be used is specified by either the SYSPARMS AOMSTMEM=*name* command or the TABLE=*name* operand.

If no SYSPARMS AOMSTMEM command has been issued, and the TABLE operand of the AOM START command is also omitted, then AOM uses an internal, default table which passes all messages to Management Services for processing.

This command does not affect the status of AOM traffic across ISR links, except that the local system does not normally produce significant traffic if AOM is not started.

A message is queued to AOMPROC if AOM starts successfully.

See Also: AOM STOP, AOM GO, AOM PAUSE, and AOM NEWTAB commands.

AOM STOP

Function: Terminates the local AOM operating system interface.

```
AOM  STOP
(2)
```

Use: This command is used to terminate the local AOM operating system Interface component. AOM must be in either active or paused status.

Examples: `AOM STOP`

Notes: When AOM is stopped all non-roll delete messages present in Management Services that were sourced from the local AOM are deleted automatically. Messages written using &WRITE NRD=YES are not affected.

A message is queued to AOMPROC informing it of the AOM STOP command. This message is followed by DOM-NOTIFY messages for all outstanding messages where AOMPROC requested notification.

See Also: AOM START, AOM PAUSE, and AOM GO commands.

AT

Function: Issues a command or series of commands or a message at the specified time of day.

```
AT      { hh.mm.ss | *+hh.mm.ss }
        [ ENV={ PRI | CUR } ]
        [ KEEP={ MON | LOG | SYS } ]
        [ LIMIT=NO | nnn ]
        [ ROUTE={ userid | MON | LOG | SYS | * } ]
        [ PMSG=YES | NO ]
        [ TID=identifier ]
        { CMD=command text | MSG=message text | command text }
(0/2)
```

Use: Management Services supports the execution of commands at a predetermined time of day using the AT command.

The format of the AT command must be as defined above. The first field following the command keyword must be the time in the format HH.MM.SS. The time is based on a 24-hour clock and the manner of entry is reasonably flexible. For example:

21.30.00	9.30 PM
21.30	9.30 PM
6.3	6.03 AM
.30	12.30 AM (30 minutes past midnight)
9	9.00 AM
..50	50 seconds past midnight
*+1.30	current time plus one hour thirty minutes

Other optional operands must be specified ahead of any command text. The command text must be last.

Operands:

hh.mm.ss
The absolute time at which the command is to be executed.

+hh.mm.ss
The current time plus an interval at which the command is to be executed.

ENV=PRI | CUR
Specifies the processing environment in which the timer command is to execute. Default (PRI) is the OCS environment. CUR indicates that the timer command is to execute within the current processing environment. This option allows NCL processes to issue *private* timer commands using &INTCMD.

ENV=CUR will cause the timer command to be executed in the environment that the AT command is issued. In this case, the timer is automatically purged if the environment is terminated (for example, exiting the OCS window, or issuing an &INTCLEAR [TYPE=ALL]). The ROUTE and KEEP parameters can not be specified if ENV=CUR is specified.

KEEP=MON | LOG | SYS

The KEEP operand provides a means of avoiding automatic deletion of commands when a user is not logged on when execution is attempted. Three special environments are provided under whose control commands can be executed unattended.

When KEEP=MON is specified, the command is executed under the control of the Background Monitor environment if the original requesting user is not logged on. In this case the results of the command are displayed at all MONITOR terminals.

When KEEP=LOG is specified, the command is executed under the control of the Background Logger if the original requesting user is not logged on. In this case the results of the command are directed to the activity log and not sent to any terminals.

When KEEP=SYS is specified, the command is executed under the control of the Background System environment if the requesting user is not logged on. In this case, the results of the command are directed to the activity log and not sent to any terminals.

The KEEP operand and the ROUTE operand can both be specified. If the target user ID specified in the ROUTE operand is not logged on, then the KEEP operand is actioned.

LIMIT={ NO | *nnn* }

The number of times the command is to be executed before being automatically deleted. A limit of 1 should be specified if a command is only to be executed once and not retained for re-execution on following days. If this operand is omitted, or specified as LIMIT=NO, then no limit is imposed.

Note

Entering AT *+*hh.mm.ss* implicitly sets LIMIT=1 if LIMIT is not specified.

ROUTE=*userid* | MON | LOG | SYS | *

ROUTE=*userid* specifies the user ID under whose control this command is to be executed. Use of this operand requires command authority level 2. This operand suppresses automatic deletion of the command when the user ID is not logged on. The command is attempted at the specified time; if the nominated user is not logged, then the command is not executed but it is reset and retained for execution at the same time on the following day.

If ROUTE=MON is specified, then the command is executed under the control of the Background Monitor. This is a special internal environment which is always active. Commands executed in this way have all messages resulting from their execution routed to all monitor terminals and logged.

If ROUTE=LOG or ROUTE=SYS is specified, then the command is executed under the control of the Background Logger or Background System internal environment. Commands executed in this way have messages logged only, and not sent to any terminals.

ROUTE=* means the user ID of the AT command issuer.

PMSG={ YES | NO }

PMSG=NO will suppress the issue of the TIMER PURGED message that is normally issued when the LIMIT is reached. This is useful with ENV=CUR timers.

TID=*identifier*

Specifies a user-assigned identifier for the timer. This is in addition to the standard system-assigned timer number. The first character must be an alphabetic or national character, and all other characters alphanumeric or national. Length can be 1 - 12 characters. This identifier will be displayed in a SHOW TIMER list, and can be used in a PURGE TIMER = command. The identifier must be unique within all timers in the system. The values ALL or OWN cannot be used.

CMD=*command text* | MSG=*message text* | *command text*

The text of the command or commands to be executed, or the text of a message to be issued. For commands, this should be in exactly the format that the command would normally be entered. Concatenated commands are supported by the separation of commands by a colon (:). This will be converted to the normal concatenation character, a semicolon (;), prior to execution. This operand must be the last one coded.

Examples:

```
AT 10.00 CMD=ACT NEWYORK
AT 10.15 L=1 CMD=MSG ALL DON’T FORGET THE 10.30 MEETING
AT 12.00 ROUTE=OPER4 LIMIT=1 D BFRUSE
AT 13.00 ROUTE=NM3BLOG D CDRMS
AT 13.20 EVERY ..10 LIMIT=4 SHOW LINK
AT 22 K=MON EX REMOTE4
AT 12:00 MSG=GO TO LUNCH
```

Notes: The ROUTE and KEEP operands require command authority level 2 by default.

Timer commands are not remembered across a system restart. It is therefore advisable to place timer commands that are to be executed on a permanent basis in either the NMINIT or NMREADY initialization procedures.

The EVERY command can be specified as the command within an AT command. This allows a sequence of operations to be started at a particular time of day.

The KEEP, ROUTE, and LIMIT operands can be abbreviated to their first character. For example:

```
R=USER1
```

AT commands entered in the NMINIT and NMREADY procedures that are automatically executed during system initialization are executed under the Background System environment, which has a user ID of *xxxx*BSYS where *xxxx* is the MS region *shortname* (domain id) as specified on the NMSUP execution parameter. Such commands also bypass the command authority level checks imposed on certain operands such as KEEP.

If neither KEEP nor ROUTE is specified, the default ROUTE=LOG operand will be assumed, to prevent automatic deletion.

You can specify your own user ID in the ROUTE operand. This technique provides a means of ensuring that a timer command is not automatically deleted when you are not logged on, and is only executed when you are logged on.

The SUBMIT command provides an alternative method of scheduling timer commands to execute under the various internal system environments.

See Also: The EVERY, PURGE, SHOW TIMER, and SUBMIT commands.

ATF DEFINE

Function: This command defines a new translation name.

```
ATF DEFINE    tablename ENCODING=encodingname
              PRIMOD=modname [ SECMOD=modname ]
(2)
```

Use: To define a new translation name, do this:

1. Code up (primary and possibly secondary) source translation files. If you are modeling your new name on an existing one, as a shortcut you can use that name's primary file and augment it with a secondary file of your own.
2. Compile the file(s) to load modules (named as you want).
3. Issue the ATF DEFINE command to register your new name.

Operands:

tablename
A user-nominated table name, from 1 to 16 characters. The first character must be alpha or national, the remainder alphanumeric, national, or underscore.
This name must not be the same as any existing defined ATF translation name.

encodingname
The encoding rule being used by this table. This must be a valid, recognized table-based encoding name. See Chapter 4 in the *Advanced Translation Facility Reference*.

modname
The name of a load module (primary and maybe secondary). These must be the load module names of the modules generated from your source files.
The TABLE operand in files used to generate the load modules need not match the tablename for this table.

Examples:

```
ATF DEFINE MYASCII ENCODING=SINGLE PRIMOD=XTASCIIP SECMOD=MYASCIIT
```

This example defines a new translation table called MYASCII based on the distributed ASCII table, with overrides as described in a table module called MYASCIIT.

Notes: As the example shows, you can create tables similar to distributed ones by using the distributed table file load module, with an overriding secondary module.

See Also: The SHOW ATF and ATF LOAD commands.

ATF LOAD

Function: This command loads translation tables for a specific translation name into storage. This is normally done the first time that a translation name is used.

ATF LOAD (2)	*tablename*

Use: By issuing the ATF LOAD command, you can see the resulting response to the command and can check for errors.

Operands: ***tablename***

The name (from 1 to 16 characters long) of a predefined table to load. The first character must be alphabetic or national, the remainder alphanumeric, national, or underscore.

Examples:

```
ATF LOAD MYASCII
```

Notes: You can only load tables that have been defined (or predefined by the MS region) and have table-based encoding rules. Algorithm-based tables cannot be loaded because they have no external tables to load.

See Also: The SHOW ATF and ATF UNLOAD commands.

ATF UNLOAD

Function: This command unloads translation tables for a specific translation name.

ATF UNLOAD *tablename* (2)

Use: If you have tailored and recompiled source translation files, you can reload the translation tables without stopping and restarting the MS region by using an ATF UNLOAD command followed by an ATF LOAD command.

Operands: ***tablename***

The name (from 1 to 16 characters long) of a predefined table to unload. The first character must be alphabetic or national, the remainder alphanumeric, national, or underscore.

Examples:

```
ATF LOAD MYASCII
```

Notes: This command is useful when you have changed the table load modules and want to use these changes immediately.

See Also: The SHOW ATF and ATF LOAD commands.

ATTACH

Function: Attaches a local non-SNA terminal to the SOLVE Subsystem Interface (SSI).

```
ATTACH addr   [ NAME=nnnnnnnn ]
              [ AUTOLOG=NO|YES ]
              [ DATA=logon-data ]
(3)
```

Use: The ATTACH command is used to attach a local non-SNA terminal to SOLVE SSI. The SOLVE SSI program supports communication with the terminal and acts as a communication program which enables communication between the terminal and your MS region.

Operands: ***addr***

Specifies the 3- or 4-digit device address of the device to be attached.

NAME=*nnnnnnnn*

Specifies a symbolic name which is to be assigned to the terminal. If this operand is omitted, a default name is assigned, which consists of a 1-5 character prefix followed by the 3 or 4 digit device address. The prefix may be specified by the TPREFIX= SOLVE SSI startup parameter, which has a default value of $LOCL.

If the device address has 4 significant digits and the TPREFIX value has 5 characters, an error will result, as the terminal name cannot exceed 8 characters. A 5-character prefix is acceptable with a 4-digit device number as long as the leading digit is zero. The last 3 digits only are used (for example, NVTRM and 04DC would result in a name of NVTRM4DC).

AUTOLOG=NO|YES

Specifies that when the device is attached to SOLVE SSI it is to be immediately passed to the region. If the terminal is turned off when it is initially attached, it is passed to the region when it is turned on. User data may be passed to the region by using the DATA= operand.

The default value is AUTOLOG=NO. This causes the terminal to remain under SOLVE SSI control. SOLVE SSI displays a logo on the terminal.

If the link from SOLVE SSI to the region is stopped and restarted, the AUTOLOG option is honored when the link becomes active again.

DATA=*logon-data*

Specifies optional user data to be passed when a terminal which is attached with AUTOLOG=YES is passed to the MS region automatically.

ATTACH

Examples:

```
ATTACH 4DF
ATTACH 4DF NAME=TERMNAME AUTOLOG=YES DATA=USERID PASSWORD OPT
```

See Also: The DETACH and SHOW SSITERMS commands.

AUTOHOLD

Function: Determines whether the OCS screen rolls to reuse the top screen display line, or freezes at the bottom of the screen until signalled to continue.

AUTOHOLD [ON | OFF | AUTO | NOAUTO] (0)

Use: The OCS screen displays from top to bottom of the screen in a rolling mode. Once the screen fills (that is, the bottom display line is written) it pauses for a predefined roll time and then wraps back to the top of the screen, overwriting the top line.

The AUTOHOLD command is used to freeze the screen after the bottom line has been written, and to wait for some entry from the operator to signal that the screen can be rolled and overwrite the top line. AUTOHOLD OFF is the default.

In addition, OCS can enter an automatic mode of autohold where message arrival would normally cause overwriting after the roll-delay had been observed. This AUTOHOLD AUTO is the default and provides a useful way of controlling forward paging so that no data is missed.

Operands:

ON

Places the OCS screen in AUTOHOLD mode. This manual autohold request overrides the AUTOHOLD AUTO option.

OFF (default setting)

Returns the screen to normal roll mode. If, however, the screen is currently frozen with the AUTOHOLD caption displayed, AUTOHOLD OFF allows all queued messages to be displayed with no further freezing of the window, pausing only for the roll-delay time at the end of each window-full of messages. After these messages are displayed, the window again behaves according to the current AUTO/NOAUTO setting.

AUTO (default setting)

Automatically freezes the window when a message fills the last line of the window *only* if there were enough messages queued to cause the messages to wrap past the line on which the first message was displayed. This is most useful when operating with a relatively short roll-delay time to ensure no messages are missed. An AUTOHOLD ON condition takes precedence over the automatic autohold.

NOAUTO

Returns the screen to the default autohold settings of AUTOHOLD OFF.

(no operands)

Specifying AUTOHOLD without any operands flips states between AUTOHOLD ON and AUTOHOLD OFF.

Examples:

```
AUTOHOLD
AUTOHOLD ON
AUTOHOLD AUTO
```

AUTOHOLD

Notes: When entering OCS from the primary menu, AUTOHOLD is set to OFF and AUTO.

When the screen freezes after the bottom line is written, the caption AUTOHOLD is displayed just above the command entry line. This is deleted when the next keyboard entry is made.

If the AUTOHOLD freeze condition is allowed to remain in effect when a large number of messages are being sent to the screen, the operator is warned when 75% of the message hold stack is full, and again at 100%. After the message hold stack limit is reached (the default size is 200 messages), subsequent messages cause the earliest message to be discarded and the newest message to be added to the bottom of the stack. The operator is again warned of the fact that messages are being lost every 10 messages from that point, until the AUTOHOLD condition is released by some entry from the terminal. These warnings are given by changing the caption above the command entry line to 75% LIMIT, 100% LIMIT, and MSGS LOST respectively.

You can immediately freeze the screen on any output line, by pressing the ENTER key with a blank command line. This is called a NULL entry and it is normally used for the operator to lock the screen momentarily while a critical message is read. When the screen is frozen in this manner, the HOLDING caption is displayed just above the command entry line. This condition is also broken by any other input, and the same warning messages can be sent as for AUTOHOLD if excessive messages accumulate.

An AUTOHOLD or HOLDING caption is changed to MSG QUED if another unsolicited message arrives.

The size of the message hold stack and the screen Roll Delay time can be altered using the PROFILE command.

When AUTOHOLD is ON, the PAGE command (which is often used in NCL procedures to clear the screen before displaying data) causes the AUTOHOLD caption to be written and then wait for some input before the screen is cleared and the new data displayed. This makes the PAGE command a better alternative than CLEAR (or K) for concatenation with timer commands, to avoid inadvertent clearing of the screen when the timer command executes.

See Also: The PROFILE command ROLL and HOLD operands.

CANCEL

Function: Cancels a nominated user or user region. Also cancels your other logical window.

```
CANCEL        { { [ U={ userid | * } ] [ LU={ region | * } ] } [FORCE] |
                ID=nnnnn | WINDOW }
(0/1)
```

Use: This command can be used to cancel a nominated user ID who is currently logged on, or who has a disconnected environment.

This command can also be used to terminate your other logical processing window.

One of the following operands must be supplied.

Operands:

U={ *userid* | * }
The user ID that is to be cancelled. An asterisk (*) specifies that your own user ID is to be cancelled.

LU={ *region* | * }
The user region (LU name) that is to be cancelled. An asterisk specifies that the issuing region is to be cancelled.

FORCE
Results in the immediate signoff of the user. It will only be accepted by regions that have previously processed CANCEL, and is only applicable to LU1 and 3270 type user regions.

ID=*nnnnn*
The specified disconnected region that is to be cancelled. The reconnection identifier *nnnnn* should be in the range 1 to 65535.

WINDOW
Specifies that your other logical window is to be terminated. If no other window is operational, then the command is ignored.

CANCEL

Examples:

`CANCEL U=USER1`	Terminate processing for USER1
`CANCEL U=USER1 FORCE`	Immediate signoff of USER1
`CANCEL U=*`	Cancel yourself
`CANCEL LU=T024L08A`	Cancel the user region for T024L08A
`CANCEL WINDOW`	Terminate other window processing
`CANCEL ID=834`	Cancel the disconnect region '834'

Notes:

Cancellation of an active user terminates the user's operational environment. Any processing currently in progress will be terminated, including MAI sessions, and the user will be logged off. The terminal remains under EASINET control if the EASINET facility is in use.

A reconnect environment, as displayed by SHOW USERS, can be cancelled by CANCEL U=*disconnected-userid*.

If a user is connected a multiple number of times, then all connections will be processed by CANCEL U=*userid* or CANCEL U=*, including disconnected regions.

Multiple connections can be cancelled independently by using CANCEL LU= or CANCEL ID=, where only the specified target region is processed for termination. An LU name or reconnection identifier is displayed by the SHOW USERS command.

CANCEL WINDOW can be used to free a window that hangs while developing an NCL procedure. Use the attention key (ATTN) to unlock the keyboard and then SWAP (F9) to open the other window. Use OCS (Primary Menu option O) to enter CANCEL WINDOW.

CANCEL LU= can be used to flush and restart the EASINET procedure at a specified terminal. The CLSDST command should be used if the terminal is to be disconnected from EASINET.

The FORCE operand may be used when a previous CANCEL command has failed to terminate the user region. To assist in problem determination, the FORCE operand results in the logging of the status of the user region affected. This information should be supplied to your product support representative.

After the success of a forced cancel, the resources in use by the user region may not be freed. For example, if a user is cancelled when performing an update to a record, other users may have subsequent problems accessing this record.

An authority level of 1 is required to cancel regions belonging to other users.

See Also: The CLSDST and SHOW USERS commands.

CLEAR

Function: Clears the current logical screen window.

```
CLEAR        [ A | ALL ]
             [ R | RESP ]
(0)
```

Use: This command can be used to erase all display lines in the current logical window. If multiple windows are operative during split screen operation only the window from which the command is entered will be cleared.

Operands:

A | ALL

Specification of ALL deletes all messages currently displayed or queued for display at the window from which the command is entered. Messages with the non-roll delete attribute are not deleted. Unsolicited messages queued for display are not deleted.

R | RESP

RESP indicates that all messages currently displayed are to be deleted, and those messages that are responses to earlier command entered from this window are also to be deleted. Messages with an 'unsolicited' attribute that are queued for display are not deleted. Non-roll delete messages are never deleted by this command.

Examples:

```
CLEAR ALL
CLEAR;D BFRUSE
```

Notes: Use of this command is ideal when concatenated ahead of another command that requires the full screen to contain the output. This command can be used to ensure that the full screen is available.

If a clear function is to be performed from within an NCL procedure it is recommended that the PAGE command be used. The PAGE command is synchronous with other messages.

See Also: The K and PAGE commands.

CLSDST

Function: Closes the session with the specified terminal(s).

```
CLSDST      { name | name* GENERIC | * }
            [ WAIT={ YES | NO } ]
(1/0)
```

Use: This command can be used to close and terminate the session with a specific terminal or all terminals whose names match the supplied mask, or to terminate your own session (when * specified). Regardless of the current functions being executed the specified terminals will be disconnected and all user functions terminated.

The terminal is released from the EASINET facility if it is operational.

Operands: ***name***

VTAM node name of terminal with which session is to be terminated.

> **Note**
> If you are closing a session with a printer, then the command authority required is 0. All other sessions require a command authority of 1.

***name** GENERIC**

When an asterisk is placed in the node name and the GENERIC operand is specified, the asterisk acts as a wildcard. All sessions with node names that match the name specified, allowing for a wildcard character, have their sessions terminated. One or more asterisks are allowed denoting wildcards in any position, including the first. If the last character of the name is an asterisk, it is treated as though all remaining characters in the name are also asterisks. Note that a single asterisk with the GENERIC operand specified is identical to ******** GENERIC and terminates all sessions.

*****

Terminate your session with Management Services (that is, command issuer is released from Management Services).

WAIT={ YES | NO }

Applies only to LU1 terminals, and is ignored if specified for other terminal types. WAIT=YES specifies that the terminal will not be closed until all messages currently queued for output to the terminal have been delivered. If, after all messages queued before the CLSDST is issued have been delivered, further messages are found to be queued, the CLSDST is ignored. The response to the CLSDST command with WAIT=YES signifies only that the command has been accepted. Messages indicating the success or otherwise of the eventual close attempt are sent to the activity log only. The default for the WAIT operand is NO.

Examples:

```
CLSDST LU01
CLSDST TERM3 WAIT=YES
CLSDST *
CLSDST AB*D* GENERIC
```

CLSDST

Notes: This command can only be used to terminate sessions for terminals which are currently in session with Management Services.

Use the CANCEL command if a user is to be cancelled while retaining the terminal under the control of EASINET.

This command also terminates the user's MAI sessions. Use the DISCONN command if you want to disconnect the user ID while maintaining MAI sessions.

See Also: The CANCEL, DISCONN, and OPNDST command descriptions.

CNM

Function: Starts and stops the VTAM CNM interface.

```
CNM          { START | STOP }
(3)
```

Use: MS regions use the VTAM CNM interface to receive both solicited and unsolicited CNM records. These records are processed by the NEWS facility of NetMaster for SNA.

The CNM command is used to open and close the CNM ACB, to enable and disable record receipt from the CNM interface.

Operands:

START

Causes the MS region to attempt to open the CNM ACB. If the open is successful, CNM records can then be received across the CNM interface, and will be processed by NEWS.

STOP

Causes the MS region to close the CNM ACB. No more records will be received across the CNM interface.

Notes: The CNM START command would normally be placed in the NMREADY procedure which is executed during system initialization. This is an alternative to using the System Support Services option Set/Alter News Defaults (option N) to start and stop the CNM interface.

CNMTRACE

Function: Defines which CNM records are to be traced.

```
CNMTRACE    { START | STOP | CANCEL=id }
            [ RU={ rutype | CNMRU | DELIVER | FORWARD |
                   NMVT | RECFMS | RECMS | REQMS |
                   TR-INQ | TR-REPLY | ALIAS } ]
            [ PU=puname ]
            [ ROUTE={ USER | LOG | BOTH } ]
            [ KEEP={ NO | LOG } ]
            [ IN ] [ OUT ]
            [ SOL ] [ UNSOL ]
            [ LOCAL ] [ REMOTE ]
            [ LINK=linkname ]
(3)
```

Use: NEWS can trace CNM RUs that are delivered and received by either the local VTAM CNM Interface or by remote systems using ISR links.

Enter the CNMTRACE command to specify criteria that select particular RUs and to instruct NEWS to deliver a hex display of any RU that satisfies these criteria to the user and/or the activity log.

By default, CNMTRACE records all trace data.

Operands:

START
Starts a NEWS CNM trace.

STOP
Terminates all NEWS CNM traces which were started by the issuing environment.

CANCEL=*id*
Terminates the specified CNM trace request.

RU=*rutype*

Specifies the RU type to trace. It can be numeric or one of the following symbolic types:

CNMRU	x'810814'	CNM RU
DELIVER	x'810812'	DELIVER RU
FORWARD	x'810810'	FORWARD RU
NMVT	x'41038D'	NMVT RU
RECFMS	x'410384'	RECFMS RU
RECMS	x'010381'	RECMS RU
REQMS	x'410304'	REQMS RU
TR-INQ	x'3F0814'	Translate-Inquiry RU
TR-REPLY	x'3F0816'	Translate-Reply RU
ALIAS	x'3F08'	TR-INQ plus TR-REPLY

If RU= is not specified, then all RUs are traced.

PU=*puname*

Specifies the name of the resource to be traced. It can be generic, for example, PU=SDPU* will trace all PUs with names beginning with SDPU.

If PU= is not specified, all PUs will be traced.

ROUTE=USER | LOG | <u>BOTH</u>

Specifies the receiver of the trace results:

Specifying ROUTE=USER sends trace results to issuing environment.

Specifying ROUTE=LOG sends trace results to activity log only.

Specifying ROUTE=BOTH sends trace results to user and activity log.

If ROUTE= is not specified, then the trace results are returned to the user and the activity log.

KEEP=<u>NO</u> | LOG

Determines if the trace request is to be deleted when the initiating environment is deleted:

Specifying KEEP=NO deletes the trace request if the initiating environment is deleted.

Specifying KEEP=LOG means do not delete the trace request if the initiating environment is deleted, but continue tracing and send trace results to the activity log.

If KEEP is not specified, then the trace request is deleted when the initiating environment is deleted.

IN
Traces RUs which are inbound to this system.

OUT
Traces RUs outbound from this system.

If IN and OUT are both omitted, then both inbound and outbound RUs are traced.

SOL
Traces solicited RUs.

UNSOL
Traces unsolicited RUs.

If SOL and UNSOL are both omitted, then both solicited and unsolicited RUs are traced.

LOCAL
Traces local processing of RUs.

REMOTE
Traces remote (ISR) processing of RUs.

If LOCAL and REMOTE are both omitted, then both local and remote RU processing are traced.

LINK=*linkname*
Provides a specific link name to trace. It can be generic for example, LINK=NML*.

If LINK= is not specified, then all links are traced, unless LOCAL is specified and REMOTE is not.

Examples:

```
CNMTRACE IN LINK=NMP*
CNMTRACE RU=NMVT IN REMOTE UNSOL
CNMTRACE SOL PU=NCP*
```

Notes:

Use the CNMTRACE command to perform tracing of CNM RUs which are of special interest. The trace results can be used to determine the source of a CNM record which is causing problems or being interpreted incorrectly.

CONNECT

Function: Connects the specified terminal to your region, by using XNF.

```
CONNECT      xxxxxx
             [ SIZE=(nnn,mmm) ]
             [ DATA=userdata ]
(1)
```

Use: This command connects the specified terminal to your region via XNF. Unless *userdata* is specified, the terminal displays the logon screen.

Operands:

xxxxxx

XNF node name of terminal to be connected.

SIZE=(*nnn,mmm*)

Set a screen size to be used on the connection. The screen size is *nnn* rows by *mmm* columns. Valid screen sizes are in the following ranges:

rows	12 to 255
columns	24 to 255

If no screen size is specified, the default is 24 X 80; that is, SIZE=(24,80).

DATA=*userdata*

Any *userdata* to be used to establish the connection. If this operand is entered, it must be the last parameter, and all text following the operand is treated as *userdata*. Quotes should not be used to enclose the data. Data should be entered in logon *userdata* format, that is, userid, password and initial menu selection.

Examples:

```
CONNECT TERM1 SIZE=(32,80)
CONNECT TERM2 DATA=MYUSER MYPASS
```

Notes: This command applies only to systems using Hitachi's XNF product.

See Also: CLSDST command description.

CS-

Function: Retrieves the command currently stored on the bottom of the command stack and displays it in the command entry area.

```
CS-
(0)
```

Use: During entry of commands while in OCS the system automatically maintains a command stack of the most recently entered commands. The commands can be retrieved from either the top or bottom of this stack and recalled into the command entry area for modification or entry. This facility is provided to assist the operator with a means of rapid reentry of previously used commands.

This command will recall the oldest command from the stack. Repeated entry will result in continued rolling of the stack.

Notes: By default the four most recently entered commands will be maintained in the command stack. The user can alter the size of the command stack using the CMDSTACK operand of the PROFILE command.

The CS- and CS+ commands are normally assigned to function keys, to allow rapid scanning of the command stack to select a required command. Unless your installation has changed the default Function key allocation, the following allocation will be in effect:

```
PF10 CS+
PF11 CS-
PF22 CS+
PF23 CS-
```

If the CS- command is the first command to be entered, then the message `NO COMMANDS ENTERED` is displayed in the command line.

You can also bring any line displayed on your OCS window into the command line for modification or reentry by placing the cursor on the selected line and pressing the ENTER key.

See Also: The PROFILE command CMDSTACK operand.

CS+

Function: Retrieves the command that is currently stored on the top of the command stack and displays it on the command entry line.

```
CS+
(0)
```

Use: During entry of commands while in OCS, the system automatically maintains a command stack of the most recently entered commands. The commands can be retrieved from either the top or bottom of this stack and recalled into the command entry area for modification or entry. This facility is provided to assist the operator with a means of rapid reentry of commands that have been used previously.

This command recalls the most recently entered command from the command stack. Repeated entry results in scrolling of the stack.

Notes: By default, the four most recently entered commands are maintained in the command stack. You can alter the size of the command stack by using the CMDSTACK operand of the PROFILE command.

The CS+ and CS- commands are normally assigned to function keys, to allow rapid scanning of the command stack to select a required command. Unless your installation has changed the default function key allocation, the following allocation is in effect:

```
PF10 CS+
PF11 CS-
PF22 CS+
PF23 CS-
```

If the CS+ command is the first command to be entered, then the message `NO COMMANDS ENTERED` is displayed in the command line.

You can also bring any line displayed on your OCS window into the command line for modification or reentry by placing the cursor on the selected line and pressing the ENTER key.

See Also: The PROFILE command CMDSTACK operand.

D

Function: Displays a specific VTAM resource.

D (0)	{ *name* | *keyword* | NET,ID=*name* }

Use: This command has three basic formats:

- The first format lets you enter a VTAM network node name.

  An example of this is: `D NCP001`

  This is translated into a command with the format: `D NET,ID=NCP001`

- The second lets you enter reserved keywords known to VTAM.

  An example of this is: `D BFRUSE`

  Which is translated into a command with the format: `D NET,BFRUSE`

  The keywords supported are listed below.

- The third format allows entry of normal fully expanded VTAM commands in an unmodified and untranslated format. An example of this is:

  ```
  D NET,NCPSTOR,ADDR=1000,LENGTH=200,ID=NCP001
  ```

Operands:

name

The VTAM name for the resource to be displayed.

keyword

Under ECS/VTAM, *keyword* is one of the following:

APPLS	BFRUSE	CDRMS	CDRSCS
LINES	MAJNODES	PATHS	PATHTAB
PENDING	PUS	LUS	NSAPS

Under VTAM-G, *keyword* is one of the following:

BFRUSE	NCPSTOR	NODESETS	PATHS
PATHTAB			

Under ACF/VTAM, *keyword* is one of the following:

ADJCLUST	ADJCP	ADJSSCPS	APING
APINGDTP	APPLS	APPNTOSA	BFRUSE
BNCOSMAP	CDRMS	CDRSCS	CLSTRS
CNOS	CONVID	COS	CSM
DIRECTRY	DISK	DLURS	EXIT
GROUPS	LINES	LMTBL	LUGROUPS
MAJNODES	MODELS	NCPSTOR	NETSRVR
PATHS	PATHTAB	PENDING	ROUTE
RSCLIST	SATOAPPN	SESSIONS	SNSFILTR
SRCHINFO	STATIONS	STATS	TABLE
TERMS	TGPS	TOPO	TRACES
TRL	TSOUSER	USERVAR	VTAMOPTS
VTAMSTOR			

For a full explanation of these keywords, see the VTAM Operation manual.

NET,ID=*name*
Standard format VTAM command including the name of the resource to be displayed.

Examples:

```
D NCP001
D BFRUSE
D TERM1,E
D CLUS1,I
D MAJNODES
D NET,ID=NCP001
D NET,MAJNODES
```

Notes:

Standard format VTAM commands are supported using the third format described above.

It is possible to allocate function keys for nodes that are regularly displayed, or build an NCL procedure that displays the required resources. Use of conversational and prefix function keys can also be a considerable benefit.

If your installation does not use NET as the name for the VTAM procedure, Management Services must be informed of the procedure name via the VTAMID operand of the SYSPARMS command, described in the *MManagement Services Administrator Guide*.

DEBUG

Function: Controls the execution and debugging of NCL processes.

```
DEBUG       BREAKPOINT
            [ ID=nclid | PROCEDURE=procname ]
            { STMT=number | VERB=verbname |
                { VARS={ name | (name,name,...) } |
                  VARS=prefix* [  RANGE=(start,end) ] |
                  ARGS [ RANGE=(start,end) ] }
               DATA=value |
               EXEC={ ENTRY | EXIT } }

DEBUG       CALLTRC [ ID=nclid | PROCEDURE=procname ]
                          { ON | OFF }

DEBUG       CLEAR [ [ ID=nclid | PROCEDURE=procname ] |
                          BREAKPOINT=number ]

DEBUG       DISPLAY [ ID=nclid ] [ PROCEDURE=procname ]
                    [ VARS={ name | (name,name,...) } |
                      VARS=prefix* [  RANGE=(start,end) | GENERIC ] |
                      ARGS [ RANGE=(start,end) ] |
                      MDO=mdoname | MDONAMES=mdonames ]
                    [ FORMAT={ CHAR | HEX | DUMP } ]

DEBUG       HOLD [ ID=nclid | PROCEDURE=procname ]

DEBUG       LIST BREAKPOINT
            [ ID=nclid | PROCEDURE=procname ]

DEBUG       MODIFY [ ID=nclid | PROCEDURE=procname ]
            [ OPT={ SETERR | RESETERR |
              SETMOD | RESETMOD |
              SETOUT | RESETOUT } ]
            [ VARS={ name | (name,name,...) } |
               VARS=prefix* [  RANGE=(start,end) | GENERIC ] |
               ARGS [ RANGE=(start,end) ] |
               MDO=mdonames ]
            [ DATA=value ]

DEBUG       RESUME [ ID=nclid | PROCEDURE=procname ]

DEBUG       SET [ NEWHOLD={ YES | NO } ]
                [ NCLTRACE={ YES | NO } ]
                [ CALLTRC={ YES | NO } ]
                [ LOG={ YES | NO } ]

DEBUG       SOURCE [ ID=nclid | PROCEDURE=procname ]
                    [ STMT=number ] [ FW=nnnn ] BW=nnnn ]DEBUG
```

```
DEBUG        START [ DEBUGID=dbid | ID=nclid |
                     [ USER=userid ]
                     [ REGION=luname ]
                     [ WINDOW=number ]
                     [ PROCEDURE=procname ]
                       KEEP={ YES | NO } ]


DEBUG        STEP [ ID=nclid | PROCEDURE=procname ]
                  [ NEXT={ nnnn | 1 | ENTRY | EXIT } |
                  [ TOPROCEDURE=procname ] [ TOSTMT=number ] ]

DEBUG        STOP [ ID=nclid | PROCEDURE=procname ]
                  [ TYPE={ CONTINUE | FLUSH | SUSPEND } ]

DEBUG        TRACE [ ID=nclid | PROCEDURE=procname ]
(0/3)
```

Use: To debug one or more NCL processes.

Operands:

BREAKPOINT
Sets a trap on either a particular statement, verb or variable of a procedure. When the trap is triggered then the NCL process's execution is suspended.

CALLTRC
Invokes procedure call tracing.

CLEAR
Clears all breakpoints or selected breakpoints from a debug session.

DISPLAY
Displays the contents of selected variables from any procedure nesting level within an NCL process.

HOLD
Flags the NCL process identified to be stopped on the next statement. The process is stopped as soon it attempts to execute the next statement in the procedure. The process is held in a suspended state.

LIST BREAKPOINT
Lists all breakpoints or selected breakpoints from a debug session.

MODIFY
Modifies the contents of selected variables from any procedure nesting level within an NCL process.

DEBUG

RESUME

Makes the NCL process that is affected by this command eligible for execution from the point at which it was suspended.

SET

Sets the profile of the debug session and displays the settings.

SOURCE

Lists the selected statements of the specified NCL process.

START

Starts a debug session and specifies a scope, within which all NCL processes will be placed under the control of the NCL debugger.

STEP

Makes the NCL process eligible for execution and sets a statement counter. When the specified number of statements have executed then the process is suspended again. If the ENTRY or EXIT options are used, the process continues until the next EXEC command (for ENTRY option) or the next procedure termination (for EXIT option). If the TOSTMT operand is used, the process is suspended when the specified statement in the specified procedure is encountered.

STOP

Terminates the debug session and all processes under its control are either flushed, suspended or allowed to continue processing.

TRACE

Displays all procedure and subroutine nesting levels for the identified NCL process.

Examples:

```
DEBUG START WINDOW=2
DEBUG SET NEWHOLD=YES NCLTRACE=YES
DEBUG BREAKPOINT PROCEDURE=MYPROC STMT=1250000
DEBUG DISPLAY PROCEDURE=MYPROC VARS=PANELNAM
DEBUG MODIFY PROCEDURE=MYPROC VARS=PANELNAM DATA=MYNEWPAN
DEBUG STEP PROCEDURE=MYPROC NEXT=15
DEBUG RESUME PROCEDURE=MYPROC
DEBUG HOLD PROCEDURE=MYPROC
DEBUG TRACE PROCEDURE=MYPROC
DEBUG SOURCE PROCEDURE=MYPROC BW=10 FW=10
DEBUG LIST BREAKPOINTS PROCEDURE=MYPROC
DEBUG CLEAR
```

Notes:

A debug session is associated with a particular region that is referred to as the *debugger*. This region is the only region from which debug commands will be accepted for the session. Any command environment can be a debugger, this includes OCS and any &INTCMD environment.

An NCL process that falls into the scope of a debugger is associated with that debugger's environment and is referred to as a *debugged* NCL process. An NCL process can only be debugged by one debugger at a time.

See Also: Detailed descriptions of each individual command.

The SHOW DEBUG command description.

DEBUG BREAKPOINT

Function: Sets a breakpoint at a particular point within a procedure.

```
DEBUG     BREAKPOINT
          [ ID=nclid | PROCEDURE=procname ]
          { STMT=number | VERB=verbname |
              { VARS={ name | (name,name,...) } |
                VARS=prefix* [ RANGE=(start,end) ] |
                ARGS [ RANGE=(start,end) ] }
            DATA=value ] |
            EXEC={ ENTRY | EXIT } }
(0)
```

Use: Sets a trap on either a particular statement, verb or variable of a procedure. When the trap is triggered, the execution of the NCL process is suspended.

This command is only valid when a debug session has been established. If the breakpoint defines an NCL ID, that process must exist and be under the control of the command issuer's debug session.

When a procedure is executed, the debugger will check if there are any statement breakpoints that are eligible for it. If so, the procedure is loaded as a private copy and the breakpoints are applied to it. If not, normal procedure loading is adhered to.

Operands:

ID=*nclid*

The breakpoint is defined as only being eligible to be set to procedures that are executed by the specified NCL process. Any other NCL process executing under the control of the debug session is not affected.

If the breakpoint defines an NCL ID, then that process must exist and be under the control of the command issuer's debug session. When the process terminates or when a DEBUG STOP ID= command is issued, any process dependent breakpoints will be cleared.

PROCEDURE=*procname*

The breakpoint is defined as only being eligible to be set to occurrences of the specified procedure name. If the ID operand is also specified then the breakpoint is further restricted to occurrences of the procedure within the particular NCL process. If a STMT breakpoint is defined then this operand is required.

STMT=*number* | VERB=*verbname* |
{ VARS={ *name* | (*name,name,...*) } |
VARS=*prefix [RANGE=(*start,end*)] |**
ARGS [RANGE=(*start,end*)] } |
[DATA=*value*] | EXEC={ ENTRY | EXIT }

These operands define the actual breakpoint that is to be set to procedures within processes. There are five types of breakpoints. The STMT breakpoint defines a trap be set to a particular line number. VERB sets the breakpoint to be actioned on the next execution of the specified verb. The VARS operand allows breakpoints to be set on the action of updating a variable (and conditionally updating the variable with a particular value). The EXEC=ENTRY operand specifies that the procedure should be suspended as it begins execution. The EXEC=EXIT operand specifies that the procedure should be suspended when it completes processing normally.

STMT=*number*

This operand specifies the line number of a statement at which execution of the procedure is to be suspended. The process is suspended immediately before the statement is executed. If this type of breakpoint is defined then the PROCEDURE operand is also required.

The format of the line number is an integer between 1 and 99999999. For procedures that are numbered, use the entire eight byte line number as shown in columns 73 to 80 of the source (leading zeros can be omitted). For unnumbered procedures, use a relative line number. If a procedure has a mixture of numbers and no-numbers, use which ever format exists for the particular line being trapped.

VERB=*verbname*

This operand specifies a verb that is to cause the process to be suspended. The process is suspended immediately before the specified verb is executed.

{ VARS={ *name* | (*name,name,...*) } |
VARS=*prefix [RANGE=(*start,end*)] |**
ARGS [RANGE=(*start,end*)] } |
[DATA=*value*]

This operand specifies one or more variables that are to cause the process to be suspended if they are updated. Once any one of the variables has been updated the process is suspended. The DATA operand allows a conditional breakpoint to be set. The process is suspended if a nominated variable is updated to the value specified in the DATA operand.

VARS={ *name* | (*name,name,...*)

This operand allows you to specify one or more variables by name. If only one variable is specified, then it does not need to be enclosed in parentheses.

VARS=*prefix [RANGE=(*start,end*)]**

This operand allows you to specify a range of variables by specifying a prefix. If the RANGE option is used or allowed to default, then variables with the specified prefix and a suffix of integers within the start and end range are targeted for breakpointing.

DEBUG BREAKPOINT

ARGS [RANGE=(*start,end*)]

This operand allows you to specify the standard argument variables (&1, &2 ... &*n*). The RANGE option indicates the first and last variable numbers.

DATA=*value*

This operand allows you to specify the value to which a variable must be set, to cause the breakpoint to be honored. Specify the data in one of the following formats:

DATA=abcd (character value)
DATA=C'abcd' (character value)
DATA=X'81828384' (hexadecimal value)

If the DATA operand is omitted, then the process is suspended when any of the variables is updated with any new value. If a null value is specified for the DATA operand, then the breakpoint is triggered if a variable is deleted (that is,is set to null).

EXEC={ ENTRY | EXIT }

The EXEC operand specifies that a procedure is to be suspended when it begins execution (EXEC=ENTRY) or completes execution normally (EXEC=EXIT). If the procedure terminates abnormally due to error, an EXEC= breakpoint will be ignored, but procedure execution will be suspended due to the error.

Examples:

```
DEBUG BREAKPOINT PROCEDURE=MYPROC VERB=INTREAD
DEBUG BREAKPOINT ID=2364 PROCEDURE=MYPROC STMT=1720000
DEBUG BREAKPOINT VARS=(PANSTAT,INTRST)
DEBUG BREAKPOINT VARS=PANLN* DATA=$   ** DELETED **
DEBUG BREAKPOINT PROCEDURE=XXPRC01 ARGS RANGE=(5,9)
DEBUG BREAKPOINT PROCEDURE=XXPRC02 EXEC=ENTRY
```

Notes:

If both ID and PROCEDURE are omitted, the defined breakpoint is eligible to be set to any procedure of any NCL process within the debug session.

See Also:

DEBUG, DEBUG HOLD, DEBUG STEP, and DEBUG RESUME command definitions.

DEBUG CALLTRC

Function: Invokes procedure call tracing.

```
DEBUG     CALLTRC
          [ ID=nclid | PROCEDURE=procname ]
          { ON | OFF }
(0)
```

Use: This command is used to invoke tracing of nested procedure calls. A trace message is produced indicating that a change in nesting levels has occurred. This change in nesting levels could be due to a procedure calling another procedure or returning from a nested procedure call.

Operands:

ID=*nclid*
Specifies the NCLID of the NCL process to be the target of the command.

PROCEDURE=*procname*
Specifies the BASE or CURRENT procedure in an NCL process which is to be the target of the command.

ON | OFF
DEBUG CALLTRC ON causes call tracing to begin.
DEBUG CALLTRC OFF causes call tracing to terminate.

Examples:

```
DEBUG CALLTRC ON ID=123456
DEBUG CALLTRC OFF
```

Notes: If ID and PROCEDURE are both omitted, then the last referenced NCL process is the target of the DEBUG CALLTRC command.

DEBUG CLEAR

Function: Clears one or more breakpoints.

```
DEBUG     CLEAR
          [ [ ID=nclid | PROCEDURE=procname ] |
              BREAKPOINT=number ]
(0)
```

Use: Clears all breakpoints or selected breakpoints from a debug session.

Operands:

ID=*nclid*

The breakpoints defined for the specified NCL ID are to be cleared. If no other operand is specified then all the breakpoints defined with the ID operand specified are cleared.

PROCEDURE=*procname*

The breakpoints defined for the specified *procname* are cleared. If the ID operand is also specified, only breakpoints defined for the specified procedure and the specified NCL ID are cleared. If only the PROCEDURE operand is specified, all breakpoints that specify that procedure name are cleared.

BREAKPOINT=*number*

Specifies the particular breakpoint to be cleared. Each breakpoint is allocated a number when it is defined and it can be used to refer to a particular breakpoint definition. The DEBUG LIST BREAKPOINT command can be used to list all breakpoints defined.

Examples:

```
DEBUG CLEAR
DEBUG CLEAR BREAKPOINT=13
DEBUG CLEAR ID=1235
DEBUG CLEAR ID=378 PROCEDURE=MYPROC
DEBUG CLEAR PROCEDURE=MYPROC
```

Notes: If no operands are specified then all breakpoints defined for the debug session are cleared.

See Also: DEBUG LIST BREAKPOINT and DEBUG BREAKPOINT command descriptions.

DEBUG DISPLAY

Function: Displays the contents of variables for a procedure in an NCL process that is being debugged.

```
DEBUG     DISPLAY
          [ ID=nclid ] [ PROCEDURE=procname ]
          [ VARS={ name | (name,name,...) } |
            VARS=prefix* [ RANGE=(start,end) | GENERIC ] |
            ARGS [ RANGE=(start,end) ] |
            MDO=mdoname | MDONAMES=mdonames ]
          [ FORMAT={ CHAR | HEX | DUMP } ]
(0)
```

Use: Displays the contents of selected variables from any procedure nesting level within an NCL process.

Operands:

ID=*nclid*

The NCL process identified by the ID operand is used to obtain the variable values. The process must be under the control of the debug session owned by the command issuer's environment.

PROCEDURE=*procname*

The variables are listed from the specified procedure. If the ID operand is also specified, the procedure is selected from the process identified. If the ID operand is not specified, the procedure is selected from the first process in which the specified procedure is found.

VARS={ *name* | (*name,name,...*) } |
VARS=*prefix [RANGE=(*start,end*) | GENERIC] |**
ARGS [RANGE=(*start,end*)] |
MDO=*mdoname* | MDONAMES=*mdonames*

These operands allow you to specify one or more variables or MDOs to be displayed. If omitted, all variables for the selected process and procedure are displayed.

VARS={ *name* | (*name,name,...*) }

This operand allows you to specify one or more variables by name. If only one variable is specified, then the parentheses are not required.

VARS=*prefix [RANGE=(*start,end*) | GENERIC]**

This operand allows you to specify a range of variables by specifying a prefix. If the RANGE option is used, or allowed to default, variables with the specified prefix and a suffix of integers (within the start and end range) are displayed. If GENERIC is specified, then any variable with the prefix is displayed, regardless of the suffix.

ARGS [RANGE=(*start,end*)]

This operand allows you to specify the standard argument variables (&1, &2, ... &*n*). The RANGE option indicates the variable start and end numbers.

DEBUG DISPLAY

M**DO=*mdoname* | MDONAMES=*mdonames***

This operand specifies the name of one or more MDOs to be displayed. For MDO=, the MAP name, MDO name and data for each MDO are displayed. For MDONAMES=, only the MAP name and MDO name are displayed.

The MDO name can be entered as a simple, compound, generic or variable indexed name.
MDO=*
MDO=AA*
MDO=AAAA
MDO=AAAA.*
MDO=AAAA.BB*
MDO=AAAA.BBBB
MDO=AAAA.BBBB{2}
MDO=AAAA.BBBB{*} RANGE=(1,5)

FORMAT={ <u>CHAR</u> | HEX | DUMP }

This operand specifies the display format for data:

CHAR	Data is shown in character format.
HEX	Data is shown in hexadecimal format.
DUMP	Data is shown in hexadecimal and character dump format.

This operand can be abbreviated to FMT. The FORMAT operand is not valid if the MDONAMES operand is entered.

Examples:

```
DEBUG DISPLAY ID=215 PROCEDURE=MYSPROC VARS=(LVAR,COUNTVAR)
DEBUG DISPLAY PROCEDURE=PROCA01 VARS=PANLN* RANGE=(5,36)
DEBUG DISPLAY VARS=TXV* GENERIC FORMAT=HEX
DEBUG DISPLAY MDO=* FORMAT=DUMP
DEBUG DISPLAY MDONAMES=AA.*
```

Notes: If the ID and PROCEDURE operands are both omitted, then the last referenced NCL process is the target of the command.

See Also: DEBUG MODIFY command description.

DEBUG HOLD

Function: Suspends the execution of an NCL process that is under the control of a debug session.

```
DEBUG     HOLD
          [ ID=nclid | PROCEDURE=procname ]
(0)
```

Use: The NCL process identified is flagged to be stopped on the next statement. The process is stopped as soon it attempts to execute the next statement in the procedure. It is held in a suspended state until either a DEBUG RESUME, DEBUG STEP or a DEBUG STOP TYPE=CONTINUE command is entered.

Operands:

ID=*nclid*

The NCL process identified by the ID operand is suspended. The process must be under the control of the debug session owned by the command issuer's environment.

PROCEDURE=*procname*

The NCL process whose base procedure name or current procedure name matches this operand is suspended. There must be only one process currently under the control of the debug session that has a base or current procedure name as specified by this operand. If more than one exists, use the ID operand. The process must be under the control of the debug session owned by the command issuer's environment.

Examples:

```
DEBUG HOLD ID=215
DEBUG HOLD PROCEDURE=MYPROC
```

Notes: If the ID and PROCEDURE operands are both omitted, the last referenced NCL process will be the target of the DEBUG HOLD command.

See Also: DEBUG, DEBUG RESUME, DEBUG STOP, DEBUG STEP, and DEBUG BREAKPOINT command descriptions.

DEBUG LIST BREAKPOINT

Function: Lists the breakpoints defined for a debug session.

```
DEBUG     LIST  BREAKPOINT
          [ ID=nclid | PROCEDURE=procname ]
(0)
```

Use: Lists all breakpoints or selected breakpoints from a debug session.

Operands:

ID=*nclid*

The breakpoints defined for the specified NCL ID are to be listed. If no other operand is specified, all the breakpoints defined with the ID operand specified are listed.

PROCEDURE=*procname*

The breakpoints defined for the specific procedure specified are listed. If the ID operand is also specified, only breakpoints defined for the specified procedure and the specified NCL ID are listed. If only the PROCEDURE operand is specified, all breakpoints that specify that procedure name are listed.

Examples:

```
DEBUG LIST BREAKPOINT
DEBUG LIST BREAKPOINT ID=1223
DEBUG LIST BREAKPOINT ID=378 PROCEDURE=MYPROC
DEBUG LIST BREAKPOINT PROCEDURE=MYPROC
```

Notes: If no operands are specified, all breakpoints defined for the debug session are listed.

See Also: DEBUG CLEAR and DEBUG BREAKPOINT command definitions.

DEBUG MODIFY

Function: Modifies the contents of variables or MDOs for a procedure in an NCL process that is being debugged.

```
DEBUG     MODIFY
          [ ID=nclid | PROCEDURE=procname ]
          [ OPT={ SETERR | RESETERR |
            SETMOD | RESETMOD |
            SETOUT | RESETOUT } ]
          [ VARS={ name | (name,name,...) } |
            VARS=prefix* [ RANGE=(start,end) | GENERIC ] |
            ARGS [ RANGE=(start,end) ] |
            MDO=mdoname ]
          [ DATA=value ]
(0)
```

Use: Modifies the contents of selected variables from any procedure nesting level within an NCL process.

Operands:

ID=*nclid*

The NCL process identified by the ID operand is used to obtain the variable values. The process must be under the control of the debug session owned by the command issuer's environment.

PROCEDURE=*procname*

The variables are modified in the specified procedure. If the ID operand is also specified, the procedure is selected from the process identified. If the ID operand is not specified, the procedure is selected from the first process found to contain the specified procedure.

OPT={ SETERR | RESETERR | SETMOD | RESETMOD | SETOUT | RESETOUT }

This operand allows you to modify the attributes of the variables identified.

VARS={ *name* | (*name,name,...*) } |
VARS=*prefix [RANGE=(*start,end*) | GENERIC] |**
ARGS [RANGE=(*start,end*)] | MDO=*mdoname*

These operands allow you to specify one or more particular variables to be modified. If omitted, all variables for the selected process and procedure are modified.

VARS={ *name* | (*name,name,...*) }

This operand allows you to specify one or more variables by name. If only one variable is specified, the parentheses are not required.

VARS=*prefix [RANGE=(*start,end*) | GENERIC]**

This operand allows you to specify a range of variables by specifying a prefix. If the RANGE option is used, or allowed to default, variables with the specified prefix and a suffix of integers (within the start and end range) are modified. If GENERIC is specified, any variable with the prefix is modified, regardless of the suffix.

DEBUG MODIFY

ARGS [RANGE=(*start,end*)]

This operand allows you to specify the standard argument variables (&1, &2 ... &*n*). The RANGE option indicates the variable start and end numbers.

MDO=*mdoname*

This operand specifies the entire data section of an MDO as the target of the modify command. The MDO name can be entered as a simple or compound name. For example:
MDO=AAAA
MDO=AAAA.BBBB
MDO=AAAA.BBBB{2}

DATA=*value*

This operand allows you to specify the value to which each variable identified is to be set to. Specify the data in one of the following formats:

DATA=abcd	(Character value)
DATA=C'abcd'	(Character value)
DATA=X'81828384'	(Hexadecimal value)

If the value is omitted, the identifying variables are deleted (that is, set to null).

Examples:

```
DEBUG MODIFY ID=215 PROCEDURE=MYSPROC VARS=(LVAR,COUNTVAR)
DEBUG MODIFY VARS=PANLN* RANGE=(5,7) DATA=$    ** DELETED **
DEBUG MODIFY VARS=EEQMC2* GENERIC DATA=YAAAHHHHOOOOO
DEBUG MODIFY MDO=MYMDO DATA=X'0123283FA0B21C4F56B0'
```

Notes:

If the ID and PROCEDURE operands are both omitted, then the last referenced NCL process is the target of the DEBUG MODIFY command.

See Also:

DEBUG DISPLAY, DEBUG TRACE, and DEBUG SOURCE command descriptions.

DEBUG RESUME

Function: Resumes execution of an NCL process that was suspended by a debug command or as a result of a breakpoint.

```
DEBUG     RESUME
          [ ID=nclid | PROCEDURE=procname ]
(0)
```

Use: The NCL process that is affected by this command is made eligible for execution at the statement on which the process was suspended.

The process can be suspended as a result of any of the following conditions:

- A breakpoint was reached
- A DEBUG HOLD or DEBUG STEP command was issued
- The process started and the current profile setting for the debug session was NEWHOLD=YES
- Execution of the process failed, for example, due to a syntax error

Operands:

ID=*nclid*

The NCL process identified by the ID operand is resumed. The process must be under the control of the debug session owned by the command issuer's environment.

PROCEDURE=*procname*

The NCL process whose base procedure name or current procedure name matches this operand is resumed. There must be only one process currently under the control of the debug session that has this base or current procedure name. If more than one exists, use the ID operand. The process must be under the control of the debug session owned by the command issuer's environment.

Examples:

```
DEBUG RESUME ID=6238
DEBUG RESUME PROCEDURE=MYPROC
```

Notes: If the ID and PROCEDURE operands are both omitted, then the last referenced NCL process is the target of the DEBUG RESUME command.

See Also: DEBUG, DEBUG HOLD, DEBUG STEP, and DEBUG BREAKPOINT command definitions.

DEBUG SET

Function: Sets profile options for a debug session.

```
DEBUG     SET
          [ NEWHOLD={ YES | NO } ]
          [ NCLTRACE={ YES | NO } ]
          [ CALLTRC={ YES | NO } ]
          [ LOG={ YES | NO } ]
(0)
```

Use: The debug session has its profile set and the settings are displayed in a message.

Operands:

NEWHOLD={ YES | NO }

The NEWHOLD=YES setting specifies that any NCL process placed under the control of this debug session is to be immediately suspended on the first statement.

If NEWHOLD=NO is in effect, any new processes execute as normal and are suspended only if a breakpoint is reached or a command to suspend the process is issued.

NCLTRACE={ YES | NO }

If a process being debugged has statement tracing in effect, specifying NCLTRACE=YES causes the debugger to send the trace messages to the debug session owner, as well as the process owner's environment.

Specifying NCLTRACE=NO stops trace messages being sent to the debug session owner's environment.

CALLTRC={ YES | NO }

CALLTRC=YES specifies that all NCL processes under the control of this debug session are to produce a *procedure call trace*. The trace messages are:

- N04023 - nested procedure call message
- N04024 - nested procedure return message

LOG={ YES | NO }

By default, all messages issued as a result of a process being suspended, resumed, activated under debug, or removed from debug control, are not logged. If LOG=YES is specified, then these messages are sent to the activity log, as well as being sent to the debug session owner.

Examples:

```
DEBUG SET NEWHOLD=YES
DEBUG SET NEWHOLD=YES NCLTRACE=YES
```

DEBUG SOURCE

Function: Displays source statements for a currently executing NCL process, as stored in the system.

```
DEBUG     SOURCE
          [ ID=nclid | PROCEDURE=procname ]
          [ STMT=number ] [ FW=nnnn ] [ BW=nnnn ]
(0)
```

Use: The selected statements of the specified NCL process are listed.

Operands:

ID=*nclid*

The NCL process identified by the ID operand is listed. The process must be under the control of the debug session owned by the command issuer's environment.

PROCEDURE=*procname*

The procedure specified by this operand is listed. If the ID operand is also specified, the procedure is listed from the process identified. If the ID operand is not specified, the procedure is listed from the first process found to contain the specified procedure.

STMT=*number*

This operand specifies the starting point for the list. If omitted, the current NCL statement for the identified procedure is used.

The format of the line number is an integer between 1 and 99999999. For procedures that are numbered, use the entire eight byte line number as shown in columns 73 to 80 of the procedure (leading zeros can be omitted). For unnumbered procedures, use a relative line number. If a procedure has a mixture of numbers and no-numbers, use which ever format exists for the particular line being listed.

For statements that span multiple lines, use the line number for the first line of the statement. If the line number doesn't exist, the first line with a line number greater than the number specified is used.

FW=*nnnn*

This operand specifies the number of statements to be listed in the forward direction. The list starts from the statement identified by the STMT operand. Specify a number between 1 and 9999.

BW=*nnnn*

This operand specifies the number of statements to be listed in the backward direction. The list starts from the statement identified by the STMT operand. Specify a number between 1 and 9999.

Examples:

```
DEBUG SOURCE PROCEDURE=MYPROC FW=5
DEBUG SOURCE
DEBUG SOURCE ID=2364 STMT=720000 BW=5
DEBUG SOURCE ID=2672 PROCEDURE=OURPROC STMT=125
```

DEBUG SOURCE

Notes: If the FW and BW operands are both omitted, only the statement identified by the STMT operand is displayed.

If the ID and PROCEDURE operands are both omitted, then the last referenced NCL process is the target of the DEBUG SOURCE command.

See Also: DEBUG TRACE command description.

DEBUG START

Function: Establishes an NCL debugging environment and scope.

```
DEBUG     START
          [ DEBUGID=dbid | ID=nclid |
          [ USER=userid ]
          [ REGION=luname ]
          [ WINDOW=number ]
          [ PROCEDURE=procname ]
            KEEP={ YES | NO } ]
(0/3)
```

Use: Starts a debug session and specifies a scope within which all NCL processes are placed under the control of the NCL debugger.

Operands:

DEBUGID=*dbid*

The DEBUGID operand is specified to reconnect to a previously suspended debug session. Use the SHOW DEBUG command to obtain a list of current debug sessions.

> **Note**
> If DEBUGID is specified, then no other operand can be specified.

ID=*nclid*

The ID operand is used to target a specific NCL process and its dependent processing environment. If specified, the process must be currently executing.

> **Note**
> If ID is specified, then no other operand can be specified.

USER=*userid*

The USER operand specifies the user ID to define the scope of the debug session. It causes all the NCL processes of this user, currently started, or that will start in the future, to be the target of this debug session. It includes all NCL processing regions of all the user's signed on regions for the specified user ID, unless other operands are used in conjunction with this one to restrict the scope.

If the WINDOW operand is specified, the USER will default to the command issuer's user ID. If the REGION operand is also specified, USER will default to that region's user ID. This operand can be specified with the REGION, WINDOW or PROCEDURE operands to control the granularity of the scope.

REGION=*luname*

The REGION operand specifies a user's logon region to define the scope of the debug session. It causes all NCL processes for the logged on region specified, currently started or that will start in the future, to be the target of this debug session. It includes all NCL processing regions for the specified user logon region, unless other operands are used in conjunction with this one to restrict the scope.

If the WINDOW operand is specified, REGION will default to the command issuer's logon region. If USER is specified, REGION will default to the logon region of that user, if only one exits. This operand can be specified with the USER, WINDOW or PROCEDURE operands to control the granularity of the scope.

If the region being debugged is a ROF session, specify the linkname of the ROF session as the region.

WINDOW=*number*

The WINDOW operand specifies a particular user's logged on region and processing window to define the scope of the debug session. It causes all NCL processes for the window specified, currently started or that will start in the future, to be the target of this debug session. It includes all NCL processing regions in the specified window for the specified logged on region, unless other operands are used with this one to restrict the scope.

If other scope operands are specified and WINDOW is not, all the windows of the defined scope will be the target of this debug session. This operand can be specified with the USER, REGION or PROCEDURE operands to control the granularity of the scope. If this operand is specified, the USER and REGION operands will default to the issuer's user ID and region.

PROCEDURE=*procname*

Only instances of the named procedure(s) will be the target of the debug session. If specified in conjunction with other scope operands, the scope is reduced to only occurrences of the procedure within the scope that is defined by the other operands.

If no other scope operands are specified, the instances of the specified procedure within the SAME NCL processing region from where the command was sourced are the target of the debugger. For example if the command was sourced from OCS, any occurrence of the procedure within any NCL process running in that OCS window, or in any of those NCL processes dependent processing regions is targeted. If, however, the command was sourced from an &INTCMD environment, instances of that procedure within any NCL process that is also running in that same &INTCMD environment (or within one of their own dependent processing regions) is targeted.

KEEP={ YES | <u>NO</u> }

If KEEP=NO is specified, or allowed to default, the scope specified must define an active environment. For example, the targeted user must be signed on to the targeted region if both USER and REGION were specified. When the target scope terminates the debug scope is automatically terminated.

If KEEP=YES is specified, the scope specification does not need to define an active environment. If the environment targeted terminates, the debug scope continues to exist and will take effect when the environment restarts

This operand is not allowed by itself or when PROCEDURE is the only scope operand. It is not allowed in conjunction with DEBUGID or the ID operands.

Examples:

```
DEBUG START WINDOW=2
DEBUG START USER=U01829 REGION=VA032 KEEP=YES
DEBUG START USER=AALZ038 PROCEDURE=USRMENU
DEBUG START REGION=LU023R1 KEEP=YES
```

Notes:

If no operands are specified, all NCL processes within the SAME NCL processing region from where the command was sourced are the target of the debug. For example, if the command was sourced from OCS, any NCL process running in that OCS window or in any of those NCL processes dependent processing regions is targeted. If however, the command was sourced from an &INTCMD environment, all NCL processes that are running in that same &INTCMD environment (or within one of their dependent processing regions) are targeted.

The command issuer must have OPAUTH authority and SYSPARMS NCLXUSER=YES must be set, to debug other user's NCL processing regions.

When the command issuer's environment terminates and no specific DEBUG STOP command has been entered, all debug scopes that the environment owns are automatically terminated (the debugged processes will have all breakpoints removed and any suspended process will be resumed). Thus if the debug commands are issued in an &INTCMD environment, the debug session will be stopped when the owning NCL process terminates.

Every debug scope must be unique, however one or more scopes can intersect. If an NCL process starts and it is within the scope of one or more debug sessions, the debug scope used is selected on a best fit basis. If the processing region in which the process is running has a scope of that environment, it is considered the best fit. If the process is executing in the dependent processing environment of a debugged process, that processes debug scope is used. Otherwise the scope is chosen based on window, region and user ID, in that order.

Multiple DEBUG START commands specifying different scopes, can be issued from the same environment. Each scope is added to the environment's debug session.

A debug session is tied to the environment that issued the DEBUG START command. Only commands from this environment will be accepted against an NCL process being debugged by this environment. If the environment that owns the debug session terminates then debug session is stopped as if TYPE=CONTINUE were specified on a DEBUG STOP command.

See Also:

DEBUG STOP and SHOW DEBUG command descriptions.

DEBUG STEP

Function: Resumes execution of an NCL process for a specified number of statements or until procedure entry or exit occurs, and then suspends it.

```
DEBUG     STEP
          [ ID=nclid | PROCEDURE=procname ]
          [ NEXT={ nnnn | 1 | ENTRY | EXIT } |
          [ TOPROCEDURE=procname ] [ TOSTMT=number ] ]
(0)
```

Use: The NCL process resumes execution but will be suspended when the specified number of statements have been executed, or when procedure entry or exit occurs, or when the specified statement within the specified procedure is encountered.

Operands:

ID=*nclid*

The NCL process identified by the ID operand is stepped. The process must be under the control of the debug session owned by the command issuer's environment.

PROCEDURE=*procname*

The PROCEDURE= operand specifies the base or current procedure for the NCL process to be stepped. There should be only one process currently under the control of the debug session that has the base or current procedure name specified. If more than one exists, use the ID operand. The process must be under the control of the debug session owned by the command issuer's environment.

NEXT={ *nnnn* | 1 | ENTRY | EXIT }

The NEXT operand specifies when execution of the NCL process should be suspended. NEXT=1 or NEXT=*nnnn* specifies that execution of the NCL process should be suspended after 1 or *nnnn* statements have been executed. NEXT=ENTRY causes the NCL process to be suspended when the next nested execution of an NCL procedure is performed via the EXEC or FSPROC command. NEXT=EXIT causes execution of the NCL process to be suspended when the next return to a higher nesting level occurs.

TOPROCEDURE=*procname*

The TOPROCEDURE= operand specifies the name of the procedure where the TOSTMT is to take effect. If the operand is not specified, the current procedure name is used as the default.

TOSTMT=*number*

The TOSTMT operand specifies the statement number at which execution is to stop. This allows the process to execute until the specified statement is encountered within the current procedure, regardless of the nesting level. The TOPROCEDURE operand can also be specified to change the procedure name to another currently executing procedure or one that can be executed in the future.

Examples:

```
DEBUG STEP
DEBUG STEP ID=215 NEXT=20
DEBUG STEP PROCEDURE=MYPROC NEXT=2
DEBUG STEP NEXT=ENTRY
DEBUG STEP PROCEDURE=MYPROC NEXT=EXIT
DEBUG STEP PROCEDURE=MYPROC TOSTMT=1240000
DEBUG STEP PROCEDURE=MYPROC TOPROCEDURE=XPROC TOSTMT=230
```

Notes:

If the ID and PROCEDURE operands are both omitted, the last referenced NCL process will be the target of the DEBUG STEP command.

If the process is suspended for any other reason, before the specified number of statements have executed or before the procedure ENTRY or EXIT occurs, the NCL process is suspended and the DEBUG STEP command is no longer in effect.

See Also:

DEBUG, DEBUG HOLD, DEBUG RESUME, and DEBUG BREAKPOINT command descriptions.

DEBUG STOP

Function: Terminates or suspends an NCL debug session.

```
DEBUG     STOP
          [ ID=nclid | PROCEDURE=procname ]
          [ TYPE={ CONTINUE | FLUSH | SUSPEND } ]
(0)
```

Use: The debug session is terminated and all processes under the debug session's control are either flushed, suspended or allowed to continue processing.

If the ID or PROCEDURE operands are specified then the NCL process identified is removed from the debug session's control and the debug session remains active (no other processes are affected).

Operands:

ID=*nclid*

The NCL process identified by the ID operand is removed from the debugger's control and any ID specific breakpoints are cleared. The process must be under the control of the debug session owned by the command issuer's environment.

PROCEDURE=*procname*

The NCL process whose base procedure name or current procedure name matches this operand is removed from the debugger's control and any breakpoints specific to that process are cleared. There must be only one process currently under the control of the debug session that has the base or current procedure name specified. If more than one exists, use the ID operand. The process must be under the control of the debug session owned by the command issuer's environment.

TYPE={ CONTINUE |FLUSH | SUSPEND }

The TYPE operand specifies the action to be taken against the actual NCL processes that have been placed under the debugger's control.

TYPE=CONTINUE

All NCL processes that are under the control of this debug session are disconnected from this debug session and allowed to continue processing. All breakpoints are removed and any processes that were suspended are resumed.

TYPE=FLUSH

All NCL processes that were placed under this debug session's control are flushed.

TYPE=SUSPEND

The NCL processes that were placed under this debug session's control are allowed to continue as if no DEBUG STOP command had been issued. All breakpoints are left intact, any suspended processes are left suspended and any executing processes are left to continue execution until a breakpoint is reached. Any new processes that fall within the scopes of this debug session are placed under its control and any breakpoints defined are applied to the process. If NEWHOLD=YES is in effect for this debug session, then any new process started is automatically suspended on the first statement. The debug session is not actually terminated, but is disconnected from the current environment. A DEBUGID is assigned to the session, which can be reconnected by this or another environment using the DEBUG START DEBUGID=*dbid* command.

If TYPE=SUSPEND is specified, then the actual debug session is suspended. No further debug commands can be entered for this debug session until a DEBUG START has been issued specifying the DEBUGID assigned by the DEBUG STOP command.

For the other TYPE values the debug session is terminated and no further debug commands can be entered for this session.

Examples:

```
DEBUG STOP
DEBUG STOP TYPE=FLUSH
DEBUG STOP TYPE=SUSPEND
```

Notes:

When the command issuer's environment terminates and no specific DEBUG STOP command has been entered, the debug session that the environment owns is automatically stopped with TYPE=CONTINUE specified. Thus if the debug commands are issued in an &INTCMD environment, the debug session will be stopped when the owning NCL process terminates.

When an individual process is removed from the debugger's control, all breakpoints that had specified that process's NCL ID will be cleared.

See Also:

DEBUG START and DEBUG SET command descriptions.

DEBUG TRACE

Function: Displays the current nesting of procedures and subroutines for a process under the control of the debugger.

```
DEBUG     TRACE
          [ ID=nclid | PROCEDURE=procname ]
(0)
```

Use: All procedure and subroutine nesting levels are displayed for the identified NCL process.

Operands:

ID=*nclid*

The NCL process identified by the ID operand is listed. The process must be under the control of the debug session owned by the command issuer's environment.

PROCEDURE=*procname*

The NCL process whose base procedure name or current procedure name matches this operand is listed. There must be only one process currently under the control of the debug session that has the base or current procedure name specified. If more than one exists, use the ID operand. The process must be under the control of the debug session owned by the command issuer's environment.

Examples:

```
DEBUG TRACE ID=296
DEBUG TRACE PROCEDURE=MYPROC
```

Notes: If the ID and PROCEDURE operands are both omitted, then the last referenced NCL process is the target of the DEBUG TRACE command.

If the ID is not specified, the last referenced NCL ID is used as the default.

See Also: DEBUG SOURCE command description.

DEFALIAS

Function: Defines an alias entry for the Alias Name Translation Facility of NEWS.

```
DEFALIAS      NAME=alias
              NET=aliasnet
              RNAME=realname
              RNET=realnet
             [ RCDRM=cdrm ]
             [ TYPE={ LU | COS | MODE } ]
(4)
```

Use: This command is used to add an alias name definition to NEWS for use by the Alias Name Translation Facility.

Operands:

NAME=*alias*

Provides the required 1- to 8-character *alias* or *origin* name of the type specified by the TYPE operand. This name, together with the origin network name (specified with the NET operand), uniquely identifies the resource concerned. For TYPE=LU definitions this is an alias name for the real resource identified by the RNAME and RNET operands. For TYPE=COS or TYPE=MODE definitions, this is an actual name used in the origin network and the RNAME operand identifies an equivalent name to be used in the target network specified by the RNET operand.

If the last character specified in the name is a right angle bracket (>), (for example, NAME=TSO>), the name is assumed to be generic and trailing input characters will be preserved in the translation. This is explained in the examples below. In this case the RNAME operand must also specify a generic name.

NET=*aliasnet*

Provides the required 1- to 8-character network identifier, known to VTAM, in which the *alias* or *origin* resource name is known. This name can be generically defined by entering an asterisk (*) as the last character. In this case the *alias* name is assumed to be defined in all networks which match the generic *aliasnet* name supplied. (For example, if NET=NTWK* is specified, then the alias name is defined for all networks that have a network identifier beginning with the characters NTWK).

RNAME=*realname*

Provides the required 1- to 8-character name of the type specified by the TYPE operand and defines the *real* or *target* name of the resource as it is known in the target network (specified by the RNET operand). For an LU type name this must be the real name of the resource and the RNET operand is its owning network. For TYPE=COS or TYPE=MODE definitions this target name identifies the name of a TYPE=COS or TYPE=MODE definition in the target network which is equivalent to the one specified in the origin network (defined by the NAME and NET operands).

If the NAME operand specifies a generic name, then the RNAME operand must also specify a generic name; otherwise neither can.

RNET=*realnet*

Provides the required 1- to 8-character network identifier, known to VTAM, in which the target resource name identified by the RNAME operand can be used. For TYPE=LU definitions this identifies the network in which the real resource exists. This *realnet* name entered must be the full (non-generic) network name.

RCDRM=*cdrm*

Can specify the 1- to 8-character name of the owning CDRM for TYPE=LU definitions only. If the *cdrm* name is specified then this must be the LU's owning CDRM or session establishment will fail.

TYPE=LU

Indicates that the *alias* name being defined is an LU name. This is the default.

TYPE=COS

Indicates that the *alias* name being defined is a COS name (VTAM Class-of-Service name).

TYPE=MODE

Indicates that the *alias* name being defined is a LOGMODE name.

Examples:

Example 1:

```
DEFALIAS NAME=IMSA NET=NETWORKB RNAME=IMS RNET=NETWORKA
```

defines an alias name IMSA as an LU (by default) in NETWORKB. It actually exists in NETWORKA and its real name is IMS.

Example 2:

```
DEFALIAS TYPE=MODE NAME=D4C32782 NET=NETA RNAME=RSNAMOD2 RNET=NA007
```

defines a logmode name D4C32782 which is used in network NETA. In network NA007 an *equivalent* logmode name of RSNAMOD2 is to be used.

Example 3:

```
DEFALIAS NAME=ATSO> NET=* RNAME=TSO> RNET=NETWORKA
```

defines alias LU names beginning with ATSO. Such names are generically defined to exist in all networks, and will be translated to real names which actually exist in NETWORKA.

If the Alias Translation Facility receives an alias input name from VTAM (for any origin network) of the form ATSO*xxxx* (where *xxxx* is any character string), it will translate it to a real name in network NETWORKA of TSO*xxxx*. For example, the name ATSO0023 would be translated to TSO0023.

Any translation required from real to alias names will also preserve trailing input characters. For example, if real name TSO003 from network NETWORKA required translation to its alias name in network NETC the result would be ATSO003.

Notes:

Take care, when using generic name definitions that a translated name (for either alias to real, or real to alias translations) cannot exceed 8 characters in length, otherwise truncation occurs.

An exact name takes precedence over a generic name and a longer generic name takes precedence over a shorter one when the Alias Name Translation Facility searches for a match. Consider, for example, the situation where the names IMSA, IMSA>, and IMS> have been defined. An input name of IMSA, although a generic match with all three definitions, is translated using the first one. Similarly, an input name of the form IMSA*xxxx* would be translated using the second definition, while any other input of the form IMS*xxxxx* would use the third definition shown.

The alias name and network name combination must be unique. An attempt to add a second definition for the same type of name specifying an existing alias name and network name will fail. Although it is possible to define more than one alias name and network name to have the same real name and network name translation, this should be avoided if possible as the reverse translation is then not unique. The reverse translation of real name to alias would proceed using the first definition added specifying the real name and network pair. If a second definition is added specifying an existing real name and network name a warning message is issued.

For a TYPE=COS definition, it is allowable to omit either (or both) *alias* and *realname*. It is valid to specify NAME or RNAME. This allows for the definition of a COS name which is all blanks.

See Also:

The REPALIAS, DELALIAS, SHOW DEFALIAS, and XLATE commands.

The *Unicenter NetMaster Network Management for SNA Implementation and Administration Guide*.

DEFCLASS

Function: To define Session, SAW, RTM, or Resource classes.

```
DEFCLASS     SESSION=sessclass
                    { [ RTMCLASS=rtmclass ] [ SAWCLASS=sawclass ] }
                    { [ COS=cosname ] [ ER=er ] [ PRI=name ]
                          [ SEC=name ] [ TP=tp ] [ VR=vr ] }
                    [ SOURCE={ ALL | LOCAL | REMOTE } ]
                    [ SSCP=sscp name ]
                    [ SCLASS={ SD | XD | XN } ]
                    [ STYPE={ LL | SL | SP | SS | MAI } ]

DEFCLASS     SAW=sawclass
                    [ ACCT={ YES | NO } ]
                    [ EVENT={ YES | NO | ERROR } ]
                    [ KEEP={ YES | NO | LOCAL } ]
                    [ LOG={ SUMMARY | DATA | ERROR | ALL | NO} ]
                    [ TRACE=(i,f) ]

DEFCLASS     RTM=rtmclass
                      BOUNDS=(value1,value2 ...  value4)
                      OBJPC=objpc
                      OBJTIME=objtime
                    [ RTMDEF={ FIRST | CDEB | KEYBD | LAST } ]

DEFCLASS     RESOURCE=rscclass
                    [ RTMCLASS=rtmclass ]
                    [ STATS=NO | STATS= YES [ LIMIT=nnn ] ]
                    { [ LINK=name ] [ PU=name ] [ LU=name ] }
(4)
```

Use: This command is used to define the attributes of the various categories or *classes* of session related information that NTS is to accumulate and process.

Operands: **SESSION=*sessclass***

Identifies this class as a session class and provides the unique *sesclass* name of up to 8 characters which identifies this class. Use this name in the REPCLASS and DELCLASS commands to replace or delete this session class respectively.

The DEFCLASS operands specific to the definition of a SESSION class are:

RTMCLASS=*rtmclass*

Specifies the *rtmclass* name which is to be used by sessions that are associated with this session class. If no RTM class is found for a session then no RTM data is collected for that session.

SAWCLASS=*sawclass*

Specifies the *sawclass* name from which sessions falling within the session class being defined are to take their SAW class attributes. If not specified then the system SAW class defaults will apply to sessions in this session class.

COS=*cosname*

Optionally specifies the *cosname* of sessions which can be considered as within this session class. Generic names are not supported.

ER=*er*

Optionally provides the Explicit Route number *er* (0-7) that sessions must have to be considered in this session class.

PRI=*name*

Optionally provides the full or partial name of the primary resources considered to be in this session class. A generic name can include an asterisk (*) as a wild character in any position in the name. This represents a single wild character only. The last character can be specified as a trailing arrow (>) which indicates all trailing character positions are wild. Alternatively the last character can be specified as a trailing hyphen (-) which has the same meaning as > but in addition only those characters in the name up to the - are used when logging the session to the database, the rest of the name considered padded with blanks. (This allows, for example, by specifying PRI=TSO- all sessions between a particular terminal and primary partners such as TSO0001, TSO0014, TSO0055 and so on, to be logged in the database as different session incidences of the same session name pairs).

SEC=*name*

Optionally provides the full or generic name of the secondary session partner for sessions within this session class. Generic names as described for primary names above are allowed.

TP=*tp*

Optionally provides the Transmission Priority number *tp* (0-2) that sessions must have to be considered in this session class.

VR=*vr*

Optionally provides the Virtual Route number *vr* (0-7) that sessions must have to be considered in this session class.

SOURCE={ <u>ALL</u> | LOCAL | REMOTE }

Selects sessions on the basis of their origin. If SOURCE=LOCAL is specified then only sessions whose data is derived from the local VTAM are selected. If SOURCE=REMOTE then only sessions whose data is derived from other NTS systems are selected (that is, data is collected from NTS systems that are not in session with the local VTAM). ALL is the default and selects both types of session.

SSCP=*sscp name*

Selects sessions derived from the domain defined by the SSCP named.

SCLASS={ SD | XD | XN }

SD specifies sessions in which both session partners are in the same domain, XD specifies cross domain sessions and XN specifies cross network sessions. A session must match this parameter to be associated with this session class. If this parameter is not specified, the class of session is not used to match the session class.

STYPE={ LL | SL | SP | SS | MAI }

LL specifies LU-LU sessions, SL specifies SSCP-LU sessions, SP specifies SSCP-PU sessions and SS specifies SSCP-SSCP sessions. STYPE=MAI selects MAI sessions. A session must match this parameter to be associated with this session class. If this parameter is not specified, the type of session is not used to match the session class.

SAW=*sawclass*

Identifies a SAW class and provides the unique *sawclass* name of up to 8 characters which identifies the class. Use this name in the REPCLASS and DELCLASS commands to replace or delete this SAW class respectively. Use this name also on the SAWCLASS operand of the DEFCLASS command when defining a *Session class* to provide the name of the SAW class applicable to the particular session class (see below).

The DEFCLASS operands specific to the definition of a SAW class are:

ACCT={ YES | NO }

Indicates whether or not accounting data is to be accumulated for sessions using this SAW class. The collection of accounting data firstly depends upon the SYSPARM NTSACCT operand value. If SYSPARM NTSACCT=NO is in effect then no accounting information is collected regardless of the ACCT operand value. If SYSPARM NTSACCT=ALL is operative then accounting data is collected for all sessions again regardless of this ACCT operand. However, if the SYSPARM NTSACCT=SELECTIVE (default) option is in effect then the ACCT operand of the SAW class definition determines whether or not accounting data is to be collected for those sessions using this SAW class.

EVENT ={ YES | NO | ERROR }

Indicates whether events are to be generated for sessions matching this class. The events that can be generated are as follows:

- Session start ($$NTS.SESSION.START)
- Session end ($$NTS.SESSION.END)
- Session failures ($$NTS.SESSION.FAIL)
- RTM objective not met ($$NTS.RTM.OBJ.EXCEEDED)

When EVENT=YES is specified, all these events are generated and sent to profiled users. When EVENT=NO is specified or defaulted, none of these events are generated. When EVENT=ERROR is specified, only the session failure event is generated.

Note

The NTS SYSPARMS NTSEVENT operand must be set to YES to cause NTS events to be generated.

KEEP={ YES | NO | LOCAL }

Specifies whether or not session (or SAW) data in this class is to be retained by NTS. If KEEP=NO is specified then NTS discards session awareness data for any sessions mapped into this SAW class. No information on these sessions is subsequently available to the user, and no other NTS information can be collected for such sessions. Hence when KEEP=NO is specified all other SAW class operands are meaningless. If KEEP=LOCAL is specified then data relating to any session in this class will not be forwarded (via an ISR link) to any other NTS.

LOG={ SUMMARY | DATA | ERROR | ALL | NO }

Determines what logging options are to be applied to data in this SAW class. The default LOG=SUMMARY option means all data except trace is logged at session end if the session ends normally, however all data including trace data is logged if the session ends in error. The LOG=DATA option means that if there is any session data (such as trace data, accounting data and so on) available for the session then all data is logged otherwise no logging takes place. The LOG=ERROR option means that all data will be logged but only if the session ends due to an error, otherwise no logging takes place. The LOG=ALL option indicates that all data is to be unconditionally logged, while LOG=NO indicates no data should be logged at all.

TRACE=*(i,f)*

Provides the depth *i* of the initial trace queue, and the depth *f* of the final trace queue for this SAW class. Both *i* and *f* can be in the range 0 to 999, the default values being 4 and 20 respectively. When session establishment trace data arrives for a session it will be placed on the initial queue until the queue depth is reached. Other trace data is placed on the final queue and when the queue limit is reached the oldest data is overwritten with the most recent. Trace data continues to wrap on the final queue until the session ends or tracing is terminated.

RTM=*rtmclass*

Identifies an RTM class and provides the unique *rtmclass* name of up to 8 characters which identifies this class. Use this name on the REPCLASS and DELCLASS commands to replace or delete this RTM class respectively. Use this name also on the RTMCLASS operand of the DEFCLASS command when defining a session class (see below) to associate sessions in the session class with this RTM class.

The DEFCLASS operands specific to the definition of an RTM class are:

BOUNDS=*(value1,value2...value4)*

Specifies up to 4 boundary values which are to be set in the control unit and used to accumulate RTM data for this session. Each of these values must be a time specified in the same manner as for *objtime* above. Each subsequent value must be greater than the previous (that is, *value1* < *value2* < ... *value4*). One of the values specified must be the same as the *objtime* specified above.

OBJPC=*objpc*

Defines the objective percentage for this RTM class as a value in the range 1 to 100. If the percentage of total transactions which complete within the *objtime* specified equals or exceeds the *objpc* value specified then the performance objectives for this RTM class are considered to be met.

OBJTIME=*objtime*

Defines the objective response time for the session. This is a time in the range 0.1 seconds to 30 minutes and can be specified as *mm:ss* (minutes and seconds), *mm:ss.t* (minutes, seconds and tenths of a second), *ss* (seconds), or *ss.t* (seconds and tenths of a second). This value must correspond to one of the boundary values (see BOUNDS operand below) or the DEFCLASS command will be rejected.

RTMDEF={ <u>FIRST</u> | KEYBD | CDEB | LAST }

Defines the RTM definition criteria used. If FIRST is specified (or defaulted) the response time is measured until the first character of the host datastream is received. For KEYBD it is measured until the keyboard is unlocked, for CDEB it is measured until an SNA Change Direction or End Bracket order is received, while for LAST it is measured until the last character of the host datastream is received. See the relevant component description manual of devices which support RTM (such as 3274 and 3174) for complete details on the operation and collection of RTM data in the control unit.

RESOURCE=*rscclass*

Identifies this class as a resource class and provides the unique *rscclass* name of up to 8 characters which identifies this class. Use this name in the REPCLASS and DELCLASS commands to replace or delete this resource class respectively.

The DEFCLASS operands specific to the definition of a Resource class are:

RTMCLASS=*rtmclass*

Optionally specifies the *rtmclass* name which is to be used for resources that are associated with this resource class. The format of RTM responses received will be compared with the format defined in the RTM class definition. Statistics will be kept for matching responses. This parameter is only relevant if statistics are being collected for resources associated with this class. If this parameter is not specified, no RTM statistics will be collected for resources associated with this resource class.

STATS={ YES | NO }

Indicates whether or not statistics are to be collected for resources associated with this class. The collection of resource statistics firstly depends on the SYSPARMS NTSRSTAT operand value. If SYSPARMS NTSRSTAT=NO is in effect, no resource statistics are collected, irrespective of the value of this operand. Resource accounting statistics are derived from session accounting data. If no session accounting information is being collected for sessions in which a resource participates, no statistics will be available for the resource. It is strongly recommended that SYSPARM NTSACCT=ALL be in effect if resource statistics collection is required.

LIMIT=*nnn*

nnn is minutes, ranging from 0 to 255. This parameter specifies the number of intervals to occur before the statistics for the oldest interval are overwritten. This parameter is only relevant if STATS=YES is specified. The operand for this parameter will override any value specified for the NTSRLIM operand of the SYSPARMS command.

Note

High settings of this operand can consume large amounts of storage.

LINK=*name*

Optionally provides the full or partial name that must be used by links to be considered associated with this class. See the *Generic Name* description below for valid formats.

PU=*name*

Optionally provides the full or partial name that must be used by PUs to be considered associated with this class. See the *Generic Names* description below for valid formats.

LU=*name*

Optionally provides the full or partial name that must be used by LUs to be considered associated with this class. See the *Generic Names* description below for valid formats.

Note

Generic Names

The generic format uses an asterisk (*) as a single wild character and a trailing arrow (>) as an indicator that all trailing characters are wild. For LU operand values a trailing hyphen (-) can be used. This is treated in the same way as the arrow with the additional function of collecting all the statistics under the single name. This name is made up of all the characters of the LU operand value excluding the hyphen.

DEFCLASS

Examples:

```
DEFCLASS SAW=KEEPACCT ACCT=YES LOG=ALL
DEFCLASS SAW=NOKEEP KEEP=NO
DEFCLASS RTM=CICS BOUNDS=(0.5,1,3,10) OBJTIME=1 OBJPC=90
DEFCLASS SESSION=CICS PRI=CICS>> SAWCLASS=KEEPACCT
             RTMCLASS=CICS
DEFCLASS SESSION=TSO PRI=TSO- SAWCLASS=KEEPACCT
DEFCLASS SESSION=VTAM COSNAME=ISTVTCOS SAWCLASS=NOKEEP
DEFCLASS RESOURCE=ALU LINK=L080BF PU=PUA> LU=LUA>
             RTMCLASS=TSORTM2 STATS=YES LIMIT=30
DEFCLASS SESSION=PUS STYPE=SSPU SAWCLASS=SAWEV<P255D>
```

Notes:

Sessions not mapped by a Session class receive the system SAW class default values (see the underscored SAW class operand values above).

If any session class selection operands are omitted all values of the type omitted are considered to be valid for the class. For example if no PRI= operand is supplied all primary names are considered valid (that is, omitting say the PRI operand is equivalent to specifying PRI=>, omitting the ER operand is equivalent to specifying ER=* and so on). Where selection options are provided sessions must match all such options to be considered in the session class.

The order of the operands as shown above for a session class definition is the order of precedence for class selection where a session could otherwise exist in more than one class. Further, a more specific name will be matched before a less specific one (for example, SEC=LU25 will be found before SEC=LU**).

If a session falls within a particular Session class and either the SAWCLASS or RTMCLASS is not defined for that Session class, the search will continue for a session class matching the session attributes to determine the missing SAW or RTM class. When both the SAWCLASS and RTMCLASS are known, or all session classes exhausted, the search ends and default SAW attributes assigned if necessary.

At least one of the LINK, PU or LU parameters must be specified in a Resource class definition. If STATS=YES is specified and no LIMIT parameter is present, the number of intervals is taken from the NTSRSINT operand of the SYSPARMS command. If no RTM Class is specified, no RTM statistics will be collected for any resource associated with the resource class.

See Also:

The REPCLASS, DELCLASS, SAWARE, NTSMOD, and STRACE command descriptions.

DEFLINK (INMC)

Function: The DEFLINK command, and the related REPLINK and DELLINK commands, are used to maintain the Dynamic Link Table, or DLT, used by INMC services.

```
DEFLINK        TYPE=INMC
               { LUNAME=luname LINK=link |
                    MASK=mask PREFIX=prefix |
                    LUNAME=luname PREFIX=prefix }
               [ AM=amname ]
               [ { COLOR | COLOUR }=color ]
               [ { HLIGHT | HLITE }=highlight ]
               [ RT=nnnnn ]
               [ MSGID=id ]
               [ DEFNAME=inmcefname ]
(3)
```

Use: INMC Dynamic links are created on request for a remote system provided the request matches the criteria set up in a dynamic link definition on this system. Use the DEFLINK command to build a dynamic link definition and to define the criteria for accepting dynamic links requested by other systems. See the chapters on INMC in your *Management Services Administrator Guide* and *Network Control Language User's Guide* for further details.

Operands:

TYPE=INMC

Indicates that this DEFLINK command defines an INMC Dynamic Link Table entry. TYPE=INMC is the default, so this operand is optional when defining INMC links.

LUNAME=*luname*

Provides the LU name that corresponds to the link name described below.

LINK=*link*

Provides a specific LINK name, and requires that the LU name parameter is used to provide the corresponding LU name of the remote system. When the LINK and LUNAME operands are used, the MASK, LUMASK, and PREFIX operands are not allowed. If a LINK START command is issued for this link name and type, then the DEFLINK definition can be used to complete the link details.

MASK=*mask*

This operand is mandatory. It defines a mask against which any user data on an incoming BIND (from another MS region) or logon request (from an NMPC system) is to be tested. The mask value can be a generic string with asterisks as wild characters. If the user data matches the mask, then a dynamic link is built. The link name assigned is the actual LU name, prefixed by the 4-character PREFIX value. The use of MASK and PREFIX is mutually exclusive with the use of LINK and LUNAME.

PREFIX=*prefix*

This operand is required when MASK is used. When a dynamic link is built as a result of this definition, its linkname will consist of this prefix concatenated in front of the VTAM nodename for the remote system. The prefix must be four (4) characters long.

DEFLINK (INMC)

AM=*amname*

Specifies the access method to be used for the link. Can be VTAM, TCPIP, or EPS (End Point Services). If this operand is omitted, then the primary access method is used.

{ COLOR | COLOUR }=*color*
{ HLIGHT | HLITE }=*highlight*

Provides a means of identifying messages originating from this set of dynamic links through extended highlighting or color displays. Color can be:

```
RED, YELLOW, WHITE, BLUE, GREEN, TURQUOISE, or PINK
```

Highlighting can be:

```
REVERSE, BLINK, USCORE
```

Both FTS and ROF messages received from these links are highlighted as requested for terminals supporting that type of data stream.

RT=*nnnnn*

Retry interval. The default is 60 seconds.

MSGID=*id*

A 1 to 4 character string that is to be used as a prefix to any message received by a ROF user across any of these dynamic links. The default MSGID is the prefix value. Individual OCS users that use ROF to receive messages from remote systems can choose their own remote system prefix via the SIGNON command, overriding the default value set on the DEFLINK command. DEFLINK commands will usually be included in the system initialization procedures.

DEFNAME=*inmcefname*

Specifies the name of an INMC multipath link definition. This is the only way that a multipath link can be established when using a DEFLINK.

If specified, when an INMC link is dynamically started on this system as a result of matching this DEFLINK, then the named INMC multipath link definition record is used as a model for the session definitions. The domain name in the definition is ignored.

A LINK START command that matches a DEFLINK definition also uses the DEFNAME, unless overridden on the LINK START.

Notes: Regardless of the DEFLINK commands in use, a remote link request does not result in a dynamic link being established if the SYSPARMS DYNLMAX limit has been reached. The default for SYSPARMS DYNLMAX is 10.

Examples:

```
DEFLINK MASK=IMPC PREFIX=NMPC
DEFLINK MASK=N*P* PREFIX=DYNL COLOR=RED HLITE=REVERSE
DEFLINK MASK=* PREFIX=LINK MSG=INMC
```

See Also: The LINK (INMC), REPLINK, SHOW DEFLINK, and SHOW LINK commands.

DEFLINK (APPC)

Function: The DEFLINK command, and the related REPLINK and DELLINK commands, are used to maintain the Dynamic Link Table, or DLT, used by LU6.2 services.

```
DEFLINK        TYPE=APPC
               { LUNAME=luname LINK=link |
                    LUMASK=lumask PREFIX=pref |
                    LUNAME=luname PREFIX=pref }
               [ AM=amname ]
               [ QUAL=qualifier ]
               [ PASSWORD={ EXIT | password } ]
               [ OPSET=opset ]
               [ { COLOR | COLOUR }=color ]
               [ { HLIGHT | HLITE }=hlite ]
               [ MON={ YES | NO } ]
               [ USERENV={ GLOBAL | LINK } ]
               [ TRACE={ YES | NO } ]
(3)
```

Use: APPC Dynamic Link Table, or DLT, entries are used to start APPC links in response to an &APPC ALLOCATE verb or a LINK START command. Use the DEFLINK command and the related REPLINK and DELLINK commands to maintain the DLT.

See the chapters on APPC in your *Management Services Administrator Guide* and *Network Control Language User's Guide* for further details.

Operands:

TYPE=APPC

Indicates that this DEFLINK command defines an APPC Dynamic Link Table entry, and pertains to LU6.2 session connections only. The default is TYPE=INMC so this operand is required for APPC DLT definitions.

LUNAME=*luname*

Provides a specific network LU name (of up to 8 characters) that, when connected to this system by an LU6.2 session, will be known by the *link* name as described below. By using the LINK and LUNAME parameters a fixed link name can be associated with a target LU, such as another LU6.2 capable application system. When an LU6.2 session starts between this system and the target *luname* this DEFLINK entry is used to complete the link details.

LINK=*link*

Provides a specific *link* name (of up to 12 characters), and can only be used when the LUNAME parameter is used to supply a specific *luname*. Both the *link* and *luname* supplied must be unique amongst all APPC DLT entries. The use of the LINK parameter is mutually exclusive with the use of the PREFIX parameter.

LUMASK=*lumask*

Provides an LU name mask that must be unique amongst APPC DLT definitions. The *lumask* can use an asterisk (*) to denote a wild character in any position, or a trailing * to denote the remainder of the name as wild. When matching an LU name against the *lumask* a more specific mask is selected over a more generic one, with leftmost characters having precedence over those occurring to the right (for example, the name N12P3L15 matches the masks, N*, N****L, and ***P3L15, but selects the second as its leftmost character takes precedence over the third mask, and it is a more specific match than the first mask). This parameter is mutually exclusive with the LUNAME parameter.

PREFIX=*pref*

Provides a 4-character string that, when prefixed to the actual LU name, creates a link name, and is used when a DEFLINK entry is matched using the *lumask* described above. This parameter is required when LUMASK is used, but is mutually exclusive with the LINK parameter.

AM=*amname*

Specifies the access method to be used for the link. Can be VTAM, PCPIP, EPS (End Point Services), or INMC. If omitted, the primary access method is used.

QUAL=*qualifier*

Provides the name of qualifier with up to 4 characters to be used for this destination to qualify transaction identifiers (see DEFTRANS). This operand is optional, and if omitted then transactions to this remote LU are unqualified.

PASSWORD={ EXIT | *password* }

Provides a 16-character hexadecimal string, representing 8 bytes of data, to be used as a password for the link. A password is required only if LU-LU session verification is required, and can be supplied in the DEFLINK definition, or on the LINK START command. DEFLINK (APPC).

The EXIT keyword is only valid if the system is running an installation-supplied security exit. If specified, the exit is invoked and will return the password.

OPSET=*opset*

Nominates the name of the OSCT entry to be used to determine the communication options supported by this remote LU.

{ COLOR | COLOUR }=*color*
{ HLIGHT | HLITE }=*highlight*

Provides a means of identifying messages originating from this set of dynamic links through extended highlighting or color displays. Color can be:

```
RED, YELLOW, WHITE, BLUE, GREEN, TURQUOISE, or PINK
```

Highlighting can be:

`REVERSE, BLINK, USCORE`

Both FTS and ROF messages received from these links are highlighted as requested for terminals supporting that type of data stream.

MON={ YES | NO }

Specifies whether or not link messages for those connected systems using this definition are to be displayed as monitor messages. If MON=NO is specified, or defaulted, then link messages are directed to the activity log only. If MON=YES is specified then link messages are sent to all OCS monitor users.

USERENV={ GLOBAL | LINK }

The USERENV operand is used to indicate whether or not an APPC user region created to service an attach request for a given user ID and received from the remote network node defined by the DEFLINK definition, will also service subsequent requests (for the same user ID) received from different remote network nodes.

GLOBAL indicates that the APPC region built will service attach requests from different remote nodes. LINK indicates that the APPC region will only service attach requests received from the remote node defined by this DEFLINK definition.

TRACE={ YES | NO }

Specifies whether or not LU6.2 session level tracing will be performed for all sessions in this link.

Examples:

```
DEFLINK TYPE=APPC LU=NMA LINK=NMPROD OPSET=NM MON=YES
DEFLINK TYPE=APPC LU=CICS1 LINK=TESTCICS
DEFLINK TYPE=APPC LUMASK=OS2* PREFIX=WSTN OPSET=WSOPSET
DEFLINK TYPE=APPC LU=NM1 PREFIX=NETM OPSET=NM
```

Notes:

DEFLINK commands will usually be included in the system initialization procedures.

Regardless of the DEFLINK commands in use, a remote link request does not result in a dynamic link being established if the SYSPARMS DYNLMAX limit has been reached. The default for DYNLMAX is zero.

When an LU6.2 session setup request for a remote LU is first processed the DLT is searched to locate an entry that will be used to initialize the APPC link. Those entries defined with specific LINK and LUNAME parameters are examined first. If a DLT entry with a matching LU name is found the APPC link is initialized using the corresponding LINK name. If no specific entry is found, the remaining entries in the DLT are examined to locate the most appropriate LUMASK match, as described for the LUMASK parameter above. If found APPC link initialization proceeds using a link name built as described for the PREFIX parameter above. Otherwise the session setup request is denied with a ‘permission rejected’ sense code.

See Also:

The LINK (INMC), SHOW DEFLINK, and SHOW LINK commands.

DEFLINK (NVC)

Function: The DEFLINK command, and the related REPLINK and DELLINK commands, are used to maintain the Dynamic Link Table, or DLT, used by LU6.2 services.

```
DEFLINK        TYPE=NVC
                LUNAME=luname LINK=link
              [ SSID=NetView-ssid ]
              [ { COLOR | COLOUR }=color ]
              [ { HLIGHT | HLITE }=hlite ]
              [ MSGID=id ]
(3)
```

Use: Dynamic links are created on request for a remote system provided the request matches the criteria set up in a dynamic link definition on this system. Use the DEFLINK command to build a dynamic link definition and to define the criteria for accepting dynamic links requested by other systems.

NVC Dynamic Link Table, or DLT, entries are used to start SOLVE:Attach for NetView links in response to NetView link definitions. These are defined in the NetView initialization member and activated when the NVCV task is started. Use the DEFLINK command and the related REPLINK and DELLINK commands to maintain the DLT.

See the chapter *Customizing and Communicating With NetView* in the *SOLVE: Attach for NetView Customization Guide* for further details.

Operands:

TYPE=NVC

Indicates that this DEFLINK command defines an NVC Dynamic Link Table entry. The default is TYPE=INMC so this operand is mandatory when defining NVC links.

LUNAME=*luname*

Provides the LU name that corresponds to the link name described below.

LINK=*link*

Provides a specific LINK name, and requires that the LU name parameter is used to provide the corresponding LU name of the remote system. If a LINK START command is issued for this link name and type, then the DEFLINK definition can be used to complete the link details.

SSID=*NetView-ssid*

A 4-character string that specifies the first 4 characters of the NetView job name. If this operand is omitted, it must be specified on the LINK START command.

{ COLOR | COLOUR }=*color*
{ HLIGHT | HLITE }=*highlight*

Provides a means of identifying messages originating from this set of dynamic links through extended highlighting or color displays. Color can be:

```
RED, YELLOW, WHITE, BLUE, GREEN, TURQUOISE, or PINK
```

Highlighting can be:

```
REVERSE, BLINK, USCORE
```

Both FTS and ROF messages received from these links are highlighted as requested for terminals supporting that type of data stream.

MSGID=*id*

A 1- to 4-character string that is to be used as a prefix to any message received by a ROF user across any of these dynamic links. The default MSGID is the prefix value. Individual OCS users that use ROF to receive messages from remote systems can choose their own remote system prefix via the SIGNON command, overriding the default value set on the DEFLINK command. DEFLINK commands will usually be included in the system initialization procedures.

Regardless of the DEFLINK commands in use, a remote link request does not result in a dynamic link being established if the SYSPARMS DYNLMAX limit has been reached. The default for DYNLMAX is zero.

See Also: The LINK (INMC), SHOW DEFLINK, and SHOW LINK commands.

DEFLOGON

Function: Defines the logon information required to support a target application for the EASINET and MAI facilities of SOLVE:Access.

```
DEFLOGON    { INPUT=text | IN=text }
            [ IGE | IEQ  ]
            [ APPL={ vtamappl | * } ]
            [ DATA={userdata | INPUT | INPUT1 | NOEASINET } ]
            [ LOGMODE=table entry ]
            [ MID=MAI-id ]
            [ MSHR={YES | NO } ]
            [ APPSTAT={ YES | NO } ]
            [ MRESP={ AFTER | BEFORE } ]
            [ MPREF=prefix ]
            [ MTLIMIT=minutes ]
            [ MTACTION={ CANCEL | CC | SCRIPT | EVENT } ]
            [ NOTIFY={ YES | NO } ]
            [ REFRESH=key ]
            [ DESC=session description ]
(4)
```

Use: This command is normally reserved for Systems Programmer use. Its use is described in the *Management Services Administrator Guide*.

Operands:

INPUT=*text*

The input string entered by the operator, or the leading characters of the string to be entered.

IGE | IEQ

Identifies the type of match to be made against the input string. IGE (the default) specifies that a logon string will match on leading characters of the logon path defined using the INPUT operand (see above). Alternatively, IEQ can be specified which indicates that an exact match must be made.

APPL={ *vtamappl* | * }

VTAM APPL name of target application or * for Management Services' own APPL. You can also use the primary VTAM ACB APPL name instead of *, that is, the same name used for the PRI= parameter on the system initialization parameters.

If this operand is omitted, then an application name matching the INPUT text is assumed.

DATA={ *userdata* | INPUT | INPUT1 | NOEASINET }

The user data to be passed to the target application. Specify INPUT if text as entered by the user is to be passed, or INPUT1 if text as entered minus the first word is to be passed. Specify NOEASINET if the target application is another MS region and its EASINET facility is to be bypassed. If the *userdata* format is used and the data contains blanks, the string must be enclosed in quotes (”).

LOGMODE=*table entry*

VTAM log mode table entry to be used when the session is established.

> **Note**
> This operand is only used by EASINET in connection with the &LOGON verb. It is not used by MAI.

MID=*MAI-id*

Default MAI session ID if one is not supplied during creation of an MAI-FS session. If not supplied the APPL name is used as the default.

MSHR={ YES | NO }

Determines whether MAI sessions created using this entry will use ACB sharing.

APPSTAT={ YES | NO }

Determines whether the application status should be queried before establishing a session.

MRESP={ AFTER | BEFORE }

Specifies whether MAI-FS sessions using this definition are to respond to the application before or after the data has been sent to the terminal. The AFTER setting (default) provides synchronous session control, with the terminal's responses being echoed to the application. The SYSPARMS MAIRESP command controls the default setting. For example SYSPARMS MAIRESP=BEFORE setting would change the default for DEFLOGON MRESP to BEFORE

The BEFORE setting indicates that MAI-FS is to always respond positively upon receipt of data. It also provides an asychronous mode of operation.

MPREF=*prefix*

Specifies an ACB prefix for MAI to use as a logical terminal pool. If this operand is omitted, it will default to the SYSPARMS MAIPREF setting.

MTLIMIT=*minutes*

Specifies a time-out limit that applies to MAI sessions established using this DEFLOGON entry.

MTACTION={ CANCEL | CC | SCRIPT | EVENT }

Specifies the action to be taken when an MAI session is inactive for the time specified by MTLIMIT. The default is EVENT.

CANCEL cancels a session, CC causes a conditional cancel, SCRIPT causes a session script to be initiated, and EVENT causes an event to be issued.

NOTIFY={ YES | NO }

Specifies whether or not EASINET should use VTAM's Third Party Notify capability when establishing a session. The default is set by the VTAMNTFY system parameter.

REFRESH=*key*

Supplies the name of the application's refresh key. MAI-FS will include this key as part of the *lost screen image* message (N54201).

DEFLOGON

DESC=*session description*

Enter a text description of the application identified by this DEFLOGON entry. If specified, this must be the last operand. It can consist of any characters (including blanks) without the need for enclosing the data in quotes. If your system is configured with the MAI feature, this description will be displayed beside the MAI menu entry for this application.

Examples:

```
DEFLOGON INPUT=IMS IEQ APPL=IMS LOGMODE=HIPRI
DEFLOGON INPUT=NM IEQ APPL=* DESC=Production SOLVE
DEFLOGON INPUT=NM2 DATA=NOEASINET
DEFLOGON INPUT=OP APPL=TSO DATA='OPTSO PR(PR1) SIZE(2000)'
DEFLOGON INPUT=TSO DATA=INPUT1
DEFLOGON INPUT=TSO APPL=TSOT MSHR=YES MID=TSO
DEFLOGON INPUT=CICSP MSHR=YES DESC=Production CICS
```

Notes:

DEFLOGON commands are normally executed from the NMINIT procedure that is automatically executed during system initialization. This ensures that the definitions are immediately available when the system becomes active.

The logon path description will be displayed against any appropriate entries on an MAI menu. The descriptions can be displayed using the SHOW DEFLOGON command, so that the descriptions can be displayed on an EASINET panel.

The SHOW SESS command can be used to display those terminals that are attached to Management Services under the control of the EASINET facility.

See Also:

The DELLOGON, REPLOGON, and SHOW DEFLOGON commands.

DEFMODE

Function: Defines an entry in the Mode Control Table, or MCT, used by LU6.2 services.

```
DEFMODE      MODE=mode
             [ MODENAME=modename ]
             [ LOGMODE=logmode ]
             [ SESSLIM=nn ]
             [ MINWIN=nn ]
             [ MINLOSE=nn
(4)
```

Use: This command and the related REPMODE and DELMODE commands are used to maintain the MCT. It is normally reserved for Systems Programmer use and is further documented in the *Management Services Administrator Guide*.

Operands:

MODE=*mode*

Identifies the unique name (of up to 8 characters) for this mode control table entry. This name can be used to relate an OSCT definition to this MCT definition by including the *mode* name in the list of the parameters following the MODE operand for the DEFOPSET command. This operand is required.

MODENAME=*modename*

Provides the actual LU6.2 *modename* for this mode definition. This *modename* can be used to relate a TCT definition to this MCT definition by specifying the *modename* on the MODENAME operand for the DEFTRANS command. This *modename* can also be specified on the MODENAME parameter of an &APPC ALLOCATE verb if necessary to override any system defaults. If this operand is omitted, then the *modenam*e defaults to the *mode* name specified on the MODE parameter.

LOGMODE=*logmode*

Provides the actual logmode name to be used on session initiation requests for this mode set definition. This can be used to assign certain network parameters such as RU sizes, and Class of Service (COS) values. If this operand is omitted, then no VTAM *logmode* is used for the session initiation request.

SESSLIM=*nn*

Specifies the session limit (up to 99) for this mode name definition (corresponding to the LU6.2 architected LU_MODE_SESSION_LIMIT parameter of the CNOS verbs) and is applicable where parallel sessions are supported. If this operand is omitted, a limit of 1 is assumed.

MINWIN=*nn*

Specifies the minimum number of contention winner sessions for this mode name definition (corresponding to the LU6.2 architected MIN_CONWINNERS_SOURCE parameter of the CNOS verbs). If this operand is omitted, a minimum of 1 is assumed.

MINLOSE=*nn*

Specifies the minimum number of contention loser sessions for this mode name definition (corresponding to the LU6.2 architected MIN_CONWINNERS_TARGET parameter of the CNOS verbs). If this operand is omitted, a minimum of 0 is assumed.

Examples:

```
DEFMODE MODE=NMMODES MODENAME=LU62SNGL SESSL=1 MINW=1 MINL=0
DEFMODE MODE=NMMODEP MODENAME=LU62PARL SESSL=20 MINW=5 MINL=5
DEFMODE MODE=CICSMODE LOGMODE=LU62PARL SESSL=10 MINW=2 MINL=2
```

Notes:

APPC conversations request the use of LU6.2 sessions by specifying a destination, via a LINK name or LU name, and a mode name. The mode name is used to identify a certain class of session required by the conversation.

See Also:

The REPMODE, SHOW DEFMODE, and the DEFOPSET commands.

DEFMSG

Function: Defines messages for delivery to the PPOPROC system level NCL procedure.

```
DEFMSG      [ SET | RESET ]
            [ DELIVER=destination ]
            [ EVENTNAME=ppo-event-suffix ]
            { msgclass | MSGID=(nnnn,...,nnnn) | nnnn ... nnnn }
(4)
```

Use: To specify delivery options for VTAM messages received from PPO and SPO, or from other systems, across ISR links.

Operands:

SET | RESET (default SET)

SET sets delivery destinations for specified messages and also sets the ACTIONED indicator. RESET resets delivery options and ACTIONED indicator.

DELIVER=*destination*

(Default is ALL). Specifies destinations to which messages are to be delivered and can only be specified with the SET option. Valid destinations are:

PPO

Pass message to PPOPROC.

LOCAL

Pass to eligible local OCS receivers.

REMOTE

Pass to remote systems which are configured to receive PPO messages across ISR.

ALL

Deliver to PPO, LOCAL, and REMOTE destinations.

NONE

Delete these messages.

EVENTNAME=*ppo-event-suffix*

This operand associates an event name with the arrival of a PPO message. If an event name is associated with a PPO message, then an EDS event with the name form $SNA.*ppo-event-suffix* is generated whenever the message is received.

DEFMSG

{ *msgclass* | MSGID=(*nnnn,...,nnnn*) | *nnnn ... nnnn* }

The *msgclass* operand specifies a class of PPO messages to be actioned. It can be one of the following values:

DEFAULT
Distributed default monitor messages.

UNSOLICITED
Unsolicited VTAM messages.

NON-VTAM
Non-VTAM messages which come across PPO ISR links with non-IBM platforms, or non-VTAM messages generated by &PPOALERTs.

ALL
All messages.

Specify a format of the MSGID operand and indicate specific message numbers which are to be actioned.

Examples:

```
DEFMSG SET DELIVER=(PPO,REMOTE) 123 245 1011
DEFMSG DEFAULT
DEFMSG RESET ALL
```

Notes:

Delivery options for VTAM messages are stored by Management Services in the DEFMSG table. The DEFMSG table acts as a high level filter for VTAM messages, to minimize the volume of messages that are passed to OCS windows or processed by PPOPROC. Management Services is distributed with a default table which defines certain known VTAM status messages. These are known to Management Services as the DEFAULT class of messages.

The DEFMSG command updates the default table values. It can also be used to SET the delivery options to the same values that were already set in the default table.

The difference is that Management Services remembers that these messages have been actioned. This is significant for UNSOLICITED messages, which can be processed as a class. However, if a particular message has been ACTIONED, then its attributes override the CLASS attributes.

DEFMSG SET DELIVER=NONE is different from DEFMSG RESET in that it sets the ACTIONED indicator.

See Also:

The SHOW DEFMSG and SHOW PPOSTAT commands.

For more information on the DEFMSG command, see the chapter titled *System Level Procedures* in the *Network Control Language User's Guide*.

DEFOPSET

Function: Defines an entry in the Option Set Control Table, or OSCT, used by LU6.2 services.

```
DEFOPSET      OPSET=opset
              [ MODE=(mode,...,mode) ]
              [ MAP={ NO | YES } ]
              [ LOG={ NO | YES } ]
              [ SECURITY={ NONE | USER | USERPSWD } ]
              [ PARSESS={ NO | YES } ]
              [ PV={ YES | NO } ]
(4)
```

Use: This command and the related REPOPSET and DELOPSET commands are used to maintain the OSCT. It is normally reserved for Systems Programmer use and is further documented in the *Management Services Administrator Guide*.

Operands:

OPSET=*opset*

Identifies the unique name (of up to 8 characters) for this option set control table entry. This *opset* name is referenced by the OPSET parameter on the DEFLINK command, and is used to relate a destination application with those service options supported by the application. This operand is required.

MODE=(*mode,...,mode*)

Provides a list of *mode* names as defined in the MCT that are available through this option set definition. Conversations performing allocations through this OSCT entry must request a *modename* included in one of the listed mode definitions or the allocation will fail. The first name in the list is used to locate the default mode entry for destinations using this OSCT entry. This operand is required.

MAP={ NO | YES }

Specifies whether or not this option set includes data mapping support. MS regions support data mapping, but most other applications do not. The default is NO.

LOG={ NO | YES }

Specifies whether or not this option set includes GDS log variable support. Management Services and many other applications support log data. The default is NO.

SECURITY={ NONE | USERPSWD | USER }

Specifies the level of security requested for this option set, and determines those security options accepted from the remote LU.

If NONE is specified then no security fields sent from the remote LU will be processed locally, and are ignored if present in an incoming transaction request. This is the default. Only transactions that are defined with a SECURITY level of NONE (or SAME) in the TCT can be executed as unsecured conversations from this remote LU.

If USERPSWD is specified, then incoming user IDs and passwords will be accepted from the remote LU and passed to the local security system for verification. The 'already verified' indicator is not accepted.

If USER is specified, then the USERPSWD option is assumed, plus support for the *already verified* indicator. If the *already verified* indicator is set, then a request for authorization is still made to the local security exit.

PARSESS={ NO | YES }

Specifies whether or not the target system has parallel session support or not. If PARSESS=NO is specified, or defaulted, then only a single session to the target system is allowed. Otherwise the MS region will allow as many parallel sessions as are required to support the total session limit for all defined MODEs for this entry.

PV={ YES | NO }

Indicates whether or not persistent verification will be accepted from the remote LU. If SECURITY=NONE is specified (or defaulted), then PV=NO is assumed, otherwise PV=YES is assumed.

If PV=NO is specified, or defaulted, the persistent signon indicator will not be accepted for the remote system.

Examples:

```
DEFOPSET OPSET=WSOPSET MODE=WSMODE MAP=Y LOG=Y SEC=USER
DEFOPSET OPSET=NMOPSET MODE=(MODEA,MODEB) MAP=Y LOG=Y
         PARS=Y
```

Notes:

The OSCT can be viewed as describing the LU6.2 options supported by a particular product that will be connected through LU6.2 session communication. The relevant OSCT entry to be used for a given APPC link is nominated when defining the appropriate DLT entry for a given network node. For more details see the *Management Services Administrator Guide*.

See Also:

The DEFLINK (INMC), DEFMODE, REPOPSET, DELOPSET, and SHOW DEFOPSET command descriptions.

DEFTERM

Function: Predefines a terminal as 3270 or 560/20, and specifies its attributes (VOS3 systems only).

```
DEFTERM      NAME=nnnnnnnn
             TYPE={ 3270 | 560/20 }
           [ DBCS={ YES | NO } ]
           [ { COLOR | COLOUR }={ YES | NO } ]
           [ { HLIGHT | HLITE }={ YES | NO } ]
(2)
```

Use: On a VOS3 system, MS regions support connection to 3270 and 560/20 devices. As the BIND information does not uniquely identify the device type, the DEFTERM command can be used to build a table of LUnames and their corresponding types. It can also be used to specify the device attributes.

When a terminal connects to an MS region, the DEFTERM table is referenced to find the terminal type. If no matching entry exists, the MS region attempts to determine the device type from the terminal's BIND.

Operands:

NAME=*nnnnnnnn* (mandatory)
This operand indicates the name of the terminal. The name can be a specific LUname or a generic mask, using asterisks (*) as wildcards.

TYPE={ 3270 | 560/20 } (mandatory)
This operand indicates the type of device.

DBCS={ YES | NO }
This operand indicates whether the terminal is capable of displaying DBCS (Kanji) characters.

{ COLOR | COLOUR }={ YES | NO }
This operand indicates whether or not the terminal is capable of seven-color display. For 3270 devices, this option implies 3270 extended datastream support.

{ HLIGHT | HLITE }={ YES | NO }
This operand indicates whether or not the terminal is capable of extended highlighting options (blink, reverse video, and outlining). For 560/20 devices, this option implies support for 560/20DS2 datastreams.

Specify HLIGHT=NO for model 2020 devices.

Examples:

```
DEFTERM NAME=LU560002 TYPE=560/20 DBCS=NO
DEFTERM NAME=L3270* TYPE=3270
DEFTERM NAME=* TYPE=560/20 DBCS=YES COLOR=YES
```

DEFTERM

Notes: Each DEFTERM command adds an entry to the DEFTERM table. When the table is searched, the MS region matches on specific name or on the best fit amongst generic names.

Use DEFTERM NAME=* to provide a default terminal type for all terminals not otherwise defined.

If the DBCS and COLOR attributes are specified for a 3270 device, and the terminal logs on with a BIND that specifies read-partition-query is supported, the read-partition-query results will override the attributes coded on the DEFTERM.

DEFTERM commands are usually included in NMINIT or NMREADY.

See Also: The DELTERM and SHOW DEFTERM command descriptions.

DEFTRANS

Function: Defines an entry in the Transaction Control Table (TCT) used by LU6.2 services.

```
DEFTRANS      TRANSID=transid*
            [ QUAL=qualifier ]
            [ TPN=tpn* ]
            [ TYPE={ USER | SYSTEM } ]
            [ ENV={ CURRENT | DEPENDENT | BACKGROUND } |
              LUNAME=luname | LINK=linkname | DOMAIN= domain ]
            [ MODENAME=modename ]
            [ PROC=procname* [ SERVER= servername
                    [ SCOPE={ REGION | USER | SYSTEM } ] ] ]
            [ SYNC={ NONE | CONFIRM } ]
            [ SECURITY={ NONE | SAME | USER | USERPSWD } ]
            [ CMDAUTH=nn ]
            [ CONVTYPE={ MAPPED | BASIC } ]
            [ OPERTYPE={ MAPPED | GDS } ]
            [ SIGNON={ YES | NO } ]
(4)
```

Use: This command (together with the related REPTRANS and DELTRANS commands) is used to maintain the TCT. It is normally reserved for Systems Programmer use and its use is further described in the *Management Services Administrator Guide*.

Operands: **TRANSID=*transid**** **(mandatory)**

This operand identifies the locally known transaction identifier (of up to 12 characters) and must be unique for the qualifier nominated by the QUAL operand on this definition. The *transid* has meaning for both allocation requests and attach requests (see Notes below for an explanation of these terms). The *transid* corresponds to the locally known name of the LU6.2 architected TPN parameter of the ALLOCATE command.

When an NCL procedure issues an allocation request for a conversation, it must supply the *transid*. The appropriate TCT entry is then located using this information to complete the allocation request.

When an attach request is received from the remote node to start a conversation, the TCT entry is located by *tpn* but is locally known by *transid*. Thus the *transid* specifies the locally known identifier only and is the transaction identifier returned by a query operation against the current conversation, regardless of whether it was allocated locally or remotely.

A generic value can be specified for *transid*. A transaction identifier with no exact match in the TCT will be compared to any defined generic entries. When a matching entry is found it is used as the entry for that transaction identifier.

A generic *transid* is usually used in conjunction with a generic *tpn* and *procname*. See details of the TPN and PROC operands for an explanation.

Note

Transaction IDs starting with a dollar sign ($) are reserved for future use by MS regions.

QUAL=*qualifier*

This operand provides the transaction qualifier (of up to 4 characters) for this transaction identifier. If this parameter is omitted, the *transid* being defined is unqualified. Use of a transaction qualifier allows the definition of multiple transactions with the same identifier, but which are distinguished by different qualifiers. This is most useful where a given transaction has different requirements according to the conversation destination. During an allocation request the *qualifier* is derived from the DLT entry for the transaction destination, and if it is set, the TCT scanned for a *transid* with that qualification. If not found, an unqualified *transid* is accepted.

TPN=*tpn**

Applies to both locally and remotely allocated conversations, and specifies the globally known Transaction Program Name (TPN), of up to 64 characters, unique for the qualification specified, that corresponds to the locally known identifier *transid*. When completing a local allocation request the *tpn*, not the *transid*, is passed to the remote end as the LU6.2 architected TPN parameter on the ALLOCATE request.

Conversely, when a remote attach request is received by the local system, the incoming TPN is matched against this *tpn* value to select the TCT entry to complete the remote allocation request. Hence for any conversation to successfully initiate, both the local and remote ends need only agree upon the *tpn* value for any particular transaction, while local transaction identifiers can be defined in any convenient manner. If this parameter is omitted it defaults to the *transid* value.

A generic value can be specified for *tpn*. A *tpn* name with no exact match in the TCT will be compared to any defined generic entries. When a matching entry is found it is used as the entry for that *tpn* name. A generic *tpn* name is usually used in conjunction with a generic *procname*.

A hex value (for example, TPN=’12ACB0’X) can also be specified for *tpn*.

TYPE={ <u>USER</u> | SYSTEM }

Indicates whether a user transaction or a system transaction has been defined. If a system transaction is being defined then TRANSID must be one of ATTACH, CONNECT, RPC, START, CNOS, or OSCALL. This allows the default transactions attributes to be modified to suit installation requirements.

ENV={ <u>CURRENT</u> | DEPENDENT | BACKGROUND }

This operand applies to locally allocated conversations only. Its presence indicates that when no destination parameters are specified in the allocation request, the attached process will be started in the local MS region, in the same region as the requesting process, or in a background server region if ENV=BACKGROUND is specified.

If ENV=BACKGROUND is specified, then the attached process is started in a background server region. If ENV=CURRENT is specified, or defaulted, then the attached process is started as a peer of the requesting process. If ENV=DEPENDENT is specified, then the attached process is started as a dependant of the requesting process.

LUNAME=*luname*

Applies to locally allocated conversations only, and by its presence indicates the default target system to be used for this transaction. The LUNAME operand is mutually exclusive with the ENV, LINK, and DOMAIN operands.

LINK=*linkname*

Applies to locally allocated conversations only, and by its presence indicates the default APPC link to be used for this transaction.

The LINK operand is mutually exclusive with the ENV, LUNAME, and DOMAIN operands.

DOMAIN=*domain*

This operand applies to locally allocated conversations only. Its presence indicates that if no destination parameters are specified in the allocation request, the default target system is an MS/APPC system with a domain ID of *domain*. If the MS/APPC system cannot be located by domain ID (for example, where an INMC link has not been activated to the system, and hence the domain ID is unknown), the request will fail.

MODENAME=*modename*

Applies to locally allocated conversations only, and nominates the default mode name to be used for the conversation. It corresponds to the LU6.2 architected MODE parameter of the ALLOCATE verb. Together with the actual destination *luname*, the *modename* forms a unique session selection criteria. Unless a session to the remote LU is active using this *modename*, then no current session can be used for the conversation. Depending upon allocation options, a new session can be opened to the remote LU using this *modename*; otherwise the allocation will fail. If omitted, a default *modename* for the selected destination will apply.

PROC=*procname**

Applies to remotely allocated conversations only, and by its presence indicates that this TCT entry can be used to complete remote allocation requests. The nominated *procname* is the procedure name that is to service this transaction when an allocation request is received. If omitted, this TCT entry cannot be the subject of an attach request.

A generic value can be specified for *procname*. It consists of a prefix which is used, in conjunction with a generic *tpn*, to construct the name of the NCL procedure to be invoked. See the notes below for an explanation.

SERVER=*servername*

This operand is a 1- to 32-character name. It applies to remotely allocated conversations only. It indicates that the NCL process that is to service this transaction, as indicated by the PROC operand, is to be registered as a server with the name *servername*.

In the case where *servername* is already a registered name, attach processing attempts to connect the attach request to the active server.

SCOPE={ REGION |USER | <u>SYSTEM</u> }

This operand is used to provide the scope for the registration of the *servername*, and is valid only if the SERVER operand is specified. Valid scopes are REGION, USER, and SYSTEM which indicate (respectively) region-wide, user-wide, and MS system-wide scoping.

SYNC={ NONE | <u>CONFIRM</u> }

Applies to locally allocated conversations only, and nominates the default conversation synchronization options. It corresponds to the LU6.2 architected SYNC-LEVEL parameter of the ALLOCATE verb.

A value of SYNC=NONE means no synchronization is used, and the &APPC CONFIRM and CONFIRMED options cannot be used, nor can the &APPC PREPARE-TO-RECEIVE TYPE=CONFIRM and &APPC DEALLOCATE TYPE=CONFIRM options.

The default value of SYNC=CONFIRM indicates that both the &APPC CONFIRM and CONFIRMED options can be used, as well as the &APPC PREPARE-TO-RECEIVE TYPE=CONFIRM and &APPC DEALLOCATE TYPE=CONFIRM options.

This operand can be overridden by the SYNC value specified on the conversation allocation.

SECURITY={ <u>NONE</u> | SAME | USER | USERPSWD }

Applies to both locally and remotely allocated conversations, and specifies the security level to be implemented for this transaction. It corresponds to the LU6.2 architected SECURITY parameter of the ALLOCATE verb.

If SECURITY=NONE is specified (or defaulted), no security checks are requested. For a local allocation, the security access information fields are omitted from the attach header sent. For a remote allocation request, the security access information fields in the incoming attach header (if present) are ignored.

If SECURITY=SAME is specified, the allocation will assume the security profile of the request source. For a local allocation from a procedure executing in any environment where the user is not signed on to an MS region, and where the USERID and PASSWORD parameters are omitted on the allocation request, the result is as if SECURITY=NONE is specified. Where the user is signed on, or where both the USERID and PASSWORD parameters are supplied on the allocation request, the result is as if SECURITY=USER is specified.

For an attach request, if the security access fields are omitted from the attach header, then SECURITY=NONE is assumed. If a user ID is supplied with the already verified indicator set, then the allocation is accepted on behalf of that user, unless rejected by the security exit. If both the user ID and password are supplied, then they must be verified by the security exit before execution is accepted on behalf of that user.

If SECURITY=USER is specified, then for a local allocation request, a USERID and PASSWORD can be supplied on the allocation request. If omitted, the requesting user ID, which has been verified by the local security exit, is passed to the remote end with the *already verified* indicator set. If the requesting user ID is not known (that is, the user is not signed on to an MS region), the allocation request will fail. For an attach request, the incoming attach header must carry either a user ID with the *already verified* indicator set, or a user ID and password, and be approved by the local security exit; otherwise the request is rejected.

If SECURITY=USERPSWD is specified, then for a local allocation request, a USERID and PASSWORD must be supplied. If not supplied on the allocation request, then where the user has been verified by the local security exit, both the user ID and password will be passed to the remote system if known; otherwise the allocation request will fail. For an attach request, the attach header must carry the user ID and password, and be approved by the local security exit; otherwise the allocation request is rejected.

CMDAUTH=*nn*

This operand applies to attach requests only, and specifies the command authority with which the nominated NCL procedure is to be started.

The command authority of the NCL procedure will be determined in the following way:

- If the attached transaction is a secure transaction, an APPC user region is signed on and the NCL procedure executes with the user's defined command authority, and this operand is ignored.
- If the attached transaction is unsecured, it executes in the BSYS region with a command authority that is the minimum of the BSYS user command authority or the CMDAUTH specified by this operand.
- If this operand is omitted, the CMDAUTH defaults to zero.

CONVTYPE={ <u>MAPPED</u> | BASIC }

Applies to locally allocated conversations only, and specifies whether the LU6.2 conversation is to be allocated as a mapped or basic conversation type. NCL processing is designed primarily for mapped conversations and the NCL &APPC verb implements the LU6.2 Mapped Conversation verb set, so the CONVTYPE=MAPPED option is strongly recommended, and is the default. The CONVTYPE=BASIC option is used only to set an indication in the attach header that the conversation is basic where the parameter expects a basic conversation to be allocated. Note that regardless of the CONVTYPE setting, the conversation is operated according to the OPERTYPE option.

OPERTYPE={ <u>MAPPED</u> | GDS }

Indicates the manner in which the conversation is to be operated. If MAPPED is specified, or defaulted, the data sent on the conversation is placed into user application GDS variables by MS/APPC, and only data from such a GDS variable satisfies a receive. This is the usual mode of operation for a user conversation.

If GDS is specified, the NCL procedure must set valid GDS variable length and ID fields, in addition to the data, when sending, and will receive the entire variable when receiving. This mode of operation is intended to be used only when full control of the GDS variables is required by NCL, and is best used in conjunction with MDOs and Mapping Services.

SIGNON={ YES | NO }

Specifies the persistent verification option to be sent to the remote end on an allocation request. If SIGNON=NO is specified or defaulted, the attach header carries no persistent verification indicators.

If SIGNON=YES is specified, the persistent signon indicator is sent to the remote LU, unless a previous transaction for the same (USER, LU) pair specified SIGNON=YES. In this case, the already signed on indicator is sent.

This operand allows procedures to establish a persistent signon environment in the remote system in which future transactions will be targeted. This avoids the overhead of signon and signoff with each conversation when it is known that multiple conversations are conducted with the same system and user ID.

Examples:

```
DEFTRANS TRAN=DBQUERY LINK=NMB SEC=USER
DEFTRANS TRAN=DBQUERY SEC=USER PROC=NCLDBQRY
DEFTRANS TRAN=DBPUT TPN=DB01 LU=WS001 MODENAME=WSMODE +
      SEC=USER
DEFTRANS TRAN=DBP TPN=DB01 SEC=USER PROC=NCLDBPUT
DEFTRANS TRAN=XCHNGE LU=* PROC=NCLX
```

Notes:

The Transaction Control Table (TCT) defines all transactions to Management Services where one end of the conversation resides in the local system. Each TCT entry can be used to control the allocation of either, or both, locally and remotely requested conversations.

APPC allows a program to communicate with another program via a conversation. An NCL procedure starts a conversation with a *remote* transaction program by issuing an *allocation request* using the &APPC ALLOCATE verb. The main purpose of allocation is to identify the name and location of the remote program. As a result of allocation, a request to start the remote program is sent to the appropriate network node. This request is referred to as an *attach request*.

Generic values for *transid*, *tpn*, and *procname* can be used to invoke different NCL procedures using the same TCT entry. Allocation requests that match a TCT entry with generic *transid* and *tpn* will form the *tpn* name to be sent to the remote system by appending the *tpn*'s generic prefix and the *transid* given in the request, excluding its prefix. For example, a TCT entry with TRANSID=*trn** and TPN=*tp** and an allocation request for TRANSID=*trnappc*01 will result in *tpn* '*tpappc*01' being sent to the remote system.

Attach requests that match a TCT entry with generic *tpn* and *procname* will form the name of the procedure to be attached by appending the *procname*'s generic prefix and the received *tpn* name, excluding its generic prefix. For example, a TCT entry with TPN=*tp** and PROC=*x** and an attach request for *tpappc*01 will result in *xappc*01 being started.

See Also:

The DEFLINK (INMC) and SHOW DEFTRANS commands.

DELALIAS

Function: Deletes an alias entry used by the Alias Name Translation Facility of NEWS.

```
DELALIAS        NAME=alias
                NET=aliasnet
               [ TYPE= { LU | COS | MODE } ]
(4)
```

Use: This command is used to delete an alias name definition from NEWS which is used by the Alias Name Translation Facility.

Operands:

NAME=*alias*
The mandatory 1- to 8-character *alias* or *origin* name, of the type specified by the TYPE operand, which together with the network name provided by the NET operand uniquely identifies the definition to be deleted.

NET=*aliasnet*
The mandatory 1- to 8-character network identifier, which qualifies the alias name and identifies the definition to be deleted.

TYPE=LU
Specifies that the *alias* name being deleted is an LU name. This is the default.

TYPE=COS
Specifies that the *alias* name being deleted is a COS name.

TYPE=MODE
Specifies that the *alias* name being deleted is a LOGMODE name.

Examples:

```
DELALIAS  NAME=ATSO> NET=*
DELALIAS  NAME=BATCH NET=NETWKA TYPE=COS
DELALIAS  NAME=D4C32784 NET=NET* TYPE=MODE
```

Notes: See the description of the DEFALIAS command for full details on coding all operands and any restrictions that apply.

See Also: The DEFALIAS, REPALIAS, and SHOW DEFALIAS command descriptions.

The *Unicenter NetMaster Network Management for SNA Implementation and Administration Guide*.

DELCLASS

Function: Deletes RTM, SAW, session or resource classes from a previously defined RTM, SAW session or resource class.

```
DELCLASS    { SAW=sawclass | RTM=rtmclass |
              SESSION=sessclass | RESOURCE=rscclass }
(4)
```

Use: This command is used to delete a specific SAW, RTM, Session or Resource class definition.

Operands:

SAW=*sawclass*
Specifies the name of the SAW class definition that is to be deleted.

RTM=*rtmclass*
Specifies the name of the RTM class definition that is to be deleted.

SESSION=*sessclass*
Specifies the name of the session class definition that is to be deleted.

RESOURCE=*rscclass*
Specifies the name of the resource class definition that is to be deleted.

Examples:

```
DELCLASS SAW=ALLCDRM
DELCLASS SESSION=TSOSESS
```

See Also: The DEFCLASS and REPCLASS commands.

DELLINK

Function: Deletes a dynamic link definition previously defined using the DEFLINK or REPLINK command.

```
DELLINK        TYPE=INMC
               { LINK=link | LUNAME=luname |  MASK=mask }

DELLINK        TYPE=APPC
               { LINK=link | LUNAME=luname | LUMASK=lumask }
(3)
```

Use: DELLINK deletes dynamic link definitions specified by an earlier DEFLINK command. Dynamic links already created using the deleted definition remain active.

Operands:

INMC

TYPE=INMC

Indicates that this DELLINK command applies to an INMC or NVC Dynamic Link Table entry. The default is TYPE=INMC so this operand is optional for INMC or NVC DLT definition maintenance.

> **Note**
> TYPE=INMC also includes NVC.

LINK=*link*

Provides the link name of the INMC DEFLINK definition, previously defined using the LINK parameter, that is to be deleted.

LUNAME=*luname*

Provides the *luname* of the INMC DEFLINK entry, previously defined using the LUNAME parameter, that is to be deleted.

MASK=*mask*

This identifies the dynamic link definition to be deleted. The mask value must match the value you specified on the corresponding DEFLINK command when the definition was created.

APPC

TYPE=APPC

Indicates that this DELLINK command applies to an APPC Dynamic Link Table entry. The default is TYPE=INMC so this operand is required for APPC DLT definition maintenance.

LINK=*link*

Provides the *link* name of the APPC DEFLINK entry, previously defined using the LINK parameter, that is to be deleted.

LUNAME=*luname*

Provides the *luname* of the APPC DEFLINK entry, previously defined using the LUNAME parameter, that is to be deleted.

LUMASK=*lumask*

Provides the *lumask* of the APPC DEFLINK entry, previously defined using the LUMASK parameter, that is to be deleted.

Examples:

```
DELLINK MASK=NMPC
DELLINK TYPE=INMC LINK=NMA
DELLINK TYPE=APPC LINK=WS01
DELLINK TYPE=APPC LU=NMPROD
```

See Also: The DEFLINK (INMC) command and the SHOW DEFLINK command.

Refer also to the chapters on INMC and APPC in the *Management Services Administrator Guide* for further information on dynamic link concepts.

DELLOGON

Function: Deletes an entry from the EASINET and MAI application table.

```
DELLOGON      { INPUT=text | APPL=applname }
(4)
```

Use: This command can be used to remove an entry or entries from the table of target applications as previously defined with the DEFLOGON command.

This command is normally intended for Systems Programmer use and is described in the *SOLVE:Access Implementation and Administration Guide*.

Operands: **INPUT=*text***
Text matching the DEFLOGON entry to be deleted from the table.

APPL=*applname*
All entries which allow logon to the application *applname* are to be deleted.

Examples:

```
DELLOGON INPUT=OP
DELLOGON APPL=CICSA
```

Notes: If an entry is deleted, this becomes immediately effective and does not require a restart of the system. The DELLOGON is only effective until the system is restarted. If the entry was added by a DEFLOGON command in the NMINIT or NMREADY procedures, the appropriate procedure should be updated also to remove the DEFLOGON command.

Currently defined entries can be displayed using the SHOW DEFLOGON command.

See Also: The SHOW DEFLOGON command.

DELMODE

Function: Deletes a Mode Control Table (MCT) entry.

```
DELMODE      MODE=mode
(4)
```

Use: The DELMODE command is used to delete an APPC MCT definition previously defined using the DEFMODE or REPMODE command. These commands are normally reserved for Systems Programmer use, and are further documented in the *Management Services Administrator Guide*.

Operands: **MODE=*mode***
Provides the name of the entry in the MCT to be deleted.

Examples:

```
DELMODE MODE=NMMODE
DELMODE MODE=LU62SNGL
```

Notes: The MCT is used by LU6.2 services.

See Also: The SHOW DEFMODE command.

DELOPSET

Function: Deletes an APPC Option Set Control Table (OSCT) entry.

```
DELOPSET     OPSET=opset
(4)
```

Use: The DELOPSET command is used to delete an APPC OSCT definition previously defined using the DEFOPSET or REPOPSET command. These commands are normally reserved for Systems Programmer use, and are further documented in the *Management Services Administrator Guide*.

Operands: **OPSET=*opset***
Provides the name of the entry in the OSCT to be deleted.

Examples:

```
DELOPSET OPSET=NMOPSET
DELOPSET OPSET=WSOPSET
```

Notes: The OSCT is used by LU6.2 services.

See Also: The SHOW DEFOPSET command.

DELTERM

Function: Deletes a terminal definition (VOS3 systems only).

```
DELTERM      NAME=nnnnnnnn
(2)
```

Use: This command deletes a terminal definition previously added by the DEFTERM command.

Operands: **NAME=*nnnnnnnn***

This operand indicates the name as previously entered on the DEFTERM command.

Examples:

```
DELTERM NAME=LU001
DELTERM NAME=*
```

See Also: The DEFTERM and SHOW DEFTERM command descriptions.

DELTRANS

Function: Deletes an APPC Transaction Control Table (TCT) entry.

```
DELTRANS      TRANSID=transid
            [ QUAL=qualifier ]
(4)
```

Use: The DELTRANS command is used to delete a TCT definition previously defined using the DEFTRANS or REPTRANS command. These commands are normally reserved for Systems Programmer use, and are further documented in the *Management Services Administrator Guide*.

Operands:

TRANSID=*transid*
Identifies the locally known transaction name, qualified by the QUAL operand value (if any), that is to be deleted from the TCT.

QUAL=*qualifier*
Provides the transaction qualifier for the transaction name being deleted. If omitted an unqualified name is assumed.

Examples:

```
DELTRANS TRAN=TP01
DELTRANS TRAN=TP01 QUAL=TEST
```

Notes: The TCT is used by LU6.2 services.

See Also: The SHOW DEFTRANS command.

DETACH

Function: Detaches a local non-SNA terminal from the SOLVE Subsystem Interface (SSI).

DETACH (3)	[*addr* | ALL]

Use: The DETACH command is used to detach a local non-SNA terminal from SOLVE SSI, or to detach all terminals from SOLVE SSI.

Operands: ***addr* | ALL**

Specifies the 3- or 4-digit device address of the device to be detached, or that all terminals attached to SOLVE SSI are to be detached.

Examples:

```
DETACH 04DF
DETACH ALL
```

See Also: The ATTACH and SHOW SSITERMS commands.

DISCONN

Function: Disconnects either the current processing region or the nominated processing region.

DISCONN (0/1)	[U={ *userid* | * } | LU={ *region* | * }]

Use: This command can be used to disconnect the terminal session from the processing region. The processing region is marked as *disconnected*. The terminal is either returned to EASINET or released from the Management Services session.

Operands:

U={ *userid* | * }

Specifies the user ID that is to be disconnected. An asterisk specifies that your own user ID is to be processed.

LU={ *region* | * }

Specifies the user region (LU name) that is to be disconnected. An asterisk specifies that the issuing region is to be disconnected (this is the default if no operands are specified).

Examples:

```
DISCONN                 - Disconnect the current region
DISCONN U=A56UNI        - Disconnect all regions for A56UNI
DISCONN LU=T024L08A     - Disconnect the region for T024L08A
```

Notes: Disconnected regions are assigned a numeric reconnection identifier. This identifier is displayed by the SHOW USERS command output and can be supplied in CANCEL and RECONN to specify processing of the disconnected region.

A disconnected region, as displayed by SHOW USERS, can be cancelled by CANCEL ID=*reconnection-id* specifically and will also be cancelled by CANCEL U=*userid.* If a user is connected a multiple number of times then all connections will be processed by DISCONN U=*userid*, or U=*.

A disconnected region continues to process normally. Terminal output is suspended until the region is reconnected with a terminal session. The RECONN command can be used to connect the current terminal session to a *disconnected region.*

An authority level of 1 is required to disconnect regions belonging to other users.

An automatic disconnection is performed whenever a session failure interrupts a signed-on user. Disconnection is also supported as a terminal inactivity time-out action.

The DISCONN command, and other forms of disconnection, are disabled when the system is in SHUTDOWN status.

See Also: The RECONN, CANCEL, and SHOW USERS command descriptions.

DNR

Function: Sets up or modifies the various parameters that the SOLVE Domain Name Resolver (DNR) needs.

```
DNR SET      [ FORCE ]
             [ MAXSTG=stg ]
             [ TIMELIM=time ]
             [ CNRSP=value ]
             [ CNFND=value ]
             [ DOMAIN=dmn-suffix ]
             [ NS=list-of-servers ]
             [ RETRY=(count,init-t/o,add-t/o,mult-t/o) ]
             [ TRACE=trace-opt ]
(4)
```

Use: You can use this command to set up the parameters for the SOLVE Domain Name Resolver (DNR). You can also use it to modify these parameters.

Operands: All operands, except SET, are optional. Values are altered only for the operands that are specified.

SET

Indicates that this command will set the DNR options.

FORCE

Indicates that the parameters are to be changed even if the TCP/IP interface is active. If a DNR SET command is issued when the TCP/IP interface is active, and FORCE is not specified, the command is ignored.

MAXSTG

Allows you to set the maximum storage (in kilobytes) to be used to cache entries.

This limit applies only to entries added as a result of name server searches. When static entries are defined, no limit applies to them.

The default at system initialization is 128. Range is 5 to 512.

TIMELIM

Allows you to set the maximum elapsed time (in seconds) that any one request may take. This time is absolute; even if the combination of name servers, retries, and timeouts has not been reached for a request, the request fails if this time is exceeded.

The default at system initialization is 30. Range is 5 to 300.

CNRSP

Allows you to control whether the DNR will cache the result of a request that had no response from any name server (within the timeouts). This is regarded as a negative response.

Setting NO or 0 disables caching of these results. Setting YES resets the value to the MS initialization default. Otherwise, setting a non-zero value causes caching of these results, for that number of seconds.

The default at system initialization is 3600. Range is 0 (meaning no) to 86400 (1 day).

CNFND

Allows you to control whether the DNR will cache the result of a request that received a response from a name server, where no answers were returned, and no authority was returned either.

Normally, when a name server does recognize a name, it returns a time to cache this negative response (which the DNR honors). However, if a name server has a generic entry in its database (for example, to return a mail exchange record for any name ending in the appropriate domain), then the return for an address query is that the DNR recognizes the name but has no information available. In this case there is no indication of how long to cache this result.

Setting NO or 0 disables caching of these results. Setting YES resets the value to the MS initialization default. Otherwise, setting a non-zero value causes caching of these results, for that number of seconds.

The default at system initialization is 3600. Range is 0 (meaning no) to 86400 (1 day).

DOMAIN

Allows you to set a domain name suffix that will be appended to the supplied search name if it has no periods in it.

For example, if DOMAIN is set to SMITH.COM and a GETHOSTBYNAME request for FRED is received, the DNR searches for FRED.SMITH.COM. However, if a GETHOSTBYNAME request for FRED.JONES is received, no suffixing is done.

The value can be from 1 to 64 characters, in valid domain-name format. No leading or trailing periods can be specified. The default at system initialization is a null value.

NS

Allows you to set the list of name servers that the DNR will contact. Up to three name servers can be specified.

The syntax of the NS operand is:

NS=NO
NS=*addr*
NS=(*addr* [,*port*] [,*addr* [,*port*] [,*addr* [,*port*]]])

Where:

addr is an IP address
port is the name server port number

The standard default port for a name server is 53. You can override the default port if necessary. Specifying NS=NO indicates that no name servers are defined. This is acceptable; however, any requests made that are not found in the cache will be given a NOT FOUND response.

RETRY

Allows you to set the retry counts and time-out strategy for name server communication. The values set here apply to any individual request that is being processed. Since the DNR uses UDP for communications, and UDP is unreliable, some sort of resend strategy is needed.

The syntax is:

`RETRY=(`*`count,init-t/o,add-t/o,mult-t/o`*`)`

The *count* value is the number of resends that will be made to a single name server. If all sends and resends time out, the DNR moves to the next name server and restarts this count and timeout(s). If no name servers are left, then the request fails (with a NO-RESPONSE reason).

The *init-t/o* value is the initial timeout (in seconds) for a single name server. When this value expires, if no response has arrived, a resend is issued.

The *add-t/o* value is the value (in seconds) to add to the current timeout to get a new timeout value for the next send.

The *mult-t/o* value is the value to multiply the current timeout (after adding *add-t/o*) to get the new timeout; that is:

```
new timeout = (current timeout + add-t/o) * mult-t/o
```

Multiplying allows you to implement exponential backoff.

Values:

Count can be from 0 to 5 (initial default is 2).
Init-t/o can be from 2 to 60 (initial default is 2).
Add-t/o can be from 0 to 60 (initial default is 0).
Mult-t/o can be from 0 (meaning *do not* multiply) to 4 (initial default is 2).
The default at system initialization is (2,2,0,2). This gives timeouts of 2, 4, and 8 seconds for a total time for any name server of 14 seconds.

If the final retry period for the final name server expires, no further sends are done. The request remains active until the request timeout value (as set by the TIMELIM parameter) expires, effectively allowing an extra time for responses from any name server to arrive. (If you do not want this, you can set the TIMELIM timeout value equal to the calculated total timeout for all name servers.)

TRACE

Allows you to activate DNR activity tracing. All trace output is sent to the MS log. You can set the following trace levels:

NO (default)	No tracing
ERRor	Trace errors only (send/rcv/bad incoming buffers)
YES	ERROR as well as trace overall flow of resolver
INTernal	YES as well as dump send/receive buffer contents

Examples: DNR SET TRACE=NO

Notes: Trace overheads are high, involving a task switch for every message. Therefore you should use the TRACE operand only if requested by system support staff.

See Also: The SHOW DNR command.

DOMAIN

Function: Defines or deletes a domain.

```
DOMAIN      { DEFINE  | REPLACE  | DELETE } DomainName
            [ VTAM= { YES | NO } ]
            [ IP= { YES | NO } ]
            [ { IPADDR=nnn.nnn.nnn.nnn | IPHOST= DomainName } ]
            [ IPPORT=nnnnn ]
            [ EPS= { YES | NO } ]
            [ XNF= { YES | NO } ]
(3)
```

Use: The DOMAIN command is used to explicitly define or delete a domain. The domain definition can then be used in the LINK START command.

Operands: **{DEFINE | REPLACE | DELETE} *DomainName***

DEFINE and REPLACE assign a domain name which uniquely defines a domain. DELETE deletes a domain name.

DomainName

The domain name must be the same as the primary ACB name of the target system. The primary ACB name is specified on the PRI= parameter in RUNSYSIN.

The domain name can be different from the link name used in the LINK START command.

VTAM= { YES | NO }

Identifies if the remote domain can be contacted with the VTAM access method. YES means that the domain name is used to identify the remote domain. NO prevents a VTAM link being started from an inbound request.

IP= { YES | NO }{IPADDR=*nnn.nnn.nnn.nnn* | IPHOST= *Domain Name* } IPPORT=*nnnnn*

Defines whether the remote domain can be contacted with the TCPIP access method. NO prevents a TCPIP link being started from an inbound request.
If IP is set to YES:

- IPADDR defines the IP address
- IPHOST defines the host name
- IPPORT defines the IP port number. If omitted this defaults to 2636.

EPS= { YES | NO }

Defines whether the remote domain can be contacted with the EPS access method. YES means that the domain name is used to identify the remote domain. NO prevents a EPS link being started from an inbound request.

XNF= { YES | NO }

Defines whether the remote domain can be contacted with the XNF access method. YES means that the domain name is used to identify the remote domain. NO prevents an XNF link being started from an inbound request.

Examples:

```
DOMAIN DEFINE SOLVPROD VTAM=YES IP=YES IPHOST=PRODSYS.SOLVE.COM
IPPORT=2636

DOMAIN DEFINE SOLVTEST VTAM=YES IP=NO
```

See Also: LINK (INMC), LINK (APPC), LINK (NVC), and SHOW DOMAINS commands.

EDB

Function: Defines, starts, stops, or deletes a connection from Management Services to the external database. It is also used to cancel the connection between the NCL procedure and the external database.

```
EDB          DEFINE=dbname
             TYPE=dbtype
           [ ID= { dbid | * } ]
           [ NOSTART ]  [ STATS | NOSTATS ]
           [ MAXTASK=number ]  [ MINTASK=number ]

EDB          START=dbname
           [ STATS | NOSTATS ]

EDB          STOP=dbname
           [ QUIESCE | IMMEDIATE | FORCE ]

EDB          DELETE=dbname

EDB          CANCEL=dbname   NCLID=nclid
(4)
```

Use: This command is used to define, start, stop, and delete Management Services connections and to cancel NCL connections.

Operands:

DEFINE=*dbname*
Specifies the external ID of the database as defined to the operating system.

TYPE=*dbtype*
Specifies the type of external database being defined. (Mandatory for the DEFINE operand.) This release of SQL for DB2 only supports DB2 type databases and restricts the number of simultaneously ACTIVE Management Services connections to one.

ID={ *dbid* | * }
Specifies the internal ID of the database being defined. This ID is used in NCL procedures to reference the database. If * is specified, then the internal database ID is *dbname*.

NOSTART
Specifies that the MS connection is to be defined, but not started.

STATS | NOSTATS
Indicates whether response time statistics are required. If STATS is specified with the START operand, then the existing statistics are not reset. If NOSTATS is specified with the START operand, then the statistics are reset to zero and no statistics are gathered.

MAXTASK=*number*

Specifies the maximum number of NCL processes that can simultaneously have open NCL connections to the database. The range is 1 to 100. The default *number* is 100.

MINTASK=*number*

Specifies the number of user subtasks in a pool of tasks, waiting for an NCL process to request a database connection. The range is 1 to 100. The default value for *number* is 5.

START=*dbname*

Starts the Management Services connection to the external database *dbname*, that was stopped or defined with the NOSTART option. That is, it starts a Management Services connection with a NOSTART or STOPPED status.

STATS | NOSTATS

Indicates whether response time statistics are required. If STATS is specified with the START operand, the existing statistics are not reset. If NOSTATS is specified with the START operand, the statistics are reset to zero and no statistics are gathered.

STOP=*dbname*

Stops the connection from Management Services to the external database *dbname*.

QUIESCE | IMMEDIATE | FORCE

Qualifies the STOP operand. QUIESCE means stop the Management Services connection after all NCL processes finish database access and all NCL connections have been closed. IMMEDIATE means attempt to immediately terminate the Management Services connection; all transactions in progress are backed out. FORCE means force-detach all NCL connections; all transactions in progress are backed out. The default is IMMEDIATE.

DELETE=*dbname*

Deletes the external database connection definition.

CANCEL=*dbname*

Terminates the connection between the NCL procedure and the external database. The connection from Management Services to the database is not cancelled.

NCLID = *nclid*

Specifies the ID of the NCL procedure whose NCL connection is being cancelled. The SHOW EDBUSER command can be used to obtain the NCLID of the process. Mandatory for the CANCEL operand.

EDB

Examples:

```
EDB DEFINE=udb2 TYPE=db2 STATS
EDB STOP=udb2 FORCE
EDB CANCEL=udb2 NCLID=1035
```

Notes: It is recommended that the NOSTATS operand be used. The CPU synchronization, caused by the STCK instructions issued in statistics gathering, decreases system performance.

See Also: The SHOW EDB, SHOW EDBSTAT, SHOW EDBUSER, and SHOW EDBUSTAT commands.

EDIT

Function: Invokes the Panel Maintenance facility from an OCS window or NCL procedure.

EDIT (0)	[*panelname* [LIB=*libname*]]

Use: The EDIT command allows an OCS operator to invoke Panel Maintenance from an OCS window directly, rather than having to select Panel Maintenance from the MODS : Primary Menu.

Operands: ***panelname***

The name of the PANEL to be edited.

LIB=*libname*

The name of the library which contains (or will contain) the panel. This library must be defined as editable on your current PATH. You can check this using the SHOW PATH command.

If not specified, the panel name is searched for in your current path. The EDIT command checks for the panel name in each library of the path in turn until it is found. If the first library in which the panel is found is editable, then you are presented with an edit screen allowing the panel to be updated. If the library is not editable, then you receive an error message. If the panel is not found, then you are presented with a blank edit screen, for the entry of the new panel into the first editable library in your current path.

Examples:

```
EDIT PANEL2
EDIT TESTPANEL LIB=PANELS
```

Notes: The EDIT command can also be issued from within NCL procedures executed from an OCS window.

If the EDIT command is entered without any operands, an NCL procedure (called EDIT) is invoked, which presents the MODS : Panel Maintenance Menu. This facility can be activated by the following command:

```
SYSPARMS CMDREPL=EDIT
```

This command would normally be placed in your NMINIT procedure.

See Also: The SHOW PATH and SHOW LIB commands.

END

Function: Terminates a paused NCL process nesting level without further processing.

END (0/2)	[ID=*nclid*]

Use: An active NCL process can be suspended during execution in several ways:

- Explicitly, by execution of an &PAUSE or &DELAY statement
- Implicitly, by systems performance control measures
- By execution of a statement that causes output, for example, &FILEPUT
- By a statement that causes the process to wait for input, for example &PANEL or &INTREAD

An END command issued from the OCS window, or from another NCL process, causes immediate termination of the suspended process nesting level without further processing. If the process was suspended in the highest level procedure, the entire process, and any dependent processes, are terminated. If the process was suspended in a nested procedure, only the nested procedure is terminated.

Operands: **ID=*nclid***

This operand is the NCL process identifier of the target process.

If the supplied ID is not associated with an NCL process running in the issuing region, a multi-region search for the target NCL process will take place, but only if SYSPARMS NCLXUSER=YES is set. (An authority level of 2 is required to do this.)

This operand does not need to be specified if there is only one NCL process active in the processing environment from which the command is issued.

Examples:

```
END
END ID=68
```

See Also: The GO, FLUSH, and SHOW NCL commands and the &PAUSE verb.

ENTER

Function: Sets the ENTER key.

ENTER [{ *text* | IMM | IMM,*text* | CONV,*text* | PREF,*text* | SUFF,*text* | RESET }] (0)

Notes: This command allows you to set the ENTER key. It is successful only if another key has already been defined by the command:

`PF`*nn* `IMM`

This redefines function key PF*nn* with the functionality of executing any text in the command line immediately the key is pressed. Four types of ENTER key definition are supported:

IMM - Immediate
Text is executed immediately when ENTER is pressed. If the ENTER key is defined as immediate with no associated text, then the text on the command line is executed when the ENTER key is pressed.

CONV - Conversational
Text associated with the ENTER key is displayed in the command line for modification. Text is executed only when a key defined as immediate with no associated text is pressed.

PREF - Prefix
Text associated with the ENTER key is prefixed to any data in the command line before being executed.

SUFF - Suffix
Text associated with the ENTER key is suffixed to any data in the command line before being executed.

If the cursor is positioned on an NRD message when the ENTER key is pressed, then the NRD message is hidden and any function associated with the ENTER key is not performed.

Operands: ***text***
Text to be associated with ENTER.

IMM
Specifies that an immediate key is being set.

CONV
Specifies that a conversational key is being set.

ENTER

PREF
Specifies that a prefix key is being set.

SUFF
Specifies that a suffix key is being set.

RESET
Specifies that the key is to be reset to an immediate key. This is the default.

Examples:

```
ENTER
ENTER IMM
ENTER D BFRUSE
ENTER PREF,SYSCMD+
ENTER SUFF,+D J,L
```

Notes:

When setting an IMMediate key, specifying accompanying text has the effect of executing the text when the key is pressed. If no text is provided, then the effect is to execute any text present in the command line when the key is pressed. This corresponds to the ENTER key's natural function. For example:

```
ENTER IMM
```

If the ENTER key is defined in the form ENTER text, then it is defaulted to the form ENTER IMM,*text*. That is, the text associated with the ENTER key is executed immediately the ENTER key is pressed. For example:

```
ENTER CLEAR
```

When setting a PREFix key, you might want to have a blank at the end of the key text, so that when it is appended to the addressed text the correct format results. This can be achieved by the addition of a plus sign (+) to the end of the key text. This plus sign (+) is interpreted as a blank when the key is processed. For example:

```
ENTER PREF,SYSCMD+
```

When setting a SUFFix key, you might want to have a blank at the start of the key text, so that when it is appended to the addressed text the correct format results. This can be achieved by the addition of a plus sign to the start of the key text. This plus sign is interpreted as a blank when the key is processed. For example:

```
ENTER SUFF,+D J,L
```

Concatenated commands can be set on a key by specification of a colon (:) as the concatenation character, rather than a semicolon (;) which is the normal concatenation character. The colons are translated to semicolons when the key text is stored. For example:

```
ENTER CLEAR:D BFRUSE:D IMS,E
```

When a conversational key is pressed, the cursor is normally positioned at the end of the command text in the command line. Precise positioning of the cursor is possible, by placing the cursor in the exact position it is required, when the text is on the command line. Thus, if you require the cursor to be positioned part way along the text, then key in the text and backspace the cursor to the required position, prior to pressing the key defined as immediate, with no associated text to set the new value. For example:

```
ENTER CONV,D NET,ID= :D IMS,E
```

When you first enter Operator Console Services, the ENTER key value is in effect until changed. This change is made either by individually entering the ENTER command, or by bulk entry from an NCL procedure using the EXEC command. Any changes you make do not affect the global ENTER key value, and remain in effect only until you exit from OCS.

The global ENTER key value is changed by including the appropriate ENTER command in the NMINIT procedure which is automatically executed during system initialization.

A description of the ENTER key setting can be obtained by a full list of all function key settings.

See Also: The EQUATE and PF commands.

EQUATE

Function: Assigns a text value to a unique string.

EQUATE	*string*
	[*equated-text*]
(0)	

Use: This command is used to tell the system that a unique 1- to 10-character string is to be interpreted as a particular text string.

This lets you abbreviate lengthy command strings as a single character, or group of characters, of your own choice (particularly those associated with ROF ROUTE commands and MAI commands):

- EQUATE commands processed during system initialization are used to replace existing MS commands with a name or string of the installation's choice.
- EQUATE commands processed during system initialization can also be defined with command strings that partially match existing commands.

Operands:

string

Any 1- to 10-character string that does not conflict with an existing command (see notes). The string cannot contain ampersands (&). A plus sign (+) can be placed at the end of the equate string. This means that a command string is replaced by the equated string only if the command entered is followed by a blank.

If the equate string begins with a minus sign (-), then this affects the response displayed when the equate is used. For example, if the following equate is set:

```
EQU -BYE LOGOFF
```

and the operator enters:

```
BYE
```

then the response that is displayed is BYE (instead of LOGOFF), followed by any system messages from the LOGOFF processing.

equated-text

Any text string, including blanks up to 130 characters maximum.

```
EQUATE # ROUTE NEWYORK+
EQUATE IMS+ MAISEND IMS+
EQUATE / MAISEND IMS /
EQUATE ? ROUTE CHICAGO MSG USER01+
EQUATE @ MSG ALL+
EQUATE P+ SYSCMD P
EQUATE IMS
```

Notes:

The substitution of equated text is performed only on the leading characters of an entered command. Substitution is not performed on occurrences of the equated text that occur within the entered command text.

The EQUATE string must end with a plus sign (+) if the equate is to be used only when the command entered matches the equated string *and* is followed by a blank. For example, if the following equate is set:

```
EQU BYE+ LOGOFF
```

and the operator enters:

```
BYEBYE
```

then the equate string is not substituted, because the command does not end with a blank. The equate is used if the operator enters:

```
BYE BYE
```

The text of an equate command should end with a plus sign if the text represents a prefix that is to be added to some other command or operand, where an intervening blank is required. The plus sign indicates that the text is to have a trailing blank. Using the example below, the operator enters:

```
EQUATE @ MSG ALL+
@GOOD MORNING
```

The system generates the command:

```
MSG ALL GOOD MORNING
```

If the plus sign had not been specified on the EQUATE command, then the system would not have included the trailing blank in the equated text, resulting in:

```
MSG ALLGOOD MORNING
```

If NCL procedures are used which include EQUATE characters that conflict with the reserved NCL control characters, then the EQUATE characters are ignored and processed according to the rules for NCL.

OCS command line usage is still valid, so avoid using asterisks (the NCL comment character), or plus signs (the highlight comment character). The suppression character, dash, is invalid in an EQUATE expression.

Global or system-wide EQUATE strings can be set by including the appropriate commands in the system initialization members. Global equates can be set at any time if the EQUATE command is issued by the BSYS user ID; you can use SUBMIT BSYS to cause BSYS to execute the EQUATE.

EQUATE

Equates included in the system initialization members can use equate strings that are identical to a unique command. In this case the EQUATE takes preference and is executed in place of the command. You can use this to force the execution of an NCL procedure in place of a normal command. These are known as REPL equates. If you use REPL equates, you cannot specify your own EQUATE to override one already defined to replace a command. Commands can also be overridden by the use of the SYSPARMS CMDREPL command, before and after initialization has completed.

Equates which are set during initialization can also be ambiguous with an existing command, but must contain a trailing plus sign *on the equated string* to remove all ambiguity.

The processing of equates can be controlled for commands entered from within an NCL procedure using the &CONTROL USRCMD/NOUSERCMD option. This option allows equate processing to be enabled and disabled. However, commands which have been replaced by a REPL equate or SYSPARMS CMDREPL, will always be detected and the original command will not be used.

Equate values set by individual users in OCS Mode are effective only for the individual user, and only for the single OCS session. Exiting from the OCS window destroys the equate values.

If a user defines an equate that matches an existing global equate, the user's equate will take preference over the global equate (unless the global equate REPlaces a command), but only for the OCS window for which it was entered. It will be reset on exit from OCS. The equates available to a user can be listed using the SHOW EQUATE command.

Local equates can be deleted by the entry of the EQUATE command, specifying the equate string to be deleted and no equated-text. A global equate can be effectively deleted for use by an individual OCS window by using this same technique. Reinstatement of a global equate, deleted in this way, is possible only by exiting and re-entering OCS.

See Also: The SHOW EQUATES and the ROUTE commands.

The *Management Services Administrator Guide* for details of command replacement EQUATES and the SYSPARMS CMDREPL operand.

EVERY

Function: Issues a command or series of commands at a specified time frequency.

```
EVERY        hh.mm.ss
           [ ENV={ PRI | CUR } ]
           [ KEEP={ MON | LOG | SYS } ]
           [ LIMIT={ NO | nnn } ]
           [ ROUTE={ userid | MON | LOG | SYS | * } ]
           [ PMSG={ YES | NO } ]
           [ TID=identifier ]
           { CMD=command text | MSG=message text | cmd text }
(0/2)
```

Use: Management Services supports the execution of commands at a specified time frequency using the EVERY command.

The format of the EVERY command must be as defined above. The first field following the command keyword must be the time in the format *hh.mm.ss*. The time is based on a 24-hour clock and the manner of entry is reasonably flexible. Some examples follow:

01.00.00	every 1 hour
.50.10	every 50 minutes and 10 seconds. 50.00 = invalid syntax - more than 24 hours
1.50	every 1 hour 50 minutes
50	every 50 hours
.50	every 50 minutes
..50	every 50 seconds

Other optional operands must be specified ahead of any command text. The command text must be last.

Operands: ***hh.mm.ss***

The frequency at which the command is to be executed. The first execution of the command will take place after the expiration of the first interval.

ENV={ PRI | CUR }

By default (ENV=PRI), the timer command will execute in the primary OCS environment of the issuing or ROUTEd user. This is true even if the EVERY command is issued from an &INTCMD environment.

ENV=CUR will cause the timer command to be executed in the environment that the EVERY command is issued. In this case, the timer is automatically purged if the environment is terminated (for example, exiting the OCS window, or issuing an &INTCLEAR [TYPE=ALL]). The ROUTE and KEEP parameters cannot be specified if ENV=CUR is specified.

KEEP={ MON | LOG | SYS }

The KEEP operand lets you avoid automatically deleting commands when a user is not logged on when execution is attempted. Three special environments are provided under whose control commands can be executed unattended.

When KEEP=MON is specified, the command will be executed under the control of the Background Monitor environment, if the original requesting user is not logged on. In this case, the results of the command will be displayed at all MONITOR terminals.

When KEEP=LOG is specified, the command will be executed under the control of the Background Logger, if the original requesting user is not logged on. In this case, the results of the command will be directed to the activity log and not sent to any terminals.

When KEEP=SYS is specified, the command will be executed under the control of the Background System environment, if the requesting user is not logged on. In this case the results of the command will be directed to the activity log and not sent to any terminals.

The KEEP and ROUTE operands can both be specified. If the target user ID as specified in the ROUTE operand is not logged on, the KEEP operand will be actioned.

LIMIT={ NO | *nnn* }

The number of times the command is to be executed prior to being automatically deleted. A limit of 1 should be specified if a command is only to be executed once, and not retained for execution on following days. If omitted, or specified as LIMIT=NO (the default), no limit will be imposed.

ROUTE={ *userid* | MON | LOG | SYS | * }

Specifies the user ID under whose control this command is to be executed. Use of this operand requires command authority level 2. This operand suppresses automatic deletion of the command when the user ID is not logged on. The command will be attempted at the specified interval; if the nominated user is not logged on the command will not be executed and it will be reset and retained for execution at the completion of another time interval.

If ROUTE=MON is specified, the command will be executed under the control of the Background Monitor. This is a special internal environment which is always active. Commands executed in this way will have all messages that result from execution of the command routed to all monitor terminals and logged.

If ROUTE=LOG or ROUTE=SYS is specified the command will be executed under the control of the Background Logger or Background System internal environment. Commands executed in this way will only have messages logged and they will not be sent to any terminals.

ROUTE=* means the user ID of the EVERY command user.

PMSG={ YES | NO }

PMSG=NO will suppress the issue of the Timer Purged message that is normally issued when the LIMIT is reached. This is useful with ENV=CUR timers.

TID=*identifier*

Allows an optional 1-12 character identifier to be assigned to this timer. The first character must be alpha or national, the rest alphanumeric or national. This identifier will be displayed in a SHOW TIMER list, and can be used in a PURGE TIMER= command. The identifier *must* be unique within all timers in the system. The values of ALL or OWN cannot be used.

CMD=*command text* | MSG=*message text* | *command text*

The text of the command or commands to be executed, or the text of a message to be issued. For commands, this should be in exactly the format that the command would normally be entered. Concatenated commands are supported by the separation of commands by a colon (:). This will be converted to the normal concatenation character, a semicolon (;) prior to execution. This operand must be the last one coded.

Examples:

```
EVERY 50 CMD=ACT NEWYORK
EVERY 1.0 CMD=D BFRUSE
EVERY 3.0 ROUTE=NM3BLOG TID=T16 CMD=CDRMS
EVERY 15 CMD=ACT NCPPROD,SCOPE=U
EVERY .50 CMD=R=NMBMON MSG ALL DON'T FORGET NEW FORMAT
```

Notes:

The ROUTE and KEEP operands require by default command authority level 2.

Timer commands are not remembered across a system restart. Therefore you should place timer commands to be executed on a permanent basis in either the NMINIT or NMREADY initialization procedures.

The KEEP, ROUTE, and LIMIT operands can be abbreviated to their first character. For example:

```
R=USER1
```

EVERY commands entered in the NMINIT and NMREADY procedures that are automatically executed during system initialization are allocated a user ID of '*xxxx*BSYS', where *xxxx* is the system short name, as specified by the NMSID execution parameter.

Such commands also bypass the command authority level checks imposed on certain operands such as KEEP. If neither ROUTE nor KEEP are specified, a default ROUTE=LOG operand will be assumed, to prevent automatic deletion.

Commands specified with ROUTE=MON, LOG or SYS will be allocated a user ID of '*xxxx*BMON', '*xxxx*BLOG' or '*xxxx*BSYS', where *xxxx* is again the system user prefix.

Certain commands are restricted from being executed by timer commands, to avoid inadvertently swamping the system by an ever increasing number of timer commands. This includes the EVERY command being sourced directly from another EVERY command. However, an EVERY command issued from an NCL procedure that was itself initiated by an AT or EVERY command is acceptable, although extreme caution should be exercised to ensure that the system is not overloaded.

It is possible to specify your own user ID in the ROUTE operand. This technique provides a means of ensuring that a timer command is not automatically deleted when you are not logged on, and is only executed when you are logged on.

Using the MSG= option allows a message to be issued instead of executing a command. The message is delivered as the command output would have been.

See Also: The AT, PURGE, and SHOW TIMER commands.

EXEC

Function: Schedules an NCL process for serial execution.

```
EXEC        { procname [ parm1 parm2 ... parmn ] |
                PROC=procname [ PARMS=parm1 parm2 ... parmn ] }
(0)
```

Use: Programs written in NCL are stored as members of a dataset known as the NCL procedure library, where individual members represent one *NCL Procedure*. Execution of an NCL program is termed an *NCL Process*. An NCL process can be invoked by the entry of a single EXEC command specifying the name for the NCL procedure to be processed.

The EXEC command can be entered from an OCS window by a user, or the command can be executed by an NCL procedure within a process in order to invoke another procedure. This is called nesting. Up to 64 levels of nesting are allowed. This gives you the ability to develop NCL logic in a structured manner.

Parameters can be passed to the target NCL procedure by entering them after the procedure name. The procedure can interrogate these parameters and take appropriate action. The parameters passed will depend upon the function of the NCL procedure.

When a procedure is scheduled for execution by an EXEC command the process will execute serially with respect to any other procedures invoked by the EXEC command within the same NCL processing environment. If another serial process is already running the new process will be queued pending completion of the first process.

A procedure within a process that issues EXEC internally is itself suspended until the nested procedure finishes its processing. Note that a nested procedure always executes immediately - it is not queued behind any other serial procedures that are waiting to run, because it is considered to be an extension of the *current* serial process.

A process invoked by an EXEC command from an OCS window is allocated a new *NCL Process Identifier*, which can be used to identify the procedure explicitly within its NCL processing environment.

If a procedure is invoked by another procedure issuing an EXEC command, the nested procedure inherits the NCL process identifier of the procedure that invoked it. Multiple nested procedures therefore do not have multiple NCL process identifiers, but are regarded as executing as part of the one process.

If a procedure is invoked by an EXEC command but is to execute as a *dependent process* (that is, EXEC is specified on an &INTCMD statement) the procedure will be executed as a new NCL process, with its own process identifier. The new process will be dependent on its originating process. That is, if the originating process ends so will all its dependent processes.

If a process issues more than one &INTCMD EXEC statement the nominated NCL processes are scheduled for serial execution as dependent processes. If the originating process ends then so do these queued dependent processes.

EXEC

Operands: **{ *procname* [*parm1 parm2 ... parmn*] |**
PROC=*procname* [PARMS=*parm1 parm2 ... parmn*] }

procname is the name of the NCL procedure to be loaded and executed. The name must follow normal naming conventions and library requirements.

The PARMS operands allow parameters to be passed to the EXECed process. If specified, the PARMS operand must be the last present. All data following the PARMS= operand is taken as a string of parameters and used to create the initial parameters (&1, &2 &ALLPARMS and so on) in the new procedure.

Examples:

```
EXEC PROC=ACQUIRE PARMS=NCP003
EXEC VTRACE
```

Notes: A sensible approach to the nesting of procedures will make it possible to execute them either individually or collectively.

The system comes supplied with an extensive NCL library that demonstrates the full range of NCL facilities.

The LIST command can be used to display the contents of an NCL procedure.

The NCLCHECK command can be used to check syntax in an NCL procedure.

In OS/VS systems, a number of different procedure libraries can be defined for any one region. The user ID definition specifies which library will be searched for the specified procedure.

In OS/VS systems the SHOW EXEC command lists the names of all NCL procedures in the NCL library.

See Also: The END, FLUSH, FSPROC, LIST, NCLCHECK, and START commands.

The *Network Control Language User's Guide* for details of NCL processing environments and the concepts of serial, parallel and dependent execution of NCL processes.

Function: Issues a VTAM modify (F) command.

```
F          { name,function | keyword | NET,ID=name,function }
(1)
```

Use: This command has three basic formats:

- The first allows entry of a VTAM network node name followed by the modify function that is required.

  An example of this is: `F TERM1,TRACE,TYPE=BUF`

  which is translated into a command with the format:
  `F NET,ID=TERM1,TRACE,TYPE=BUF`

- The second supports the entry of reserved keywords known to VTAM.

  An example of this is: `F NOTNSTAT`

  which is translated into a command with the format: `F NET,NOTNSTAT`

  The supported keywords are listed in the table below.

- The third format allows entry of normal fully expanded VTAM commands in an unmodified and untranslated format. An example of this is:

  `F NET,TRACE,TYPE=BUF,ID=TERM1`

Operands: ***name,function***
The VTAM name for the resource to be modified and the required modify function.

keyword
Under ACF/VTAM (VSE/SP system), *keyword* is one of the following:

ATTACH	CDRM	DETACH	DR
DUMP	IMR	IOPD	LL2
LOAD	MSG	MSGMOD	NEGPOLL
NOTNSTAT	POLL	SESSION	SGALIMIT
SUBTASK	SUPP	TABLE	TNSTAT
TPRINT	USERVAR		

A full explanation of these keywords is provided in the *VTAM Operation* manual.

Under ACF/VTAM (z/OS, OS/390, or VM), *keyword* is one of the following:

ALSLIST	APINGDTP	CDRM	CHKPT
CNOS	COMPRESS	CSALIMIT	CSM
DEFAULTS	DEFINE	DIRECTRY	DR
DUMP	ENCR	EXIT	IMR
IOPD	IOPURGE	LL2	LOAD
MSGLEVEL	MSGMOD	NEGPOLL	NOTNSTAT
POLL	PPOLOG	PROFILES	RESOURCE
RTP	SESSION	SECURITY	SUPP
TABLE	TGP	TNSTAT	TOPO
USERVAR	VTAMOPTS		

A full explanation of these keywords is provided in the *VTAM Operation* manual.

Under ECS/VTAM, *keyword* is one of the following:

ABBNOS	CDRM	CLSNO	DISP
DUMP	LOG	REP	SSCPLM
SUPP	SUPPLOG	SWAP	

Under VTAM-G, *keyword* is one of the following:

CSALIMIT	IOEMSUPP	LOGSUPP	NOTRACE
PLF	TRACE	TOLTEC	

NET,ID=*name,function*

Standard format VTAM command including name of resource to be modified and the modify function required.

Examples:

```
F NOTNSTAT
F IMR,ID=PU2
F TERM1,TRACE,TYPE=BUF
F NET,ID=TERM2,TRACE,TYPE=BUF
```

Notes: Standard format VTAM command usage is supported by the third format described above.

FLUSH

Function: Terminates an NCL process without further processing.

FLUSH (0/2)	[ID=*nclid*]

Use: NCL processes executing in an NCL processing region can be flushed unconditionally by a FLUSH command issued from either OCS window or from another NCL process within the same NCL processing region. FLUSH causes immediate termination of the entire NCL process, *not* just the procedure which happens to be executing at the time. Dependent NCL processes that were invoked by any procedures within the NCL process (via &INTCMD EXEC or START) will also terminate.

Operands: **ID=*nclid***

Identifies the NCL process that is to be ended. *nclid* is the NCL process identifier of the process that is the target of the command. The target process must be within the NCL processing region of the issuing user or process. If more than one NCL process is executing in the issuer's processing region the ID operand must be specified. If only one NCL process is executing when the command is issued the FLUSH command is assumed to apply to that process and the ID operand can be omitted. If the ID operand is omitted the scope of the FLUSH command is limited to the NCL processing environment of the OCS window from which the command is entered.

Examples:

```
FLUSH
FLUSH ID=68
```

Notes: Use the SHOW NCL command to identify the NCL processes executing in the OCS window's NCL processing environment.

FLUSH can be used to end any NCL process within the user's NCL processing region. If the user has a command authority high enough to allow it (compared to the opauth authority of the FLUSH command, default 2), and the SYSPARMS NCLXUSER=YES option is in effect, FLUSH can be used to end NCL processes that execute outside the user's NCL processing region.

See Also: The GO, END, and SHOW NCL commands.

The *Network Control Language User's Guide* for details of NCL processing environments and concepts of NCL execution.

FORCE

Function: Forcibly inactivates a VTAM network node.

FORCE (1)	*name*

Use: This command will build and issue a VTAM VARY command with the format:

```
V NET,INACT,F,ID=name
```

This command can be used to forcibly inactivate network resources.

Operands: ***name***

The VTAM name for the resource to be forced inactive.

Examples:

```
FORCE TERM1
FORCE LU054
```

Notes: Alternatives to this command would be:

```
V NET,INACT,ID=TERM1,F
V TERM1,INACT,F
```

See Also: The V command.

FSPROC

Function: Places an OCS window in fullscreen mode and executes a specific NCL procedure.

```
FSPROC        { procname [ parm1 parm2 ... parmn ] |
                    PROC=procname [ PARMS=parm1 parm2 ... parmn ] }
(0)
```

Use: An OCS window normally operates in line by line fashion with the displayed lines rolling from top to bottom of the screen.

If an NCL process is invoked by an EXEC or START command issued from the OCS window, the command is accepted and the NCL process scheduled for execution but the keyboard is unlocked immediately. This allows additional input (potentially) before the invoked NCL process issues an &PANEL statement to *take over* the window and place it in fullscreen mode.

The FSPROC command allows an NCL process to be invoked but prevents further input from the keyboard until the procedure issues an &PANEL. This allows synchronization between the command to execute the procedure and the procedure taking over the OCS window for its own display purposes. FSPROC is intended for use only when this synchronization is required. Use EXEC or START to invoke NCL processes that do not need synchronization with keyboard input.

Operands: **{ *procname* [*parm1 parm2 ... parmn*] |**
PROC=*procname* [PARMS=*parm1 parm2 ... parmn*] }

Specifies the name of the NCL procedure to be loaded and executed. The name must follow normal naming conventions and library requirements.

The PARMS operands allow parameters to be passed to the FSPROCed process. If specified, the PARMS operand must be the last present. All data following the PARMS= operand is taken as a string of parameters and used to create the initial parameters (&1, &2, &ALLPARMS, and so on) in the new procedure.

Examples:

```
FSPROC SNAPSHOT
FSPROC $LOGBROW
```

Notes: FSPROC issued from an OCS window as a command causes invocation of the target process as an asynchronous NCL process within the NCL process area associated with the OCS window. If FSPROC is issued from within a procedure it executes the same as an EXEC command, invoking the target procedure as a nested procedure of the original.

See Also: The EXEC and START command descriptions. See your *Network Control Language User's Guide* for details of the NCL processing environment and concepts.

FSTOP

Function: Forces an immediate shutdown of the system, terminating all user sessions regardless of the function being performed.

```
FSTOP
(3)
```

Use: Two methods of shutting down a system are provided. A quiesced shutdown can be achieved using the SHUTDOWN command. This will close the system down after all current users have logged off and stop the establishment of further sessions.

The FSTOP command will terminate the system regardless of the functions currently being performed. All users will be given a warning message of the pending shutdown, and the system will be closed in an orderly manner.

Notes: The FSTOP command cannot be cancelled or reversed after entry. However, the SHUTDOWN command does provide a CANCEL operand that allows the SHUTDOWN command to be reversed.

If the FSTOP command is to be entered from the system console the SYSOPER definition that provides the authorization for the console must have the correct command authority level.

See Also: The SHUTDOWN command.

FTS

Function: Invokes File Transmission Services from an OCS window or a procedure.

```
FTS
(0)
```

Use: The FTS command allows an OCS operator to invoke the FTS menus from an OCS window directly, rather than having to select FTS from the primary menu.

Notes: The FTS command can also be issued from within an NCL procedure which is executed from an OCS window.

FTSINIT

Function: Modifies the status of FTS initiators.

```
FTSINIT        destination
               { [ ENABLE={ YES | NO } ] | [ MAXIN=max ] |
                     [ STOP={ init | ALL } ] | [ START={ init | ALL } ] |
                     [ MOD=init ] }
               [ CLASS=list ]
               [ HI={ YES | NO } ]
               [ PS=sess ]
(0)
```

Use: FTSINIT can be used to start, stop or modify FTS initiators for a given destination. It can also be used to enable or disable FTS to a destination, or to set the maximum number of concurrent incoming transmissions accepted from a destination.

Operands: ***destination***

The name of the remote system to which these initiator changes apply.

ENABLE={ YES | NO }

Enables or disables FTS to this destination. ENABLE=YES means that FTS will attempt to schedule transmission requests to this destination if any are available, and will accept inbound transmissions. ENABLE=NO causes FTS to be disabled for this destination. Active transmission requests, both inbound and outbound, will complete processing. However, no new requests will be scheduled and no new inbound transmissions from the remote system will be accepted.

MAXIN=*max*

The maximum number of concurrent inbound transmissions that this system is prepared to accept from the destination system. Valid values for *max* are 0 to 16.

Setting MAXIN=0 prevents the remote system from transmitting any datasets to this system, regardless of the number of initiators the remote system might have started. Setting MAXIN=16 allows the remote system to start up to 16 concurrent transmissions to this system. Setting the field to any intermediate value allows control over the inbound transmission count according to operational requirements.

STOP={ *init* | ALL}

STOP=*init* stops initiator number *init* to the specified destination. STOP=ALL stops all defined initiators for this destination. If a selected initiator is currently processing a request, the initiator will be drained. That is, the request will continue transmitting but no new one will be started by the initiator when the current request has ended.

START={ *init* | ALL }

START=*init* causes initiator number *init* to the specified destination to be started. START=ALL starts all defined initiators for this destination.

MOD=*init*

causes initiator number *init* to this destination to be modified as indicated by the succeeding parameters.

The following operands can be used with a START or MOD request.

CLASS=*list*

Defines the request class or classes to be serviced by this initiator. Up to 26 request classes (A-Z) can be entered. Duplicate classes are not allowed. The order in which the classes are specified, left to right, determines the order in which the initiator will be allocated work.

HI={ YES | NO }

HI=YES indicates that the initiator is allocated a higher transmission priority than normal priority (HI=NO) initiators.

PS=*sess*

Work serviced by this initiator is tied to INMC multipath session number *sess*. Valid session numbers are 1 to 8. If INMC multipath is not in use, this operand is ignored.

Examples:

```
FTSINIT SYDNEY EN=Y                  -* Enable FTS to Sydney

FTSINIT DALLAS START=2 CL=ABC HI=Y   -* Start initiator 2 to
                                     -* Dallas, high priority
                                     -* for classes A, B, and C

FTSINIT PERTH MOD=1 CL=Z             -* Change Perth's initiator
                                     -* 1 to service class Z

FTSINIT ADELAIDE STOP=ALL            -* Drain Adelaide's
                                     -* initiators
```

Notes:

Operands on this command can be abbreviated to any length which maintains their unique recognition (for example, E for ENABLE, STA for START).

FTSMOD

Function: Modifies the status of incoming or outgoing FTS transmissions.

```
FTSMOD        keyword=reqname
            [ COLD={ YES | NO } ]
            [ CLASS=x ]
            [ FROM=origin ]
(0)
```

Use: FTSMOD can be used to modify the status of transmission requests. Outgoing transmissions can be held, released, restarted, deleted or acknowledged. Incoming transmission can be cancelled, deleted or acknowledged.

Operands: ***keyword=reqname***

Specifies the type of modification and the request name (*reqname*) to be modified.

For an outgoing transmission, the following keyword values can be used:

ACK or DELETE

Causes transmission request *reqname* to be acknowledged or deleted depending on its current status. The processes of deleting and acknowledging a transmission are equivalent, so these keywords can be used interchangeably. ACK/DELETE is not valid if *reqname* is currently active. The request in this case must first be HELD.

HOLD

Causes transmission request *reqname* to be placed into HELD status. The request will not be eligible for execution until the operator releases it from HELD status. If *reqname* is active when this command is issued, the transmission is interrupted and terminated.

RELEASE

Causes transmission request *reqname* to be released from HELD status. The request will be scheduled for execution when a matching initiator becomes available. This keyword can be accompanied by optional operands COLD=YES/NO or CLASS=*x*.

RESTART

Should be used if transmission request *reqname* previously failed with an error status. Once the error condition has been rectified, a RESTART command will make *reqname* eligible for rescheduling when an appropriate initiator becomes available (unless *reqname* has also been HELD). RESTART is ignored if *reqname* is not currently flagged as *in-error*. RESTART can be accompanied by optional keywords COLD=YES/NO or CLASS=*x*.

To modify an incoming transmission, the following keywords can be used. The FROM operand MUST be coded with these keywords.

ACK or DELETE
Causes transmission request *reqname* to be acknowledged or deleted depending on its current status. The processes of deleting and acknowledging an incoming transmission are equivalent, so these keywords can be used interchangeably. ACK/DELETE is not valid if *reqname* is currently active. The request in this case must be CANCELLED.

CANCEL
Causes an active incoming transmission to be cancelled. This keyword is invalid if *reqname* is not active.

The following optional operands can be used for outgoing transmissions:

COLD={ YES | NO }
If the transmission request has previously failed or been interrupted it will normally be restartable from its last checkpoint. COLD=YES overrides the checkpoint information and forces the transmission to be coldstarted, from the beginning. This operand is only valid with RELEASE or RESTART.

CLASS=*x*
Causes the class of the request to be changed to x. This operand is only valid with RESTART or RELEASE.

The FROM= operand must be specified in order to modify an incoming transmission:

FROM=*origin*
Specifies the origin of the incoming transmission.

Notes: Operands on this command can be abbreviated to any length which maintains their unique recognition (for example, H for HOLD, REL for RELEASE).

Examples: Outgoing transmissions:

```
FTSMOD HOLD=TOSYD          -* Hold transmission request TOSYD
FTSMOD REL=TOSYD CL=Z      -* Release request TOSYD into class Z
FTSMOD RES=REPORT1 COLD=Y -* Coldstart REPORT1 after it failed
FTSMOD DEL=XYZ             -* Delete request XYZ
```

Incoming transmissions:

```
FTSMOD CAN=FILE23 FROM=NEWYORK  -* Cancel request FILE23
                                -* from New York
FTSMOD ACK=ACCTS FR=PERTH       -* Acknowledge ACCTS from
                                -* Perth
```

GO

Function: Resumes processing for a paused NCL process.

```
GO                ID=nclid
                [ [ DATA= ] parm parm ... parm ]
(0/2)
```

Use: NCL processes that are invoked with the EXEC or START commands can be suspended during execution by execution of an &PAUSE or &DELAY statement. A GO command issued from the OCS window or from another NCL process within the same NCL Processing Region causes the suspended process to resume processing. Parameters can be passed to the process by entering them after the process identifier. Since GO acts across the user's whole NCL processing region, GO can be entered from one OCS window to satisfy a process executing in the other OCS window.

Operands:

ID=*nclid*

Identifies the NCL process that is the target of the command.

The target process must be within the NCL processing region of the issuing user or process. If more than one NCL process is executing in the issuer's processing region the ID operand must be specified. If only one NCL process is executing when the command is issued the GO command is assumed to apply to that process and the ID operand can be omitted. If ID= is omitted the scope of the GO command is limited to the NCL processing environment of the OCS window from which GO was entered.

[DATA=] *parm parm ... parm*

The parameters to be passed to the NCL process, if any.

Examples:

```
GO ID=38 YES
SUBMIT BMON GO ID=56
```

Notes: Use the SHOW NCL command to determine the identifiers of the NCL processes executing in your NCL processing region.

The SUBMIT command can be used to say GO to a process executing under a background environment.

The GO command *cannot* be used to satisfy an outstanding &INTREAD statement issued by an NCL procedure. To do this, use the INTQUE command.

GO can be used to communicate with any NCL process in the system (that is, processes outside the user's NCL processing region) if the user has a command authority that matches the operating authority of the GO command and the SYSPARMS NCLXUSER=YES option is in effect.

See Also: The END, FLUSH, INTQUE, and SUBMIT command descriptions.

INACT

Function: Inactivates a VTAM network node.

INACT (1)	*name*

Use: This command will build and issue a VTAM VARY command with the format:

```
V NET,INACT,ID=name
```

This command can be used to deactivate network resources.

Operands: ***name***

The VTAM name for the resource to be varied inactive.

Examples:

```
INACT NCP001
INACT LU054,I
```

Notes: Alternatives to this command would be:

```
V NET,INACT,ID=NCP001
V NCP001,INACT
```

See Also: The V command.

INTQUE

Function: Passes data to an NCL procedure's &INTREAD statement.

```
INTQUE      [ ID=nclid ]
            [ TYPE={ RESP | REQ } ]
            [ [ DATA= ] parm parm ... parm ]
(0/2)
```

Use: This command allows an OCS operator or NCL process to communicate with other NCL processes in the same NCL processing region by passing data to an &INTREAD statement issued by the target process. If the target NCL process does not have an &INTREAD outstanding, INTQUE will queue the data to the target process pending its issuing an &INTREAD to accept the data. Since INTQUE operates within the user's whole NCL processing region, INTQUE entered from one OCS window can be used to communicate with an NCL process executing in the other OCS window's NCL environment.

If the SYSPARMS NCLXUSER=YES option is in force, an authority level of 2 is required to use INTQUE to communicate with processes in other NCL processing regions.

Operands:

ID=*nclid*

The NCL process identifier of the target process. This is not needed if there is only one process executing in the NCL processing region.

TYPE=RESP | REQ

Specifies that the data passed to the target process is to be placed on the process's *dependent response queue* or the *dependent request queue* (the default). The choice will depend upon the target procedure's use of the &INTREAD statement and whether it specifies one particular message flow.

[DATA=] *parm parm ... parm*

Parameters to be passed to the target process.

Examples:

```
INTQUE ID=208 DATA=MONDAY JOB1
INTQUE ID=129 TYPE=RESP DATA=RECOVER CICS
```

Notes: The target process might or might not have an &INTREAD outstanding when the INTQUE command is issued. INTQUE can therefore be issued to prime the internal command environment of the target process before it issues its &INTREAD statement. The only restriction is that the target process must be executing when the INTQUE command is issued.

Use the SHOW NCL command to determine the processing identifiers of the NCL processes executing in your window's NCL processing region.

See Also: The SHOW NCL, GO, END, and FLUSH command descriptions.
The &INTREAD verb description in the *Network Control Language Reference*.
The *Network Control Language User's Guide* for details and concepts of NCL Processing Environments.

Function: Changes the status of ISR (Inter System Routing) conversations.

```
ISR          { ENABLE [=class-list ] | DISABLE [ =class-list ] |
                 ALTER [ =class-list ] | RESET }
             [ ALL | DEFAULT | LINK=linkname | INACTIVE ]
             [ COMPRESS= { * | YES | NO | INBOUND | OUTBOUND } ]
             [ QMAXK=nnnn ]
             [ SOLICIT={ YES | NO } ]
             [ UNSOLICIT={ NO | YES | ALL | INBOUND | OUTBOUND } ]
             [ OPTIONS  options ]
(2)
```

Use: This command can be used to change the status of ISR conversations between systems. The nominated or defaulted link(s) need not be active.

The PPO options allow the alteration of PPO-specific ISR status. These options control the automatic message flow and allow attribute alteration for inbound messages. ISR operands which are specific to PPO, AOM and NTS are described beginning on pages 2-166, 2-169, and 2-174 respectively. (There are no NEWS-specific operands.)

Operands:

ENABLE [=*class-list*]
Indicates you wish to enable the ISR conversation(s) for the nominated classes or links (see below).

DISABLE [=*class-list*]
Indicates you wish to disable the ISR conversation(s) for the nominated classes or links (see below).

ALTER [=*class-list*]
Indicates you wish to alter the attributes of the ISR conversation(s) for the nominated classes or links (see below) without explicitly enabling or disabling them.

RESET
Indicates you wish to reset the ISR status for all classes to the default values. The SOLICIT, UNSOLICIT and OPTIONS keywords are invalid with RESET.

class-list
Indicates the ISR conversation classes you wish to affect. If this is omitted, ALL is assumed. The format of class-list is:

`ALL` or *`class`* or (*`class`*,*`class`*,...,*`class`*) where

ALL means that all configured classes are to be processed.
class means that only the nominated class is to be processed.
(*class*,*class*,...) means that the nominated classes are to be processed.

The following are valid classes, although some are only valid if your region has the corresponding feature installed:

`PPO` - PPO message traffic

`NEWS` - NEWS (CNM) traffic

`NTS` - NTS traffic

`AOM` - AOM status, unsolicited traffic, and DOM flow

ISR operands which are specific to PPO, AOM, and NTS are described on the following pages. There are no NEWS-specific operands.

ALL

Indicates all current and future links are to be altered as requested. Both the default ISR definition and all specific ISR definitions are changed. This is the default scope.

DEFAULT

Indicates that only the default ISR definition and all active links using the default definition are to be changed.

LINK=*linkname*

Indicates that only the ISR definition for the nominated link name is to be created or changed. Note that the link need not be active or defined.

INACTIVE

Only allowed with RESET. Indicates that all specific ISR definitions for links that are currently inactive are to be removed, thus resetting them to the default values.

COMPRESS={ * | YES | NO | INBOUND | OUTBOUND }

Indicates the status of ISR traffic compression:

* means YES if the destination system is on another SSCP/NETID, otherwise * means NO (* is the default).

YES means enable compression inbound and outbound.

NO means disable compression.

INBOUND means accept incoming compressed data but do not send compressed data.

OUTBOUND means send outgoing compressed data but do not accept compressed data.

The effective compress status for an active link is determined by both ends.

Altering the compress status will not affect an active link. The new status will take affect the next time the link is started.

Note

Links between SOLVE:Netmaster V2.1 and V2.2 systems should have COMPRESS=NO.

QMAXK=*nnnn*

Specifies the outbound queue limit in kilobytes. *nnnn* must be in the range 4 to 1000. This sets an upper limit on the data (in bytes) that can be internally queued to a remote domain. Once this limit is reached, no more records will be accepted on the ISR queue, until the queue drops to 75% of QMAXK.

If QMAXK is not specified, it is set to 320 times the SYSPARM NMIQLIM value (divided by 1024 to get kilobytes).

The limit is displayed as the QMAX value on the SHOW ISR command display.

SOLICIT={ YES | NO }

Allow (YES) or disallow (NO) solicited traffic flow. The default is YES. The NTS feature uses solicited traffic.

UNSOLICIT={ NO | YES | ALL | INBOUND | OUTBOUND }

Allow (YES) or disallow (NO) unsolicited traffic flow. The default is NO. YES or ALL allows unsolicited traffic in both directions to and from your system. INBOUND only allows inbound unsolicited flow, which means that your system will accept unsolicited ISR traffic from the remote system, but will not send any. OUTBOUND allows only outbound unsolicited flow.

Note

The net unsolicited status for a conversation is determined by the UNSOLICIT setting at both ends. For example, if both ends have UNSOLICIT=INBOUND, neither will send, so the net result is NO.

OPTIONS *options*

Only allowed with ENABLE/DISABLE/ALTER=*class* (that is, exactly one class). This operand allows class-specific options to be set or altered. The OPTIONS keyword must follow all other ISR command keywords, and precede all keywords specific to that class.

ISR (Options for PPO)

PPO Specific Options:

```
ISR          { ENABLE=PPO | DISABLE=PPO | ALTER=PPO }

             ( Other ISR operands...)

             [ ATTR={ ASIS | LOCAL | LINK } ]
             [ DELIVER={ DEFMSG | ( [ LOCAL ],[ REMOTE ],[ PPO ] ) | ALL } ]
             [ PPOMSG={ YES | ALL | NO | INBOUND | OUTBOUND } ]
             [ SPOMSG={ YES | ALL | NO | INBOUND | OUTBOUND } ]
             [ SPOCMD={ YES | ALL | NO | INBOUND | OUTBOUND } ]
(2)
```

Operands:

ATTR={ ASIS | LOCAL | LINK }

Indicates whether or not inbound PPO message COLOR and HLIGHT attributes are to be altered.

ASIS (default) means leave COLOR and HLIGHT as they are, that is, as set by the SYSPARMS PPOCOLOR/PPOHLIGHT in the remote system.

LOCAL means set COLOR and HLIGHT to the local SYSPARMS values.

LINK means set COLOR and HLIGHT to the value specified on the LINK START command or INMC link definition.

DELIVER={ DEFMSG | ([LOCAL],[REMOTE],[PPO]) | ALL }

Specifies override delivery options for inbound PPO messages.

Specifying DEFMSG (the default) will set DEFMSG flags for received messages according to the local DEFMSG table.

Any combination of the LOCAL, REMOTE, and PPO delivery options can be specified. If set, they imply that all inbound PPO messages will be delivered to the destinations specified. This can be useful where, for example, the domain is simply acting as an intermediate node and wishes to immediately pass all incoming PPO to another domain by specifying DELIVER=REMOTE.

ALL is shorthand for DELIVER=(LOCAL,REMOTE,PPO).

PPOMSG={ YES | ALL | NO | INBOUND | OUTBOUND }

Indicates whether PPO (unsolicited) messages will be sent or received:

Specifying YES | ALL (the default), indicates PPO messages will be both sent and received, as far as this system is concerned.

NO means PPO messages will not be sent or received.

INBOUND means PPO messages will be received, but not sent.

OUTBOUND means PPO messages will be sent , but not received.

SPOMSG={ YES | ALL | NO | INBOUND | OUTBOUND }
Indicates whether SPO (solicited) messages will be sent or received (copies of command responses):

Specifying YES | ALL (the default), indicates SPO messages will be both sent and received, as far as this system is concerned.

NO means SPO messages will not be sent or received.

INBOUND means SPO messages will be received, but not sent.

OUTBOUND means SPO messages will be sent, but not received.

SPOCMD={ YES | ALL | NO | INBOUND | OUTBOUND }
Indicates whether SPO (solicited) command copies will be sent or received:

Specifying YES | ALL (the default), indicates SPO commands will be both sent and received, as far as this system is concerned.

NO means SPO commands will not be sent or received.

INBOUND means SPO commands will be received, but not sent.

OUTBOUND means SPO commands will be sent , but not received.

Examples:

```
ISR DISABLE
ISR ENABLE=AOM UNSOLICIT=YES
ISR ENABLE=PPO UNSOLICIT=IN OPTIONS ATTR=LOCAL DELIVER=DEFMSG
```

Notes:

For security, each system communicating across an ISR link must set the appropriate ISR options.

When Management Services initializes, ISR defaults to:

```
ISR ENABLE SOLICIT=YES UNSOLICIT=NO
```

You can change this default by including an alternative ISR command in the NMINIT or NMREADY NCL procedures.

The delivery of inbound SPO messages will also be affected by SYSPARM PPOSOMSG/PPOSOCMD values. The SYSPARM settings do not, however, affect outbound SPO delivery specified on the SPOMSG and SPOCMD options to which they correspond.

ISR (Options for PPO)

All PPO message traffic is regarded as unsolicited for ISR purposes.

Since ISR defaults to UNSOLICIT=NO, no PPO traffic will flow unless (at least) an

```
ISR ALTER=PPO UNSOLICIT=YES/ALL/IN/OUT
```

command is issued.

The DEFMSG= [ASIS | LOCAL] operand is still supported for V2.1 compatibility, however DEFMSG=ASIS will be treated as DELIVER=DEFMSG, and DEFMSG=LOCAL will be treated as DELIVER =PPO.

See Also: The SHOW ISRSTATS command.

The description of the ISR facility in your *Management Services Administrator Guide*.

ISR (Options for AOM)

Function: Enables, disables, or alters the status of AOM remote system connections.

```
ISR          { ENABLE=AOM | ALTER=AOM | DISABLE=AOM }

                 (Other ISR command options)

             [ UNSOLICIT={ NO | YES | ALL I INBOUND | OUTBOUND } ]
             [ OPTIONS
                 [ LINKATTR={ NO | OPT | OPTG | FORCE } ]
                 [ LCLROUTE={ REMOTE | ASIS | route } ]
                 [ RMTROUTE={ ASIS | route } ]
                 [ NEWMSGCLASS={ ASIS | list ]
                 [ { MSGCLASS=list |
                     [ INMSGCLASS=list ]
                     [ OUTMSGCLASS=list ] } ]
                 [ {MSGLEVEL=list |
                     [ INMSGLEVEL=list ]
                     [ OUTMSGLEVEL=list ] } ]
                 [ { ROUTCDE=list |
                     [ INROUTCDE=list ]
                    [ OUTROUTCDE=list ] } ]
(2)
```

Use: The ISR command controls the status of *Inter-System Routing*. This facility of Management Services allows various other facilities of Management Services to exchange information across an INMC link. AOM is one component that makes use of ISR.

AOM will send and receive message and DOM flow across ISR links. Both ends of the link must have the AOM component installed.

Use of the ISR command is explained in the *Management Services Administrator Guide*. The following explains base ISR command options that are relevant to AOM, and the extensions to the ISR command that are related to AOM only.

Acceptable abbreviations of the AOM options are shown to the right of each option.

Operands: **ENABLE=AOM | ALTER=AOM | DISABLE=AOM**

Indicates that the status of AOM conversations across the nominated ISR link(s) is to be enabled, altered, or disabled.

You *must* indicate a class of AOM if the OPTIONS keyword, and any of the AOM-specific operands is to be specified.

ISR (Options for AOM)

other ISR command options

The ISR command has several operands relating to the scope of links that can be specified. As none of these are specific to the AOM conversation class, they are not described here. See the ISR command description in this manual for a description of these operands.

UNSOLICIT={ NO | YES | ALL | INBOUND | OUTBOUND }

This operand controls the status of *unsolicited* traffic flow across the nominated link(s), for the nominated classes. AOM ISR traffic is unsolicited.

NO indicates that AOM in this system is not prepared to send or receive traffic. This effectively disables AOM traffic on this link.

YES and ALL indicates that AOM in this system is prepared to both send and receive traffic.

INBOUND means that AOM in this system will receive traffic, but will not send any.

OUTBOUND means that AOM in this system will send traffic, but will not receive any.

The effective UNSOLICIT status across any given link is determined by both ends. Thus, if either end sets UNSOLICIT to NO, the entire link status is NO. The same would apply if both had INBOUND, or OUTBOUND. If one had YES (or ALL), and the other (say) INBOUND, only traffic outbound from the YES end would flow.

Note

Once a NRD message, or a message with DOM-TRACK=YES in effect, is sent, the eventual DOM is force-sent, even if, in the meantime, ISR commands have been issued to change the UNSOLICIT status of the link. Only if the actual AOM class was disabled, would the DOM not flow, although, in this case, a dummy DOM is generated at the receiving end as part of the DISABLE processing.

OPTIONS

This operand indicates that AOM-specific ISR command options follow. It is permitted only if ALTER=AOM, ENABLE=AOM, or DISABLE=AOM is specified on the ISR command.

The OPTIONS keyword must precede any AOM-specific keyword.

LINKATTR={ <u>NO</u> | OPT | OPTG | FORCE } (LA)

This operand controls the optional application of the LINK message color and highlight options (as specified on the LINK command or INMC multipath link definition) to *inbound* AOM message traffic.

LINKATTR=<u>NO</u>

(default) Indicates that inbound AOM message color and highlight are to be left as is.

LINKATTR=OPT

Indicates that the link color and highlight are to be applied to the inbound message(s), if an inbound message has both COLOR=NO and HLITE=NO.

LINKATTR=OPTG

Indicates that the link color and highlight are to be applied to the inbound messages, as per LINKATTR=OPT, but to treat inbound messages with a color of GREEN as if they had no color. This is useful in the case where AOM is coloring messages green, rather than leaving them with no color, since green is normally the default color displayed on a color screen when no color is specified.

LINKATTR=FORCE

Will force the link color and highlight onto inbound messages in all cases.

LCLROUTE={ REMOTE | ASIS | *route* }(LRT)

This operand controls the value of the local route option that will be assigned to inbound messages. AOM allows messages originating in a system to be assigned different *local* and *remote* route values, such as PROC and LOG. When a message is sent across an ISR link, as it arrives, it can have the LCLROUTE attribute altered based on this operand. This operand is honoured before the RMTROUTE operand when processing inbound messages.

LCLROUTE=REMOTE

(the default) Causes inbound messages to have their *local* route attribute set to the inbound *remote* route value. Thus, the value seen by local processing for the message is the value set by the sending end as the remote option. This is normally the desired behavior.

LCLROUTE=ASIS

Will cause the inbound message *local* route option to be left as is, that is the value it had set by the sending system.

LCLROUTE=*route*

Where *route* is PROC, PROCONLY, MSG, BOTH, LOG, or NO sets all inbound messages' *local* route to the nominated value. See the appendix, *Message Routing Codes*, in the *Automation Services Common User Guide* for the valid values that can be specified.

RMTROUTE={ ASIS | *route* } (RRT)

This operand controls the value of the *remote* route option that will be assigned to inbound messages.

RMTROUTE=ASIS

(the default) Indicates that inbound message *remote* route option will be left as is.

RMTROUTE=*route*

Where *route* is PROC, PROCONLY, MSG, BOTH, LOG, or NO will set all inbound messages' *remote* route to the nominated value. See the appendix, *Message Routing Codes*, in the *Automation Services Common User Guide* for the valid values that can be specified.

ISR (Options for AOM)

NEWMSGCLASS={ ASIS | *list* } (NMC)

This operand allows inbound messages to be assigned a new list of AOM ISR delivery message classes. NEWMSGCLASS=ASIS will cause the message classes on inbound messages to be left alone.

Message classes are numbered from 1 to 8. NO, (indicating none), a single value, or a list of values, including ranges, can be specified as shown:

NEWMSGCLASS=NO
NEWMSGCLASS=7
NEWMSGCLASS=(3,5,7-8)

By convention, message class 8 is used by AOM for status messages relating to AOM.

MSGCLASS=*list* |(MC)
[INMSGCLASS=*list*](IMC)
[OUTMSGCLASS=*list*](OMC)

These operands indicate the AOM ISR message classes that this system is prepared to send (MSGCLASS and OUTMSGCLASS), and receive (MSGCLASS and INMSGCLASS). Specifying MSGCLASS indicates that the send and receive MSGCLASS values are to be the same.

The effective message classes (inbound and outbound) are determined by the intersection of the relevant message classes set at each end of a link. Thus, if one system sets MSGCLASS=ALL, and the other sets MSGCLASS=3, only class 3 messages are sent (in either direction in this case). Message classes are numbered from 1 to 8. NO, (indicating none), a single value, or a list of values, including ranges, can be specified as follows:

MSGCLASS=NO
MSGCLASS=7
MSGCLASS=(3,5,7-8)

By convention, message class 8 is used by AOM for status messages.

MSGLEVEL=*list* |(ML)
[INMSGLEVEL=*list*](IML)
[OUTMSGLEVEL=*list*](OML)

These operands indicate the message levels that this system is prepared to send (MSGLEVEL and OUTMSGLEVEL), and receive (MSGLEVEL and INMSGLEVEL). Specifying MSGLEVEL indicates that the send and receive message level values are to be the same.

The effective message levels (inbound and outbound) are determined by the intersection of the relevant message levels set at each end of a link. Thus, if one system sets MSGLEVEL=(I,CE), and the other sets MSGLEVEL=CE, only CE level messages are sent (in either direction in this case).

Note

Messages with the BC (broadcast) message level are sent without performing ROUTCDE testing. They must still pass MSGLEVEL testing (for BC) at both ends.

See the appendix, *Message Levels*, in the *Automation Services Common User Guide*, for the message level values that can be specified. If more than one value is wanted, a list of values, enclosed in parentheses, can be specified.

ROUTCDE=*list* |(RC)
[INROUTCDE=*list*](IRC)
[OUTROUTCDE=*list*](ORC)

These operands indicate the routing codes that this system is prepared to send (ROUTCDE and OUTROUTCDE and receive (ROUTCDE and INROUTCDE). Specifying ROUTCDE indicates that the send and receive ROUTCDE values are to be the same.

The effective routing codes (inbound and outbound) are determined by the intersection of the relevant routing codes set at each end of a link. Thus, if one system sets ROUTCDE=(1,5,10) and the other sets ROUTCDE=(5) only messages with at least ROUTCDE 5 set are sent (in either direction in this case).

See the appendix, *Message Routing Codes*, in the *Automation Services Common User Guide*, for the routing code values that can be specified. If more than one value is wanted, a list of values, enclosed in parentheses, can be specified. Inclusive ranges of values can be specified as *a-b*.

Examples:

```
ISR ENABLE=AOM LINK=LA UNSOLICIT=OUTBOUND OPTIONS ROUTCDE=5
ISR ALTER=AOM  LINK=NY OPTIONS LINKATTR=FORCE LCLROUTE=PROC
```

Notes:

Aside from the ISR link being enabled for unsolicited traffic in the correct direction, a message will flow only if it passes MSGCLASS, MSGLEVEL, and ROUTCDE filtering. Messages with the BROADCAST (BC) message level that pass message level filtering will not be ROUTCDE tested.

Messages are never delivered to the link that originated them, or to the link that sent them to this system.

A message is queued to AOMPROC informing it of enablement or disablement of an ISR link, if the ISR command (or the LINK command, and so on) actually causes a change in status.

See Also:

The SHOW ISR command description.

ISR (Options for NTS)

Function: Enables, disables, or alters the status of remote system connections.

```
ISR          { ENABLE=NTS | ALTER=NTS | DISABLE=NTS }
             [ LINK=linkname ]
             [ OPTIONS
                 SAW={ YES | NO | INBOUND | OUTBOUND } ]
(0)
```

Use: To facilitate the operation of SAW data sharing, the ISR command supports specialized options which are valid only for NTS conversations. See the *Management Services Administrator Guide* for a full description of the Inter System Routing facility. The range and syntax of the ISR command options shown above are specific to NTS.

Operands:

ENABLE=NTS | ALTER=NTS | DISABLE=NTS
This operand is mandatory and indicates the NTS class definition to be changed.

LINK=*linkname*
Specifies that only the ISR definition for the nominated *linkname* is to be changed. Note that the link need not be active or defined.

OPTIONS
Specifies that NTS-specific parameters follow.

SAW={ YES | NO | INBOUND | OUTBOUND }
Indicates whether session awareness (SAW) data is to flow between systems across the specified links.

YES
Indicates that SAW data can flow in both directions (that is, both INBOUND and OUTBOUND from this system) on this ISR conversation.

NO
Indicates that no SAW data can flow in either direction on this ISR conversation. This is the default.

INBOUND
Indicates that SAW data can flow INBOUND to this system on this ISR conversation.

OUTBOUND
Indicates that SAW data can flow OUTBOUND to this system on this ISR conversation.

Notes: Specifying SAW=YES implies SAW=INBOUND and SAW= OUTBOUND.

More specifically, the following commands must be issued:

In the sending system:

```
ISR ENABLE=NTS LINK = receiving system name
 OPTIONS SAW=OUTBOUND or SAW=YES
```

In the receiving system:

```
ISR ENABLE=NTS LINK=sending system name
 OPTIONS SAW=INBOUND or SAW=YES
```

The links might already have been defined and active when these commands are issued. If the link is not currently active the following command must be issued in both systems to activate the ISR link:

```
LINK START=link name
```

where *link name* is the NMID of the system at the other end of the link.

There are no restrictions on which systems can share SAW data. It is possible for one NTS system to send and receive SAW data at the same time. As noted previously, however, only data derived from the local VTAM can be sent to another system. (See the *SAW Data Sharing Rules* section for details on cross-network considerations.) There is no specific order in which events have to occur for SAW data sharing to be enabled. It is possible to issue the above ISR commands before or after session awareness is started in either or both systems. SAW data sharing terminates as soon as one of the conditions required for transfer is disabled.

JOURNAL

Function: Opens the alternate NDB journal dataset and closes the current dataset to make it available for batch backup and forward recovery processing.

```
JOURNAL SWAP
(2)
```

Use: The JOURNAL command provides functions for controlling the journal datasets. If the journal option is in effect for an NDB, two VSAM entry sequenced datasets are required. This allows the NDB to write journal records to one dataset while the other is processed by batch utilities.

Operands: **SWAP**

Causes the active journal to be closed and the alternate journal to be opened. Journaling then continues on the alternate dataset.

See Also: JOURNAL operand of NDB START command.
Chapter 17 and 18 of the *Network Control Language User's Guide* for a description of how to use journaling.

Function: Clears the current OCS window.

K (0)	[{ A | ALL } | { R | RESP }]

Use: This command can be used to erase all display lines in the current OCS window. If multiple windows are operative during split screen operation only the window from which the command is entered will be cleared.

Operands:

A | ALL

Specification of ALL deletes all messages currently displayed or queued for display at the window from which the command is entered. Messages with the non-roll delete attribute are not deleted.

R | RESP

RESP indicates that all messages currently displayed are to be deleted, and those messages that are responses to earlier commands entered from this window are also to be deleted. Messages with an 'unsolicited' attribute that are queued for display are not deleted. Non-roll delete messages are never deleted by this command.

Examples:

```
K ALL
K;SHOW SESS
```

Notes: Use of this command is ideal when concatenated ahead of another command that requires the full screen to contain the output. This command can be used to ensure that the full screen is available.

If a clear function is to be performed from within an NCL procedure it is recommended that the PAGE command be used. The PAGE command is synchronous with other messages.

See Also: The CLEAR and PAGE commands.

LIBPATH

Function: Controls library path definitions.

```
LIBPATH      DEFINE  pathname
                       LIB={ libname | (libname,libname,...) }
                     [ EDIT={ ALL | NONE | libname |
                                (libname,libname,...) } ]
                     [ DESC=path description ]

LIBPATH      DELETE  pathname

LIBPATH      REPLACE  pathname
                       LIB={ libname |  (libname,libname,...) }
                     [ EDIT={ ALL | NONE | libname |
                                (libname,libname,...) } ]
                     [ DESC=path description ]
(4)
```

Use: The LIBPATH command is used to manipulate library path definitions. A number of commands are provided, executable from an OCS window, or from an NCL procedure running in either fullscreen or non-fullscreen mode.

Operands:

DEFINE
Create a new library path definition.

DELETE
Remove a library path definition.

REPLACE
Change a library path definition.

Examples:

```
LIBPATH DEFINE PROD LIB=(PROD,PANELS) DESC=A Panel Path
LIBPATH DELETE PROD
```

LIBPATH DEFINE

Function: Defines a concatenation (path) of panel libraries.

```
LIBPATH DEFINE     pathname
                    LIB={ libname | (libname,libname,...) }
                  [ EDIT={ ALL | NONE | libname |
                                   (libname,libname,...) } ]
                  [ DESC=path description ]
(4)
```

Use: This command is used to define a concatenation (search order) of panel libraries. Panel libraries are defined using the LIBRARY DEFINE command.

Operands:

pathname
the name of the path being defined.

LIB=*libname* | (*libname,libname,...*)
The list of libraries which make up the path. When a user of this path requires a panel to be loaded, the libraries are searched in the order specified to locate the panel definition. The libraries referenced need to be defined using the LIBRARY DEFINE command prior to use. LIB= must be specified.

EDIT=ALL | NONE | *libname* (*libname,libname,...*)
Determines which libraries on the path can be edited by authorized users. Note that in order to edit a library it must be defined as editable by both the LIBPATH DEFINE, and LIBRARY DEFINE commands. If not specified, all libraries on the path can be edited.

DESC=*path description*
Specifies a description to be associated with this path. If specified, this operand must be last and can contain any characters (including blanks) without the need for enclosing the data in quotes.

Examples:

```
LIBPATH DEFINE PROD    LIB=(PROD,PANELS) EDIT=NONE
LIBPATH DEFINE DEVELOP LIB=(TEST,PROD,PANELS) EDIT=TEST
LIBPATH DEFINE SYSPROG LIB=(TEST,PROD,PANELS) EDIT=ALL
LIBPATH DEFINE GERMAN  LIB=(GERMAN,PANELS) EDIT=GERMAN
```

Notes: LIBPATH DEFINE commands are normally stored in the NMINIT procedure that is automatically executed during system initialization. This ensures that the definitions are immediately available when the system becomes active.

Libraries defined using the LIBRARY DEFINE command can be concatenated together using the LIBPATH DEFINE command. A library can be included in any number of PATHS.

The LIBPATH DEFINE command can be used to create different concatenations of paths for different users. For example paths could be set up to isolate the normal users of the system from panels which are updated by NCL developers. Or different paths could be set up for people of different languages.

LIBPATH DEFINE

The SHOW PATH command can be used to display the paths which have been defined to Management Services.

The SHOW LIB command can be used to display the libraries which have been defined to Management Services.

See Also: The LIBRARY DEFINE, LIBPATH REPLACE, LIBPATH DELETE, SHOW LIB, and SHOW PATH commands.

LIBPATH DELETE

Function: Removes a panel library path definition.

LIBPATH DELETE *pathname* (4)

Use: This command is used to delete a panel path definition which was previously defined by a LIBPATH DEFINE command.

Operands: ***pathname***
The name of the path being deleted.

Examples:

```
LIBPATH DELETE GERMAN
```

Notes: If a path is deleted, this becomes effective immediately and does not require a restart of the system. Anyone using this path will no longer be able to display panels. You should check that there will be no adverse impact before deleting a path definition.

See Also: The LIBPATH DEFINE, LIBPATH REPLACE, and SHOW PATH commands.

LIBPATH REPLACE

Function: Replaces a concatenation definition (path) of panel libraries.

```
LIBPATH REPLACE    pathname
                    LIB={ libname | (libname,libname,...) }
                  [ EDIT={ ALL | NONE | libname |
                              (libname,libname,...) } ]
                  [ DESC=path description ]
(4)
```

Use: This command is used to completely replace a library path definition.

Operands: ***pathname***

The name of the path being defined.

LIB=*libname* | (*libname,libname,...*)

The list of libraries which make up the path. When a user of this path requires a panel to be loaded, the libraries are searched in the order specified to locate the panel definition. The libraries referenced need to be defined using the LIBRARY DEFINE command prior to use. LIB= must be specified.

EDIT=ALL | NONE | *libname* | (*libname,libname,...*)

Determines which libraries on the path can be edited by authorized users.

Note

In order to edit a library it must be defined as editable by both the LIBPATH DEFINE and LIBRARY DEFINE commands. If not specified, all libraries on the path can be edited.

DESC=*path description*

Specifies a description to be associated with this path. If specified, this operand must be last. It can contain any characters (including blanks) without the need for enclosing the data in quotes.

Examples:

```
LIBPATH REPLACE PROD LIB=(PROD,PANELS) EDIT=NONE
LIBPATH REPLACE DEVELOP LIB=(TEST,PROD,PANELS) EDIT=TEST
LIBPATH REPLACE SYSPROG LIB=(TEST,PROD,PANELS) EDIT=ALL
```

Notes: LIBPATH DEFINE commands are normally stored in the NMINIT procedure that is automatically executed during system initialization. This ensures that the definitions are immediately available when the system becomes active. These definitions can be replaced dynamically by the LIBPATH REPLACE command. The SHOW PATH command can be used to display the paths which have been defined to Management Services.

The SHOW LIB command can be used to display the libraries which have been defined to Management Services.

See Also: The LIBRARY DEFINE, LIBPATH DEFINE, LIBPATH DELETE, SHOW PATH, and SHOW LIB commands.

LIBRARY

Function: Controls library definitions.

```
LIBRARY      DEFINE  libname
                     [ ID=fileid ]
                     [ EDIT={ YES | NO } ]
                     [ DESC=library description ]

LIBRARY      DELETE  libname

LIBRARY      REFRESH  libname

LIBRARY       REPLACE  libname
                     [ ID=fileid ]
                     [ EDIT={ YES | NO } ]
                     [ DESC=library description ]
(4)
```

Use: The LIBRARY command is used to manipulate library definitions. A number of commands are provided, executable from an OCS window, or from an NCL procedure running in either fullscreen or non-fullscreen mode.

Operands:

DEFINE
Create a new library definition.

DELETE
Remove a library definition.

REFRESH
Refresh any in-storage panel images loaded from a library.

REPLACE
Change a library definition.

LIBRARY DEFINE

Function: Defines a panel library.

```
LIBRARY DEFINE    libname
                 [ ID=fileid ]
                 [ EDIT={ YES | NO } ]
                 [ DESC=library description ]
(4)
```

Use: This command is used to define a panel library to the system. The panel library can be used in a panel library path, defined with a LIBPATH DEFINE command.

Operands: ***libname***

The name of the library being defined.

ID=*fileid*

The file ID of the library. This is the ID specified on the UDBCTL OPEN command used to open the library for use. If ID= is not specified an ID matching the library name will be assumed.

EDIT={ YES | NO }

Determines if members in this library can be edited by authorized users. The default is YES.

DESC=*library-desc*

Specifies a description to be associated with this library. If specified, this operand must be last and can contain any characters (including blanks) without the need for enclosing the data in quotes.

Examples:

```
LIBRARY DEFINE PROD
LIBRARY DEFINE TEST ID=TESTLIB EDIT=NO
LIBRARY DEFINE DISTRIB ID=PANELS DESC=Distributed Panels
```

Notes: LIBRARY DEFINE commands are normally stored in the NMINIT procedure that is automatically executed during system initialization. This ensures that the definitions are immediately available when the system becomes active. Libraries defined using the LIBRARY DEFINE command can be logically concatenated together using the LIBPATH command. A library can be included in any number of PATHS.

Libraries defined using the LIBRARY DEFINE command need to be opened for use. The UDBCTL OPEN command can be used to open a library.

The SHOW LIB command can be used to display the libraries which have been defined to Management Services. The library named PANELS is automatically defined and opened by the system.

See Also: The LIBRARY REPLACE, LIBRARY DELETE, LIBPATH, SHOW LIB, and UDBCTL commands.

LIBRARY DELETE

Function: Removes a panel library definition as previously defined by a LIBRARY DEFINE command.

LIBRARY (4)	DELETE *libname*

Use: This command is used to delete a library definition. The file is not closed and the contents are not affected.

Operands: ***libname***
The name of the library being deleted.

Examples:

```
LIBRARY DELETE TEST
```

Notes: If a library is deleted, this becomes effective immediately and does not require a restart of the system. Any users whose path contains this library will be immediately effected. You should check that there will be no adverse impact before deleting a library definition.

LIBRARY REFRESH

Function: Refreshes in-storage copies of members of a library.

LIBRARY (0)	REFRESH	*libname*

Use: The region retains copies of frequently used panels in storage to improve performance. The LIBRARY REFRESH command is used to remove from storage any panels which might not be valid due to changes in the library.

Operands: ***libname***

The name of the library being refreshed.

Examples:

```
LIBRARY REFRESH DISPANL
```

Notes: All in-storage panels loaded on any path which contains the named library will be invalidated. This process occurs automatically when using the other LIBRARY commands, and the LIBPATH command. When a panel is updated by the EDIT command any in-storage copies are also invalidated. This command is useful when the contents of a library have changed for some other reason (for example, an external utility was used to copy new members to the panel library).

The panels which are retained in storage can be displayed with the SHOW PANELS command.

LIBRARY REPLACE

Function: Replaces a panel library definition.

```
LIBRARY REPLACE    libname
                   [ ID=fileid ]
                   [ EDIT={ YES | NO } ]
                   [ DESC=library description ]
(4)
```

Use: This command is used to replace a library definition which was previously defined using the LIBRARY DEFINE command.

Operands:

libname
The name of the library definition being replaced.

ID=*fileid*
The file ID of the library. This is the ID specified on the UDBCTL OPEN command used to open the library for use. If ID= is not specified, an ID matching the library name will be assumed.

EDIT={ YES | NO }
Determines if members in this library can be edited by authorized users. The default is YES.

DESC=*library-desc*
Specifies a description to be associated with this library. If specified, this operand must be last and can contain any characters (including blanks) without the need for enclosing the data in quotes.

Examples:

```
LIBRARY REPLACE PROD
LIBRARY REPLACE TEST ID=TESTLIB EDIT=NO
LIBRARY REPLACE DISTRIB ID=PANELS DESC=Distributed Panels
```

Notes: The effect of the LIBRARY REPLACE command is the same as if you first deleted and then redefined the library.

If no library definition exists, a new one will be created. After a library is defined it will need to be opened for use. The UDBCTL OPEN command can be used to open a library of the correct file ID.

The SHOW LIB command can be used to display the libraries which have been defined to Management Services.

LINK (APPC)

Function: The LINK command is used to start and stop an APPC link connecting the local domain with another network node via one or more LU6.2 sessions.

```
LINK        { START | STOP }=linkname
              TYPE=APPC
            [ AM=amname ]
            [ DOMAIN=DomainName ]
            [ PASSWORD=password ]
            [ DRAIN | QUIESCE | FORCE ]
(3)
```

Use: The LINK command is used to activate and deactivate APPC links that have previously been defined using the DEFLINK command. It is possible to start a single session APPC link without having previously defined the link to the local domain, provided that the *DomainName* of the remote domain is specified in the LINK START command.

Operands:

{ START | STOP }=*linkname*

Indicates the function to be performed on the link. It must be the first operand after the LINK command.

START=*linkname*

Requests the initiation of an APPC connection. The *linkname* supplied provides the name of the link concerned. If a corresponding APPC Dynamic Link Table entry has been defined using the LINK=*linkname* parameter of the DEFLINK command, then that entry is used to complete the information used in the APPC link initialization. Otherwise the DOMAIN parameter must be supplied to specify the *DomainName* of the target domain.

STOP=*linkname*

Requests the termination of an APPC connection. The *linkname* supplied provides the name of the link concerned. Termination proceeds according to the DRAIN, QUIESCE, or FORCE parameter (see below).

TYPE=APPC

Indicates that this LINK command applies to an APPC link only. As the default is TYPE=INMC, this operand is required to start and stop APPC links.

AM=*amname*

Specifies the access method to be used for the link. Can be VTAM, EPS (End Point Services), INMC, or TCPIP. If this operand is omitted, the primary access method is used.

DOMAIN=*DomainName*

Applies to LINK START only. Defines the name of the target domain to be connected by the APPC link. Using this parameter on a LINK START command provides the only way to start an APPC link without the need to have a Dynamic Link Table entry.

If the remote domain is an MS domain, then the *DomainName* should have been set by using the DOMAIN command. If the remote application is not an MS domain, then the network node name of the remote application is specified.

The domain name is the same as the primary ACB name of the target system. The primary ACB name is specified on the PRI= parameter in RUNSYSIN.

Note

The domain name can be different from the link name.

PASSWORD=*password*

Applies to LINK START only, and provides the password to be used in verifying the link connection to the target network LU. The password must be in the form of 16 hexadecimal characters, representing 8 bytes of actual data. If this operand is omitted, and the DLT has no password defined, no LU-LU security verification is performed on the link.

DRAIN | QUIESCE | FORCE

Applies to LINK STOP only, and provides the termination condition for the link.

The default DRAIN option indicates that all conversation activity is to be fully serviced before closing the link. Active conversations continue unaffected, pending conversations remain pending, and new conversations can be requested. When a conversation terminates on a session, and there are no queued requests, the session is terminated. If it is a single session link, then the link is closed; otherwise processing continues until all sessions comprising the link have terminated.

If the QUIESCE option is specified, then all active conversations continue unaffected, but pending conversations complete with an *allocation error* type return code, as do any new requests for the link. As active conversations complete, sessions are terminated, and when all sessions for the link have terminated the link is closed.

If the FORCE option is specified, then all conversations are immediately terminated. Active conversations complete with a *resource failure* type return code, and pending conversations complete with an *allocation error* type return code. Sessions are immediately terminated and the link is closed.

Examples:

```
LINK START=NMA TYPE=APPC
LINK STOP=NEWYORK TYPE=APPC
LINK STOP=NEWYORK TYPE=APPC FORCE
```

LINK (APPC)

Notes:

LINK commands defining static links to remote systems are usually included in the system initialization procedures.

For any two domains that are to be connected by static links, a LINK command is required in each system, so that each system has the other defined to it.

LINK START=*linkname* commands (without the MSGID and DOMAIN operands) can be used to re-activate a LINK definition that was stopped earlier by a LINK STOP=*linkname* command.

LINK STOP=*linkname* commands break the connection to the remote system immediately, without regard for any sessions that might exist.

LINK RESET=*linkname* commands are allowed only if the specified link has been stopped with a LINK STOP=*linkname* command. STOP processing must have completed and the link must have been reported as being inactive. The LINK RESET=*linkname* command removes the specified link definition and no further communication is possible with the remote system until a new link is defined.

Parallel and single APPC session links can be defined.

Parallel APPC links support multiple concurrent LU6.2 sessions with a remote node. To start a parallel APPC link it is necessary to define a DLT entry (using the DEFLINK command) and an associated OSCT entry (using the DEFOPSET command), specifying that the remote node supports parallel sessions.

If the LINK command with the DOMAIN=*DomainName* operand is used to start an APPC link and no DLT entry has been defined, Management Services attempts to establish a single session link to the remote node.

The LU=*luname* parameter can be used instead of DOMAIN=*DomainName*. This ensures compatibility with previous versions of MS regions.

See Also:

The DEFLINK (APPC), DEFLINK (APPC), DOMAIN, SHOW DOMAINS, and SHOW LINK command descriptions.

LINK (INMC)

Function: Defines or changes the status of the INMC link between two domains.

```
LINK        { START | STOP | RESET }=linkname
            [ TYPE=INMC ]
            [ AM=amname ]
            [ DOMAIN=DomainName ]
            [ MSGID=id ]
            [ { COLOUR | COLOR }=color ]
            [ { HLIGHT | HLITE }=highlight ]
            [ RT=nnnnnn ]
            [ DEFNAME=inmcefname ]
(3)
```

Use: Domains that are connected by using the INMC facility communicate across INMC links. These links are defined by, and can be started, stopped, and deleted by the LINK command. LINK commands are usually included in the system initialization procedures to define the remote systems with which static INMC links are to be established.

The LINK command is used to define static links and to control the status of both static and dynamic links to other domains. See the chapter *Communicating Between Domains Using INMC* in the *Management Services Administrator Guide* for information on the differences between static and dynamic links.

Operands:

{ START | STOP | RESET }=*linkname*

Indicates the function to be performed on the link. It must be the first operand after the LINK command. The *linkname* is the 1- to 12-character symbolic name, of your installation's choice, by which the remote domain is known. The name can contain any graphic character except asterisk (*) and must start with an alpha or numeric character.

START=*linkname*

Requests the initiation of an INMC connection. The *linkname* supplied provides the name of the link concerned.

STOP=*linkname*

Requests the termination of an INMC connection. The *linkname* supplied provides the name of the link concerned.

RESET=*linkname*

Removes the link definition specified as *linkname*. No further communication is possible with the remote system until a new link is defined.

LINK (INMC)

TYPE=INMC

Indicates that this LINK command applies to an INMC link only. As the default is TYPE=INMC, this operand is optional.

AM=*amname*

Specifies the access method to be used for the link. This can be VTAM, EPS (End Point Services), XNF, or TCPIP. If this operand is omitted, the primary access method is used.

DOMAIN=*DomainName*

Specifies the domain name that contains the physical connection details. This operand is valid only with the START keyword. *DomainName* is defined with the DOMAIN DEFINE command. If this operand is omitted, *DomainName* defaults to the first eight characters of *linkname*.

The domain name is the same as the primary ACB name of the target system. The primary ACB name is specified on the PRI= parameter in RUNSYSIN.

Note

The domain name can be different from the link name.

MSGID=*id*

A 1-4 character string that is to be used as a default prefix to any message received by a ROF user from the remote domain. This operand is valid only with the START keyword and defaults to the first four characters of the *linkname*. Individual OCS users that use ROF to receive messages from remote domains can choose their own prefix via the SIGNON command, overriding the default value set on the LINK command.

{ COLOUR | COLOR }=*color*
{ HLIGHT | HLITE }=*highlight*

Provides a default means of identifying messages originating from the link through extended highlighting or color displays of such messages. Color can be any one of:

```
RED, YELLOW, WHITE, BLUE, GREEN, TURQUOISE, or PINK.
```

Highlight can be any one of the following:

```
REVERSE, BLINK, or USCORE
```

Both FTS and ROF messages received from this link are highlighted as requested for terminals supporting that type of datastream. OCS users that use ROF to receive messages from remote domains can use the SIGNON command to override the default link color and highlighting options.

The default color and highlight attributes apply only to messages sourced from a remote domain that do not carry color and highlight attributes already; for example, as a result of options specified on an &WRITE statement.

RT=*nnnnn*

Specifies the required retry interval to be used when Management Services is attempting to contact the remote system following a link outage. The system default is 60 seconds.

Specify the frequency required in seconds, in the range 60 to 28800 inclusive. This allows a maximum retry interval of eight hours.

DEFNAME=*inmcefname*

Specifies the name of an INMC multipath link definition. Any APPL name in the INMC multipath link definition is ignored, and the domain name from the LINK START command is used.

If specified, the named INMC multipath link definition record is used as a model for the outbound sessions for this link.

Use of this operand allows you to set up a single, model INMC multipath definition and use it for several INMC links.

If a matching DEFLINK definition is found for this link, any specification of the DEFNAME operand overrides any DEFNAME specification on the DEFLINK definition.

Examples:

```
LINK START=NEWYORK DOMAIN=BIGAPPL AM=VTAM MSGID=NY HLITE=BLINK
LINK START=NEWYORK DOMAIN=BIGAPPL AM=TCPIP COLOR=RED HLITE=BLINK
LINK START=NEWYORK DOMAIN=BIGAPPL AM=VTAM COLOR=YELLOW RT=300
LINK STOP=NEWYORK
LINK RESET=NEWYORK
```

Notes:

LINK commands defining static links to remote domains are usually included in the system initialization procedures.

For any two domains that are to be connected by static links, a LINK command is required in each domain, so that each domain has the other defined to it.

LINK START=*linkname* commands (without the MSGID and DOMAIN operands) can be used to reactivate a LINK definition that was stopped earlier by a LINK STOP=*linkname* command.

LINK STOP=*linkname* commands break the connection to the remote domain immediately, without regard for any sessions that might exist.

LINK RESET=*linkname* commands are allowed only if the specified link has been stopped with a LINK STOP=*linkname* command. STOP processing must have completed and the link must have been reported as being inactive. The LINK RESET=*linkname* command removes the specified link definition and no further communication is possible with the remote domain until a new link is defined.

The APPLID=*name* parameter may be used instead of DOMAIN=*DomainName*. This ensures compatibility with previous versions of MS regions.

Note

For INMC multipath links, the LINK command still operates as described here, unless you have used the DEFNAME=*inmcefname* operand. However, your systems programmer will have defined the various links to other domains through the INMC multipath Link Definition Maintenance function, which is part of the Management Services System Support Communication Administration facility. In this case no operands are used on the LINK START command.

See Also: The DEFLINK (INMC), DELLINK, DOMAIN, ROUTE, SIGNON, SHOW DOMAINS, and SHOW LINK command descriptions.

LINK (NVC)

Function: Defines or changes the status of the NVC link between MS and NetView domains.

```
LINK        { START | STOP | RESET }=linkname
              TYPE=NVC
            [ SSID=NetView_ssid ]
            [ MSGID=id ]
            [ { COLOUR | COLOR }=color ]
            [ { HLIGHT | HLITE }=highlight ]
(3)
```

Use: MS domains and NetView regions can communicate across NVC links. These links are defined by, and can be started, stopped, and deleted by the LINK command. LINK commands are usually included in the system initialization procedures to define the remote systems with which static NVC links are to be established.

The LINK command is used to define static links and to control the status of both static and dynamic links to other domains. See the chapter *Communicating Between Domains Using INMC* in the *Management Services Administrator Guide* for information on the differences between static and dynamic links.

Operands:

{ START | STOP | RESET }=*linkname*
Indicates the function to be performed on the link. It must be the first operand after the LINK command. The *linkname* is the 1- to 12-character symbolic name, of your installation's choice, by which the remote domain is known. The name can contain any graphic character except asterisk (*) and must start with an alpha or numeric character.

START=*linkname*
Requests the initiation of an NVC connection. The *linkname* supplied provides the name of the link concerned.

STOP=*linkname*
Requests the termination of an NVC connection. The *linkname* supplied provides the name of the link concerned.

RESET=*linkname*
Removes the link definition specified as *linkname*. No further communication is possible with the remote domain until a new link is defined.

TYPE=NVC
Indicates that this LINK command applies to an NVC link only. As the default is TYPE=INMC, this operand is mandatory if an NVC link is required.

SSID=*NetView_ssid*
A 4-character string that specifies the first four characters of the NetView job name.

MSGID=*id*

A 1-4 character string that is to be used as a default prefix to any message received by a ROF user from the remote domain. This operand is valid only with the START keyword and defaults to the first four characters of the *linkname*. Individual OCS users that use ROF to receive messages from remote domains can choose their own prefix via the SIGNON command, overriding the default value set on the LINK command.

{ COLOUR | COLOR }=*color*
{ HLIGHT | HLITE }=*highlight*

Provides a default means of identifying messages originating from the link through extended highlighting or color displays of such messages. Color can be any one of:

```
RED, YELLOW, WHITE, BLUE, GREEN, TURQUOISE, or PINK.
```

Highlight can be any one of:

```
REVERSE, BLINK, or USCORE
```

ROF messages received from this link are highlighted as requested for terminals supporting that type of datastream. OCS users that use ROF to receive messages from remote domains can use the SIGNON command to override the default link color and highlighting options.

The default color and highlight attributes apply only to messages sourced from remote domains that do not carry color and highlight attributes already; for example, as a result of options specified on an &WRITE statement.

Examples:

```
LINK START=NETVIEW1 TYPE=NVC SSID=CNM0
LINK STOP=NETVIEW1
LINK RESET=NETVIEW1
```

Notes:

LINK commands defining static links to remote systems are usually included in the system initialization procedures.

For any two domains that are to be connected by static links, a LINK command is required in each system, so that each system has the other defined to it.

LINK START=*linkname* commands (without the MSGID and SSID operands) can be used to reactivate a LINK definition that was stopped earlier by a LINK STOP=*linkname* command.

LINK STOP=*linkname* commands break the connection to the remote system immediately, without regard for ROF sessions that might exist.

LINK RESET=*linkname* commands are allowed only if the specified link has been stopped with a LINK STOP=*linkname* command. STOP processing must have completed and the link must have been reported as being inactive. The LINK RESET=*linkname* command removes the specified link definition and no further communication is possible with the remote system until a new link is defined.

See Also:

The DEFLINK (NVC), DELLINK, ROUTE, SIGNON, and SHOW LINK command descriptions.

LIST

Function: Lists statements in a specific member of the procedure library without executing them.

```
LIST          procname | PROC= procname
            [ ID=library-name ]
(0)
```

Use: Management Services supports the definition of groups of commands and logic statements called Network Control Language (NCL) procedures. Procedures reside in a library which, in an OS/VS system, is a partitioned dataset, in a VM/SP system is a CMS minidisk or in a VSE/SP system is a private source statement library. Within Management Services this library is known as the *procedure library.*

Often it is necessary to preview the contents of an NCL procedure prior to executing it. The LIST command is provided for this purpose.

Columns 1 to 79 of each line within the procedure are displayed. When this command is invoked in an internal environment (using the NCL &INTCMD verb), all 80 columns are returned.

Operands: ***procname* | PROC=*procname* (Mandatory)**

procname is the name of the NCL procedure to be listed. The name must follow normal naming conventions and library requirements.

ID=*library-name*

The library from which the member is to be read. Library name is DDNAME in z/OS/OS/390/MSP/VOS3, FILE TYPE in VM, and SUBLIBRARY in DOS. The default is the NCL library assigned to the user.

Examples:

```
LIST ACQUIRE
LIST LOCAL
LIST LOCAL ID=MYLIB
```

Notes: In an OS/VS system a technique often employed is to concatenate the VTAM definition library after the command library. This then makes it possible for network operators to display VTAM definition members from a network terminal.

The EXEC, FSPROC, and START commands are used to execute NCL procedures in the procedure library.

The NCLCHECK command can be used to check the syntax of an NCL procedure.

In an OS/VS system a number of different procedure libraries can be defined for any one MS system. The user ID definition specifies which library will be searched for the specified procedure.

LIST

A *member browse* facility is provided with the distributed system and is available from the User Services menu. This provides full scroll and search logic for detailed examination of a procedure and is recommended in preference to the List function for examination of large procedures or where a lengthy review of a particular procedure is required.

See Also: The EXEC, FSPROC, NCLCHECK, and START command descriptions.

LOCK

Function: Interrupts the current region processing.

```
LOCK
(0)
```

Use: The LOCK command is used to prevent unauthorized access to a terminal.

Notes: The LOCK facility is used to prevent unauthorized access to a terminal. The LOCK facility can be invoked by the LOCK command, or initiated automatically by the inactivity time-out facility.

Once initiated, no further screen interaction is supported until your current password is supplied.

To resume the interrupted processing, the password must be entered as prompted by the LOCK screen. The security system will be called to validate the password. If correct, the LOCK function will terminate, allowing the interrupted processing to resume screen control. If the password is not entered correctly, the processing region can be disconnected and the user automatically logged off.

A locked terminal can be disconnected using the F3 key. The disconnect function does not cancel any currently active sessions.

The LOCK function is also supported as an action by terminal inactivity time-out.

See Also: The RECONN, CANCEL, and SHOW USERS command descriptions.

LOG

Function: Logs messages to the activity log.

LOG (0)	[*free form text to be logged*]

Use: The system generates an activity log that can be used as an aid to problem determination. The LOG command can be used to enter messages into the log for future reference. These messages are written to the log prefixed by ===> to assist in detection during log browsing.

Operands: ***free form text to be logged***
The message text to be logged.

Examples:

```
LOG REPORT TERMINAL LU054 FOR SERVICE, FAULTY KEYBOARD.
```

Notes: The message will be logged with the time, terminal and user ID of the person entering the command.

From an OCS window the maximum LOG message length is governed by the command line length. In an NCL procedure (where variables can be used to hold the data) a total of 80 characters can be written in one LOG message.

LOGSWAP

Function: Releases the current activity log and makes it available for printing. Logging continues on the next available log.

LOGSWAP [*n*] (3)

Use: The system provides an activity logging facility to record all important activity. The need could arise, to make the current log available for observation. The LOGSWAP command is provided to release the current log to make it available for printing and to commence logging on the next available log. The SHOW LOGS command displays the current status of the system's logs.

Operands: ***n***

If specified, indicates the number of the log dataset (range 1 to 9) that is to be reused or dynamically allocated for reuse. This operand applies only if the SYSPARMS LOGWRAP=YES option is specified.

Examples:

```
LOGSWAP
LOGSWAP 5
```

Notes: Depending on the Management Services implementation in your installation, logging will be performed to hardcopy, disk, or tape. The use of LOGSWAP is also dependent on the operating system under which the system is running. In an OS or VM/SP system where logging is performed to a disk or tape file, LOGSWAP closes the log and the information can then be printed using a utility program. A maximum of nine (9) logs can be used in a single execution of the system. If the current log is the last log the LOGSWAP command will be rejected unless the SYSPARMS LOGWRAP=YES option is in force, in which case logging will resume on the next available log that has been dynamically reallocated.

In VSE/SP systems or VM/SP systems logging to a printer, LOGSWAP closes the current log and the information is available for immediate printing by the operating system. There is one log, which is closed and reopened.

Logging continues to the next logical log. The reason for swapping is logged and is displayed by the SHOW LOGS command.

When a log swap is performed using the LOGSWAP command, correlating information is printed on the last page of the old log and on the first page of the new log, providing a trace from one log to the next. When a log swap is performed all monitor status terminals are notified.

LSRPOOL

Function: Sets attributes for the VSAM shared resource pool.

```
LSRPOOL     { REPLACE | UPDATE |
                RESET={ CURRENT | DEFAULT } }
            [ FORCE ]
            [ POOL=n ]
            [ KEYLEN=n ]
            [ STRNO=n ]
            [ FIXBFR={ NO | YES } ]
            [ FIXIOB={ NO | YES } ]
            [ DATA
                [  size=number ... ]
                [ { HIPERSPACE | HS } [ size=number ... ] ] ]
            [ INDEX
                [  size=number ... ]
                [ { HIPERSPACE | HS } [ size=number ... ] ] ]
            [ size=number ]
(2)
```

Use: This command allows the attributes of the VSAM local shared resource pool to be set.

Operands:

REPLACE
Indicates you wish to completely reset an existing pending LSR pool definition to the command defaults, and then apply the specified parameters.

UPDATE
Indicates you wish to update an existing pending LSR pool definition with the specified parameters. Unspecified parameters are not changed.

RESET=CURRENT
Indicates you wish to reset an existing pending LSR pool definition to the existing current (that is, in-use) pool definition.

RESET=DEFAULT
Indicates you wish to reset an existing pending LSR pool definition to the command defaults.

FORCE
Required if the nominated LSR pool is presently active.

Note
An active LSR pool is never altered. It is only upgraded to the pending definition when the number of datasets opened, using that LSR pool, drops to 0, and then another dataset is opened for that LSR pool.

POOL=*n*

Indicates the specific LSR pool to process. The default is zero. MVS/XA and MVS/ESA support 16 pools, numbered zero to 15. All other environments support only one pool, numbered zero.

KEYLEN=*n*

Indicates the largest allowable KSDS key length in the nominated LSR pool. The default is the VSAM maximum of 255. The minimum is 1.

STRNO=*n*

Indicates the maximum number of strings that will be available in the nominated LSR pool. The range is 1 to 255. The default is 12.

FIXBFR=NO | YES

Although accepted in all environments, this operand only has meaning in a z/OS, OS/390, MVS/XA, MVS/ESA, MSP, or VOS3 environment. It indicates whether or not to fix the I/O buffers in the LSR pool. The default is NO.

FIXIOB=NO | YES

Although accepted in all environments, this operand only has meaning in a z/OS, OS/390, MVS/XA, MVS/ESA, MSP, or VOS3 environment. It indicates whether or not to fix the I/O control blocks in the LSR pool. The default is NO.

DATA

Indicates that the following *size=number* operands apply to the base LSR pool. This is the default. MVS/XA and MVS/ESA support separate data and index definitions for a pool (see the INDEX operand).

INDEX

Indicates that the following *size=number* operands apply to the index pool definition. This operand is supported in MVS/XA and MVS/ESA only.

{ HIPERSPACE | HS }

Indicates that the following *size=number* operands specify hiperspace buffers for the preceding DATA or INDEX part of the LSR pool. This operand is supported in MVS/ESA only.

size=number

Indicates the size, and number, of buffers for a particular pool. Sizes allowed are .5K, 1K, 2K, 4K, and then 4K increments to 32K. Number must be zero, or 3 to 32,767, with *size* times *number* being less than 16,777,216.

In MVS/XA and MVS/ESA, additional sizes of .5K increments from .5K to 8K, and then every 2K to 32K, are allowed. Number can be zero, or 3 to 65,535. Hiperspace buffers must be 4K increments, with numbers from zero to 16,777,215.

LSRPOOL

Examples:

```
LSRPOOL UPDATE POOL=2 DATA 4k=10 INDEX 4k=8
```

sets LSR pool number 2 to include 10 4k data buffers and 8 4k index buffers.

LSRPOOL REPLACE FIXIOB=YES 4096=40 8192=40

LSRPOOL UPDATE FORCE 2048=10

Notes:

The LSRPOOL command replaces the NMDEFLSR module distributed before Version 2.1.

Three LSR definitions are maintained:

DEFAULT	containing the distributed default values.
PENDING	containing the pending values, as set or altered.
CURRENT	containing the actual values as used by an active LSR pool.

The pending definition is copied to the current definition when the first VSAM dataset is opened with the LSR attribute. The SHOW LSR command displays the default, current, and pending LSR pool definitions, as well as the active pool statistics.

See Also:

The SHOW LSR, UDBCTL, and SHOW VSAM command descriptions.

LTITLE

Function: Sets the title for display at the top of the Unicenter NetMaster logo panel.

LTITLE (4)	[*text of title* | DBCS=*dbcs-character-string*]

Use: In a multi-domain environment where more than one MS region is active, it can be difficult to distinguish which system you have logged on to when the Unicenter NetMaster logo is displayed. The LTITLE command allows the installation to define a unique 70 character identifier or title that is displayed near the top of the Unicenter NetMaster logo panel.

Operands:

text of title

Text of logo title exactly as it is required to appear.

The text will be centered on the display screen and should not be entered within quotes. The maximum length is 70 characters.

An existing title will be erased if you enter an LTITLE command without text.

DBCS=*dbcs-character-string*

The double byte character set text required as the logo title. This is the title that will be displayed on terminals enabled for DBCS operation (for example, KANJI).

The text will be centered. Maximum length is 34 (double-byte) characters.

Examples:

```
LTITLE -- CHICAGO --
```

Notes: The LTITLE command is normally included in the NMINIT procedure which is executed automatically during system initialization.

One each of the different titles can be set. The DBCS title string will be sent automatically to terminals that are operating with DBCS enabled.

If no DBCS LTITLE is set, DBCS-enabled terminals will display the standard LTITLE.

LUTRACE

Function: Traces session traffic to selected terminals.

LUTRACE (4)	{ START=*name* | STOP={ *name* | ALL } }

Use: The session traffic flowing between the MS region and one or more selected terminals can be traced dynamically, with the trace information being written to the activity log. Multiple terminals can be traced concurrently, each terminal being identified by a separate START= operand.

Session trace messages written to the log can be intercepted by LOGPROC and either analyzed in-line or written to a separate file for later processing.

Operands:

START=*name*
Specifies that tracing is to start for the nominated terminal. If there is currently no session with the terminal the request is queued and will start as soon as a session is established.

STOP=*name* | ALL
Stops the active trace for the nominated terminal, or stops ALL trace activity.

Examples:

```
LUTRACE START=TERM1
LUTRACE START=TERM1 START=TERM3 STOP=TERM8
LUTRACE STOP=ALL
```

Notes: Use SHOW LUTRACE to display current trace activity and to review traffic counts for the terminals that are being traced.

MAIDISC

Function: Forces disconnection of an MAI-OC session.

```
MAIDISC        session ID  [ TYPE=UNCOND | COND | FORCE ]
(0)
```

Use: Used to force disconnection of an MAI-OC session.

Operands: ***session ID***

The identifier allocated to the MAI-OC session. The SHOW MAI command can be used to determine the ID allocated to each session.

TYPE=UNCOND

Generate an unconditional termination of the session. Utilizes the VTAM TERMSESS TYPE=UNCOND macro. This is the default if the TYPE operand is omitted.

TYPE=COND

Generate a conditional termination of the session. Utilizes the VTAM TERMSESS TYPE=COND macro.

TYPE=FORCE

Generate a forced termination of the session. MAI-OC will simply close its VTAM ACB.

Examples:

```
MAIDISC IMS
MAIDISC TSO T=UNCOND
```

Notes: The TYPE operand can be abbreviated to T.

Exiting from Operator Console Services mode will simulate an MAIDISC TYPE=FORCE for each MAI-OC session created from that window.

All forms of MAIDISC command generate some form of *lost terminal* (LOSTERM) condition in the connected application. The effects of this LOSTERM will vary depending on the application. For instance, a conditional or forced termination when connected to TSO will terminate the session, but will leave a 'reconnect' environment in the operating system, pending the user's reconnection (this environment is retained for a period defined by your computer operations staff). An unconditional request will terminate the logon completely.

It is also possible, of course, to terminate an MAI-OC session normally by sending to the application the normal *logoff* string that would be required from a real terminal. For example, LOGOFF for TSO, /RCL for IMS.

MAIINT

Function: Interrupts an MAI-OC application.

```
MAIINT        session ID  [ TYPE=ATTN | CNCL ]
(0)
```

Use: Used to send specific interrupts on an MAI-OC session.

Operands: ***session ID***

The identifier allocated to the MAI-OC session. The SHOW MAI command can be used to determine the ID allocated to each session.

TYPE=ATTN

Generate an attention interrupt to the application. This is the equivalent of pressing the ATTN key on a keyboard. This value is the default.

TYPE=CNCL

Generate a cancel interrupt to the application. This is the equivalent of pressing the CNCL key on a 3767 keyboard.

Examples:

```
MAIINT IMS TYPE=CNCL
MAIINT TSO T=ATTN
MAIINT TSO
```

Notes: The TYPE operand can be abbreviated to T.

The effect of sending one of these interrupts on an MAI-OC session depends on the application. For example, TYPE=ATTN on a TSO session cancels the current operation (equivalent to pressing the PA1 key on a 3270 keyboard), while on an IMS session it dequeues the current message and requests the next one (equivalent to pressing the PA2 key on a 3270 keyboard). TYPE=CNCL, if issued while receiving a large amount of data on an IMS session, cancels that output. See the relevant application manual to find out the effect each type of interrupt has.

TYPE=CNCL is accepted only if entered while data is actually being received by MAI-OC.

MAILOGON

Function: Creates an MAI-OC session with an application.

```
MAILOGON     applid
             [ LU=luname ]
             [ ID=sessionID ]
             [ NL=newline ]
             [ WAIT={ NO | YES | PERM } ]
             [ COLOUR | COLOR=color ]
             [ HLIGHT=hlight ]
             [ LOGMODE=entry ]
             [ PAREN={ YES | NO } ]
             [ PREFIX={ BEFORE | AFTER | NO } ]
             [ DATA=userdata ]
(0)
```

Use: Used to request a session with another application.

Operands:

applid

The name of the target application.

LU=*luname*

The name of the logical unit on which the session is to be created. The application will *see* the session request as coming from a terminal of this name. Only certain names will be available on your system. If this operand is omitted, MAI-OC will choose a name whose format has been defined by your computer operations staff. See your computer operations staff for more information.

ID=*sessionID*

The 1 to 8 character ID which will prefix any messages received from this session, and which must be used on other MAI-OC commands to identify the session. If not specified, the APPLID will be used. If multiple sessions are to be created with the same application from the same Operator Console Services window, the second and subsequent sessions must specify this parameter so that the session can be uniquely identified.

NL=*newline*

The *New Line* character associated with this session. Whenever this character is entered within a string sent to the application, it is translated to an SCS New Line character. Only one character can be specified. If the NL operand is omitted, then no new line characters can be sent.

WAIT={ NO | YES | PERM }

Specifies the action to be taken if the session cannot be successfully created because the target application is not active, or if the session later fails.

NO
No action to be taken; session request fails.

YES
Wait for target application to acquire the session when the application starts.

PERM
If the session fails, then session environment is to be maintained; target application must acquire the session when the application restarts. This option also implies WAIT=YES.

COLOUR | COLOR=*color*
Specifies a color attribute to be associated with messages arriving on this MAI session. The color can be any one of red, white, blue, green, yellow, pink or turquoise.

HLIGHT=*hlight*
Specifies a highlighting attribute to be associated with messages arriving on this MAI session. This can be any one of REVERSE, USCORE or BLINK, providing *reverse video*, *under-scored* or *blinking highlighting* respectively. (The HLIGHT operand can also be specified as HLITE).

LOGMODE=*entry*
Specifies the name of a VTAM logmode table entry that will be used to supply the session parameters to be used on the session. If not specified, default parameters will apply.

PAREN={ YES | NO }
Specifies whether or not a session identifier that prefixes messages from the application is to be enclosed in parentheses.

PREFIX={ BEFORE | AFTER | NO }
Specifies how a session identifier is to be prefixed to messages from the application.

BEFORE
The identifier is prefixed to each message before being delivered to any MSGPROC in force. MSGPROCs are NCL procedures your installation can use to process messages.

AFTER
The identifier is prefixed to each message after MSGPROC has processed the message, but before it is delivered to the screen.

NONE
Messages are not prefixed by the session identifier.

DATA=*userdata*
Any user data to be passed to the target application at logon. If entered, this must be the last parameter, and all text following this operand will be passed as user data, including any embedded blanks. Quotation marks should not be used to delimit the data. DATA can be used, for instance, to pass user ID and other information to TSO, rather than being prompted for it.

Examples:

```
MAILOGON IMS NL=+
MAILOGON IMST N=: LU=MTO COLOR=YELLOW
MAILOGON TSO
MAILOGON TSO ID=TSO2 DATA=USER01 PROC(PROC23)
MAILOGON CICSA H=REVERSE C=RED
```

Notes:

All operands except APPLID and LOGMODE can be abbreviated to their first letter (for example, DATA=USER01 can be entered as D=USER01). LOGMODE can be abbreviated to LOG.

The WAIT=PERM operand is useful for Master Terminal-type operations. For instance, IMS can be set up to acquire the Master Terminal at startup. If an MAI-OC session is, in fact, the Master Terminal, the session can be created in advance, and can be automatically maintained.

Network Control Language (NCL) facilities allow full automation of logon procedures. For standard logons for operational areas, a single procedure can be created to establish any number of MAI-OC sessions required and simultaneously create the EQUATES to simplify the sending of data on those sessions.

Color and highlighting are an effective means of identifying messages from particular MAI-OC sessions. On monochrome terminals, the use of prefixing can be used to distinguish the message source.

See Also:

The MAIINT, MAISEND, MAIDISC, and SHOW MAI command descriptions.

MAISEND

Function: Sends data to an application connected by an MAI-OC session.

MAISEND *session ID* [*data*] (0)

Use: Used to send data on an MAI-OC session created by a previous MAILOGON command. *session ID*
The identifier allocated to the MAI-OC session. The SHOW MAI command can be used to determine the ID allocated to each session.

data
The data to be sent to the application. If only the equivalent of the ENTER key is required, then no data need be entered. Data entered is positional, starting at the second character after the session ID. Leading blanks are preserved.

Examples:

```
MAISEND IMS /DIS A
MAISEND IMST AXS DBRVVDB
MAISEND TSO LISTC
MAISEND TSO2
```

Notes: You should consider using the EQUATE command to define an equate for commonly used MAISEND commands:

```
EQU / MAISEND IMS /
```

This equate would allow entry of IMS commands in their standard format.

See Also: The MAILOGON, MAIINT, MAIDISC, and SHOW MAI command descriptions.

MAISESS

Function: Provides functions for the control of MAI-FS sessions.

```
MAISESS   BUILD

MAISESS   CANCEL[ ={ UNCOND | FORCE | COND } ]
          { ID=session-id [ REGION={ luname | userid } ] |
              RELNUM=nnnn }

MAISESS   DELETE
          { ID=session-id | RELNUM=nnnn }

MAISESS   MODIFY
          { ID=session-id | RELNUM=nnnn }
          { [ COMPRESS={ YES | NO } ]
            [ FASTJUMP={ YES | NO } ]
            [ RESTART={ YES | NO } ]
            [ MENU1=[ key-name ] ]
            [ MENU2=[ key-name ] ]
            [ SWAP1=[ key-name ] ]
            [ SWAP2=[ key-name ] ]
            [ FWDJUMP1=[ key-name ] ]
            [ FWDJUMP2=[ key-name ] ]
            [ REVJUMP1=[ key-name ] ]
            [ REVJUMP2=[ key-name ] ] }

MAISESS   START
          { ID=session-id | RELNUM=nnnn }
(0/4)
```

Use: Used to control the operation of MAI-FS sessions.

Examples:

```
MAISESS BUILD
MAISESS START ID=TSOP
MAISESS START RELNUM=1
MAISESS CANCEL=COND ID=IMST
MAISESS CANCEL ID=CICSP
MAISESS DELETE ID=NM1
MAISESS MODIFY ID=SESSA FWDJUMP1=F22 MENU1=ATTN
MAISESS MODIFY ID=DBM COMPRESS=YES FASTJUMP=NO SWAP1=
```

Notes: The MAISESS command provides flexibility in the creation and management of MAI-FS sessions. It enables sessions to be established and controlled automatically without the need for terminal operator involvement.

MAISESS can be used from NCL procedures to control MAI-FS sessions.

MAISESS

MAISESS enables non-terminal owning regions to establish and control MAI-FS sessions, thus widening the scope of information sources for system/network management by background tasks. These sessions should be defined with a specific logmode, as no terminal default exists.

When no MAI-FS sessions are defined, an MAISESS START request results in the MAI-FS session list being built from the user's MAI-FS profile (which is stored on the system VFS file).

See Also: The SHOW MAI and SCRIPT command descriptions.

MAISESS BUILD

Function: Initiates the building of the MAI-FS session list.

```
MAISESS  BUILD
(0)
```

Use: The MAISESS BUILD command initiates the building of the MAI-FS session list, according to the user's MAI user attributes. The system VFS dataset is accessed for the MAI session list information. This command can be used where access to the MAI Menu is not possible or desired, for example, from non-terminal owning regions such as BSYS.

Notes: The &MAISADD verb can be used to build the MAI session list from sources other than VFS.

MAISESS CANCEL

Function: Terminates the nominated session.

```
MAISESS  CANCEL   [ ={ UNCOND | FORCE | COND } ]
                  { ID=session-id [ REGION={ luname | userid } ] |
                      RELNUM=nnnn }
```

Use: The nominated session is terminated with session termination codes reflecting the CANCEL operation selected. This command is equivalent to the .C, .CU, .CF, and .CC session commands. A script procedure, if active, is flushed and the session ended.

Operands:

ID=*session-id*

This operand supplies the identifier of the session to be cancelled.

REGION={ *luname* | *userid* }

The REGION operand identifies the session of another user region. The ID parameter must be used to identify the session. The REGION operand can be specified with either a user ID or region name (*luname*). The targeted session is terminated in error. The session end panel displays a message indicating that the session has been cancelled.

RELNUM=*nnnn*

This operand supplies a relative session number within the MAI-FS session list to identify the target session.

MAISESS DELETE

Function: Deletes the nominated session.

```
MAISESS      DELETE
             { ID=session-id | RELNUM=nnnn }
(0)
```

Use: The nominated inactive session is deleted from the MAI-FS session list. This command can be used to release resources for a session definition that is no longer required. An error message is issued if the session is still active.

Operands:

ID=*session-id*

This operand supplies the identifier of the session to be deleted.

RELNUM=*nnnn*

This operand supplies a relative session number within the MAI-FS session list to identify the target session.

MAISESS MODIFY

Function: Alters the attributes of the nominated session.

```
MAISESS     MODIFY
            { ID=session-id | RELNUM=nnnn }
            { [ COMPRESS={ YES | NO } ]
              [ FASTJUMP={ YES | NO } ]
              [ RESTART={ YES | NO } ]
              [ MENU1=[ key-name ] ]
              [ MENU2=[ key-name ] ]
              [ SWAP1=[ key-name ] ]
              [ SWAP2=[ key-name ] ]
              [ FWDJUMP1=[ key-name ] ]
              [ FWDJUMP2=[ key-name ] ]
              [ REVJUMP1=[ key-name ] ]
              [ REVJUMP2=[ key-name ] ] }
(0)
```

Use: The attributes of the nominated active session are altered as specified by the operands that follow the command.

Operands:

ID=*session-id*

This operand supplies the identifier of the session to be modified.

RELNUM=*nnnn*

This operand supplies a relative session number within the MAI-FS session list to identify the target session.

COMPRESS={ YES | NO }

This operand controls the use of compression for the session. COMPRESS=NO deactivates compression and COMPRESS=YES activates compression.

FASTJUMP={ YES | NO }

This operand controls the use of the fastjump option for the session. Specifying FASTJUMP=YES causes MAI to jump when a jump key is pressed, without saving user input.

RESTART={ YES | NO }

This operand controls the use of the restart option for the session. Setting RESTART=YES causes MAI to maintain the session environment when the session ends and to attempt to reconnect the session to the application.

MENU1=[*key-name*]
MENU2=[*key-name*]
SWAP1=[*key-name*]
SWAP2=[*key-name*]
FWDJUMP1=[*key-name*]
FWDJUMP2=[*key-name*]
REVJUMP1=[*key-name*]
REVJUMP2=[*key-name*]

These operands control the jump keys in use for the session. Values can be either F1 to F24, PA1, PA2, PA3, ATTN, or null. Setting a null value deactivates any currently specified key value for the jump function.

MAISESS START

Function: Activates the specified MAI-FS session definition.

```
MAISESS     START
            { ID=session-id | RELNUM=nnnn }
(0)
```

Use: This command activates the specified MAI-FS session definition. Session establishment is performed in the normal manner for MAI-FS operation. If a script is allocated to the session, it is invoked to manage the datastream flows.

Under terminal owning regions, the session operates the same as if started directly from the menu, except that the Logon Details panel is not displayed if the target application is unavailable. In this case the session end panel is displayed with appropriate messages.

Under non-terminal owning regions (ROF, CONSOLES, BSYS, AOMPROC and so on) the following applies:

- The session should be defined with a specific LOGMODE as no terminal default exists.
- Session output that is not handled by the script is automatically deleted, that is, it is treated as having been passed to a physical terminal.
- Session termination is as per a terminal owning region.

Operands:

ID=*session-id*
This operand supplies the identifier of the session to be started.

RELNUM=*nnnn*
This operand supplies a relative session number within the MAI-FS session list to identify the target session.

MAXUSERS

Function: Displays and limits the maximum number of concurrent users logged on to MS regions.

MAXUSERS [*number*] (3)

Use: The MAXUSERS command can be used to limit the maximum number of concurrent users of MS regions at any time. During a period of critical system load you might want to limit the number of users. If you enter this command without any operands, then the current number of users and the current limit are displayed.

Operands: ***number***

Sets a new limit to the number of concurrent users. Range 0 to 9999. The system default is 500. If *number* is omitted, then the display shows the current number of users and the limit.

Examples:

```
MAXUSERS 10
MAXUSERS
```

Notes: This limit applies only to the number of active users. It does not restrict the number of terminals that can be under the control of the EASINET facility.

MSG

Function: Sends a message to the specified OCS operator or to all OCS operators and dependent environments.

MSG (0)	{ *user ID* | *nodename* | *consoleID* | ALL | * } *text of message*

Use: It is possible to send messages to another user in OCS. The MSG command allows you to send a free form text message to a user ID operating in OCS mode, by specifying the user ID or the VTAM node name of the destination to receive the message. To send a message to all OCS Operators and dependent environments, specify the ALL operand.

Operands:

user ID
The user ID that is to receive the message.

nodename
The VTAM node name that is to receive the message.

consoleID
The system console to which the message is to be sent

ALL | *
Send message to all OCS operators and dependent environments profiled to receive unsolicited messages.

text of message
The text of the message to be sent.

Examples:

```
MSG USER1 IS THE SYSTEMS PROGRAMMER THERE?
MSG TERM1 CAN YOU REPORT FOR MEETING AT 5.00?
MSG ALL HAS ANYONE SEEN JIM BROWN?
```

Notes: User IDs and node names should be unique within the network. If a user ID is currently in session that has the same name as a connected VTAM node the user ID will receive the message.

A message can be sent to multiple users of terminals by specifying a generic name. In example 3 above, the user IDs ZHSTM, ZHSFR and so on, would receive the message dependent on whether they were using OCS.

A single user can receive the message in more than one environment if multiply connected. This would occur, for example, if a user had both terminal session and an ROF session when a message was sent.

In the OCS environment, the PROFILE MSG operand can be set to N, to specifically block the receipt of message command generated messages. Similarly, the UNSOL operand can block messages generated by this command as they are classified as unsolicited.

A message showing the number of receivers indicates the successful sending of the message. If a message cannot be delivered, an error message explaining the reason is returned.

If messages are to be sent regularly to a particular user ID or node you should consider allocating a PREFix Function key. If this is done you need only type the message text in the command line and press the appropriate Function key for the message to be sent. For example:

```
PF4 PREF,MSG USER1+
```

If the target user ID has multiple signons to the same system, then the MSG text will be sent only to the first available environment of each connection.

Messages can be sent to the operating system console by specifying the user ID allocated to it (use the SHOW USERS command to determine the user ID).

On z/OS, OS/390, and MSP systems, a specific console can be sent a message by specifying the terminal name, using the console's ID number. That is, the console with an ID of 02 can be sent a message by specifying CONS#02 as the destination. Consoles 01 to 99 can be sent a message in this way.

Note

The consoles must be defined to the operating system.

Receipt of MSG command messages can be controlled by the PROFILE of the target environments. An environment set with PROFILE MSG=N or PROF UNSOL=N will not receive such messages.

See Also:

The NOTIFY command description.
The PROFILE command description (MSG and UNSOL operands).

NCLCHECK

Function: Tests syntax for an NCL procedure without execution.

NCLCHECK { *procname* | PROC=*procname* } (0)

Use: NCLCHECK lets you test the syntax of an NCL procedure without executing it. The nominated procedure is loaded and verified as if it were to be executed. Any syntactical errors will be reported. The procedure is never executed, whether errors are found or not.

Operands: ***procname* | PROC=*procname* (Mandatory)**

procname is the name of the NCL procedure to be tested. The name must follow normal naming conventions and library requirements.

Examples:

```
NCLCHECK PROC1
NCLCHECK PPOPROC1
NCLCHECK PROC=MYPROC
```

Notes: The following are checked by the NCLCHECK command:

- Correct pairing of DO, DOWHILE, DOUNTIL with DOEND statements
- IF/THEN/ELSE structuring
- Invalid statements after ENDAFTER
- Statement concatenations exceeding system limit
- ELSE on same statement as IF/THEN
- Invalid syntax on IF, DOWHILE, DOUNTIL, DO, DOEND, ELSE, and ENDAFTER.

Note

Only the first error encountered is reported.

The EXEC, FSPROC and START commands are used to execute NCL procedures in the procedure library.

The LIST command is used to list an NCL procedure in the procedure library.

See Also: The EXEC, FSPROC, START, and LIST command descriptions.

Function: Sets, resets, or displays NCL test status for this window.

```
NCLTEST        [ ON | LOG | OFF ]
(0)
```

Use: NCL procedures can be shared among multiple users, via autoshare, retain, and preloading.

When testing NCL procedures, it is useful to be able to force load a fresh copy of procedures. The NCLTEST command allows you to indicate that all NCL procedures loaded for execution in this window are to be fresh, private copies.

In addition, you can force all NCL processes in the region to execute as if CONTROL INTLOG is always in effect. This allows you to see the results of all internally issued commands, even if the process specifies CONTROL NOINTLOG.

Operands: **[ON | LOG | OFF]**

Controls the status of NCL TEST mode for this window.

Omitted

Displays the current status of NCL TEST mode.

ON

Sets NCL TEST mode on (procedure reloading only).

LOG

Sets CONTROL INTLOG override on (force logging of all INTCMD-issued commands in the region).

OFF

Sets NCL TEST mode off (both reloading and INTCMD).

Examples:

```
NCLTEST ON
NCLTEST
```

Notes: A window starts with NCLTEST OFF in effect. Each window remembers the status independent of the other.

Skipping within a window does not alter the status. Swapping windows does not alter the status of each window.

Exiting a window and reopening it will reset the status to OFF.

You cannot reset just procedure reloading or INTLOG override. If you have both on and wish to reset just one, issue the following command:

`NCLTEST OFF;NCLTEST` *`xxx`* `(where` *`xxx`* `is ON or LOG)`

NCLTEST

See Also: PROFILE command (NCLTEST operand)—an alternative way of specifying the NCL TEST status.

NCLCHECK command

NCLTRACE

Function: Dynamically alters the trace status of an executing NCL process.

```
NCLTRACE      ID=nclid
            { ON | [ TERM | LOG ]
            [ STD | ALL | LABEL ]
            [ RESET ] |
              OFF }
(0/2)
```

Use: The NCLTRACE command allows you to dynamically start and stop NCL procedure tracing for an executing NCL process.

This allows you to debug executing processes without stopping them, editing the NCL to include trace options, and then restarting them.

Operands:

ID=*nclid*

The NCLID of the executing process. This can be obtained by use of the SHOW NCL command.

The NCL process must be executing in your own region unless you have the OPAUTH required by this command, in which case the target NCL process can be in any region.

ON

Indicates that NCL tracing is to be started for the nominated process

TERM | LOG

For ON, indicates whether output from trace messages is to be delivered to the terminal or the log. The default is TERM.

STD | ALL | LABEL

For ON, indicates the trace type. Correspond to the TRACE, TRACEALL, and TRACELAB operands of the &CONTROL verb. The default is STD.

RESET

For ON, this operand indicates that the process trace limit is to be reset. Normally, once *n* trace messages have been written, tracing for a process is disabled. NCLTRACE will not clear this limit unless RESET is specified. The value of *n* is set by the SYSPARMS NCLTRMAX command.

OFF

Indicates that NCL tracing is to be terminated for the nominated process

Examples:

```
NCLTRACE ON ID=143 LOG ALL RESET
NCLTRACE OFF ID=143
```

NCLTRACE

Notes: NCLTRACE is especially useful for debugging system procedures such as PPOPROC, AOMPROC, and so on.

If LOG is specified, trace messages are only sent to the log. In this case, the ALL option is useful to provide NCLID and procedure information on the messages.

See Also: The SHOW NCL command description.

NDB ALTER

Function: Allows (re)building or validating of the key indexes either for an entire NDB, or for a single field.

```
NDB  ALTER     ndbname
               OPT={ BLDX | CHKX }
               WORK=fileid
             { DB | FIELD=fieldname
             [ UNIQUE={ NO | YES } ] }
             [ SORT=sortsize ]
             [ DETAIL={ NO | YES } ]
             [ PAD=percent ]
             [ UNLOCK ]
             [ LOCK ]
(2)
```

Use: This command can be used to build, rebuild, or validate the index records in an NDB. If the NDB is in LOADMODE, NDB ALTER *ndbname* OPT=BLDX is the only way to reset LOADMODE.

Operands: ***ndbname***

The mandatory 1- to 8-character name of the database to be altered. This must be the same name as the ID specified in a previous UDBCTL OPEN command (or DDNAME, if ID=* was specified).

OPT={ BLDX | CHKX }

Indicates whether to build indexes (BLDX) or check indexes (CHKX).

The database can be in load mode when BLDX is specified. It must not be in loadmode for CHKX or for BLDX on a single field.

WORK=*fileid*

Specifies the file ID of a VSAM work file. The work file must be current open (UDBCTL OPEN=...), and the file must be completely empty (use RESET on open). The file must not be in use by any other MS facilities.

The following VSAM attributes are required for this file:

- KSDS. Key length = 4, Key offset = 0.
- Min CISZ is 1024. Best CISZ is approx. 1/2 track.
- Max RECSZ must be the specified CISZ - 7.

The work file is used to write sorted key strings, if the sort space is insufficient to hold all keys in storage. To allow for merging, at least twice the total key storage space is required. The VSAM records on the work file contain blocks of keys, each consisting of a 10-byte prefix and the key itself. The keys are *n* bytes for character keys, 4 for numeric, 3 for date, *n* (binary) for hex, 3 for cdate, 5 for time, 8 for float and 8 for timestamp. The entire record has the 4-byte VSAM key at the front.

The recommended values are 22K CISZ on 3380 and 4.5K on 3350 (if DFP which allows granular CI sizes is installed).

The UDBCTL OPEN should specify STRNO=3, BUFNI=5, and BUFND=6.

DB | FIELD=*fieldname* [UNIQUE={ NO | YES }]

Specifies that NDB indexes are to be built or checked.

DB indicates that the entire database is to have its indexes built or checked.

FIELD nominates a single field that is to have indexes built (the field must not be currently keyed) or checked (the field must be keyed). In either case, the database must not be in load mode.

UNIQUE allows specification of whether or not a field that is having indexes built is to be treated as unique keyed (YES) or not (NO). The default is NO. This option is not allowed for CHKX.

SORT=*sortsize*

Specifies the amount of storage (in Kilobytes) that is to be used for sorting keys. Obviously, the more sort storage (within paging and region size considerations) the faster the ALTER will be. The minimum size allowed is 5(K). The default and maximum size depends on the operating system, as follows:

OS	Default(K)	Max (K)
XA, ESA:	1000	100000
z/OS, OS/390, MSP, VOS3:	400	8000
VM:	320	8000
VSE:	160	8000

If the sort space is insufficient, sorted strings are written to the work file and then merged into a single sorted string. A larger sort space will mean fewer strings and thus less I/O to the work file. If all keys can be held in storage, no work file I/O will be done at all.

DETAIL={ NO | YES }

Controls whether additional progress messages will be written, for example, sort/merge activity and record interval reports. This is useful for monitoring the progress of the ALTER.

PAD=*percent*

Indicates the percentage of the new key records that is to be left free for addition of record IDs later. The percentage must be from 0 to 90 (the default is 25). This percentage is of the usable data portion of the VSAM record.

Note

This option is not allowed for CHKX.

UNLOCK

Allows a locked database to be unlocked to run the ALTER. While the ALTER is running, it is a *long-running-command* and no other commands will be accepted (except NDB STOP IMM).

LOCK

Causes the database to be locked after the ALTER. Note that if a CHKX operation detects errors, the database is unconditionally locked afterwards.

Examples:

```
NDB ALTER NDB1 OPT=BLDX DB WORK=MYWORK SORT=30000
```

Notes:

While an ALTER is in progress, no other NDB commands are allowed (except STOP IMM, see below). Also, the database must not be active before an ALTER is initiated. It can be LOCKED, and the ALTER UNLOCK will unlock it.

An ALTER can be immediately stopped by using the NDB STOP db IMM command. This is the only command that is accepted while the alter is in progress.

If an ALTER encounters a unique key violation, an error message is written but the keys are loaded. The message information (or a key histogram pass) can be used to identify and correct the records which are in error.

If an ALTER encounters a field (char or HEX) that is too long to key within the VSAM keylength, it is reported and scanning continues. Only the first 10 violations for any field name are reported. After all data has been scanned, a total of errors is written which includes unreported errors. If any errors are detected, the ALTER terminates.

If building keys for the entire database, ALTER will force the NDB into LOADMODE if it is not already in load mode just prior to deleting existing key records. Prior to this point, no harm is done if the ALTER is terminated. After setting load mode, an ALTER is the only way of resetting load mode.

If building keys for a field (that is, dynamically altering to KEY=Y/U) then the database is not placed into load mode. If any failure occurs, the field is not marked keyed and any inserted key records are ignored by $NDB.

When checking indexes, if errors are detected a message indicates the error.

There are three errors possible:

- No key for field in record (record present)
- No record for key field
- Mismatch between key and record values for a field

These are all reported with information about field name, value, and RID.

NDB ALTER will automatically build key statistics when building indexes (BLDX). Therefore, you do not have to issue an NDB START KEYSTATS afterwards.

NDB CREATE

Function: Initializes a new $NDB database.

```
NDB  CREATE        ndbname
                  [ LANG=lc ]
                  [ LOADMODE ]
                  [ LOGSIZE=size ]
                  [ UNLOCK ]
                  [ LOCK ]
(2)
```

Use: This command is used to initialize an empty $NDB database. The database is formatted as an NDB database.

Operands: ***ndbname***

The mandatory 1- to 8-character name of the database to be created . This must be the same name as the ID specified in a previous UDBCTL OPEN command (or DDNAME, if ID=* was specified). The nominated UDB must have the following attributes:

- It must be a VSAM KSDS
- It must be completely empty, that is, just defined or opened with the RESET option on the UDBCTL command
- The keylength specified should be the maximum of 16, that is, 8 plus the longest character field to be keyed, and 9 plus half the length of the longest hex field to be keyed (counting displayable characters). The maximum possible is 255 (this is a VSAM restriction). The minimum of 16 guarantees that numeric and date format data can always be keyed.
- The minimum value of the VSAM maximum data length must be 274 plus twice the keylength
- It must not have SPANNED specified, as $NDB handles records longer than the VSAM record length automatically

LANG=*lc*

Specifies a system defined language code which flags the database as language specific. The uppercase translation table for that language will be used for all uppercase processing.

See the appendix titled *Supported Language Codes for National Language Support* in the *Network Control Language Reference* for a list of system defined language codes.

LOADMODE

Causes the newly created NDB to be in load mode. This means that no keys are built or managed when records are added or updated. An NDB ALTER can later be used to build the keys and reset load mode. This allows for fast loading.

LOGSIZE=*size*

Allows a specific number of journal blocks to be formatted. If not specified, the SYSPARMS NDBLOGSZ operand value is used. *size* must be in the range 10 to 200.

Note

The journal can dynamically extend if it needs more blocks than are originally formatted.

UNLOCK

Allows a locked database to be unlocked to run the CREATE.

LOCK

Causes the database to be locked after the CREATE.

Examples:

```
NDB CREATE MYNDB LANG=GR
NDB CREATE DB2  LOADMODE LOCK LOGSIZE=100
```

Notes:

The NDB CREATE command should only be used once for a database. The command will not execute against a non-empty dataset—this protects against inadvertent loss of an existing database.

See Also:

The NDB RESET command.

NDB FIELD

Function: Allows a database field definition to be added, deleted, or updated without needing to use the &NDBDEF NCL verb.

```
NDB  FIELD      ndbname
                { ADD=fieldname | DELETE=fieldname |
                     UPDATE=fieldname }

                [ other operands ]
(2)
```

Use: The NDB FIELD command can be used to add, delete and update field definitions in an NDB. If the &NDBDEF verb is disabled via NCLEX01, this can prevent accidental field definition deletion, and so on.

Operands: ***ndbname***

This operand is the required 1- to 8-character name of the database field definition to be added, deleted, or updated. The NDB must be started.

ADD=*fieldname* | DELETE=*fieldname* | UPDATE=*fieldname*

Indicates whether a field is being added, deleted, or updated. When adding, operands can follow. Unspecified operands get a default value. If deleting, no other operands are allowed. If updating, operands can follow. Unspecified operands are not altered.

Allowable operands:

```
FMT={CHAR | NUM | HEX | DATE | FLOAT | CDATE | TIME | TIMESTAMP}
KEY={ NO | YES | UNIQUE | SEQUENCE }
UPDATE={ YES | NO }
CAPS={ YES | NO | SEARCH }
NULLFIELD={ YES | NO }
NULLVALUE={ YES | NO }
DESC='description'
USER1='userdata' USER2='userdata'
USER3='userdata' USER4='userdata'
NEWNAME=newname
BASE={ NONE | DECIMAL | HEX }
```

FORMAT={ CHAR | NUM | HEX | DATE | FLOAT | CDATE | TIME | TIMESTAMP }

The field format. For ADD, the default is CHAR. For UPDATE, the default is the current field format. Normally the format cannot be changed.

KEY={ NO | YES | UNIQUE | SEQUENCE }

The field key option. For ADD, the default is NO. For UPDATE, the default is the current field key option. See the *Network Control Language User's Guide* for update rules.

UPDATE={ YES | NO }

Indicates whether or not the field value can be updated.

CAPS={ YES | NO | SEARCH }
For character fields, indicates whether the field will be uppercased or not. CAPS=SEARCH will result in the data being stored as is, but searches involving the field will treat the data as uppercase.

NULLFIELD={ YES | NO }
Indicates whether or not the field can be omitted from database records.

NULLVALUE={ YES | NO }
Indicates whether or not a null value can be provided for the field (0 or blank).

DESCRIPTION=*'description'*
Up to 60 characters of free-format field description.

USER*n*=*'userdata'*
Four user fields; each can contain up to 8 characters of information.

NEWNAME=*name*
For UPDATE only, the field can be renamed by using this operand.

BASE={ NONE | DECIMAL | HEX }
Specifications of BASE=NONE or BASE=DECIMAL for a numeric field results in an external representation as a signed decimal number. BASE=HEX causes the external representation to be a hexadecimal string, with leading zeros suppressed on output.

Examples:

```
NDB FIELD NDB1 ADD=MYFIELD FORMAT=CHAR KEY=YES
NDB FIELD NDB1 DELETE=FIELD23
```

Notes:

The advantage of the NDB FIELD command is that it is &INTCMDable and thus the response messages can be easily accessed from NCL. Also, it is subject to command authority checking. The &NDBDEF verb is not. &NDBDEF can be disabled via NCLEX01, thereby making the NDB FIELD command the only way to modify field definitions. See the notes on load mode and empty database for restrictions on field attribute alterations.

The NDB FIELD command supersedes the &NDBDEF NCL verb. See the *Network Control Language Reference* for full details.

Rules regarding what can be changed when updating are documented in the *Network Control Language User's Guide*. Load mode and NDB empty are relevant in some cases. It is never an error to specify an operand with the same value as it currently has on update.

NDB PURGE

Function: Frees a locked or halted database.

```
NDB PURGE   dbname
(2)
```

Use: This command is used to unlock or reset the halted status of an NDB that has either been locked via an NDB STOP LOCK command or halted due to I/O errors, and so on, without actually starting the database. It will be possible to restart the database without specifying the UNLOCK option on the NDB START command.

Operands: ***dbname***

This is the mandatory 1 to 8 character name of the database whose status is to be purged. Must be the ID of an NDB that is marked as 'LOCKED' or 'BAD-LOCKED' in a SHOW NDB=ALL display.

Examples:

```
NDB PURGE MYNDB
```

Notes: The NDB PURGE command is useful after trying to START a database that has not been CREATEd. The START failure will have flagged the database as 'BAD-LOCKED', and an NDB CREATE will not be allowed. Use NDB PURGE first, then NDB CREATE.

See Also: The NDB START and NDB STOP commands.

NDB RESET

Function: Deletes all data from an $NDB database while preserving all field definitions.

```
NDB  RESET     ndbname
               [ LOADMODE ]
               [ UNLOCK ]
               [ LOCK ]
(2)
```

Use: This command is used to delete all data from a $NDB database while preserving the field definition information. As such, it is useful to clear an accumulation database, while preventing the need to rebuild the field definitions, or the need to CREATE it again.

Operands:

dbname
This is the mandatory 1- to 8-character name of the database to be reset. The database must not be currently active.

LOADMODE
Causes the NDB to be placed in load mode after the reset. Thus, it is as if the NDB was created with LOADMODE specified. See CREATE.

UNLOCK
Allows a locked database to be unlocked to run the RESET.

LOCK
Causes the database to be locked after the RESET.

Examples:

```
NDB RESET MYNDB
```

Remove all data from MYNDB, but preserve all field definitions.

Notes: This command should be used with care. Once an NDB is RESET, there is no way to recover the data. Take special care with NDBs being used with INFO/MASTER, as the database holds INFO/MASTER control information as well as data, so resetting it will stop INFO/MASTER from working.

See Also: The NDB CREATE command.

NDB START

Function: Starts a $NDB database.

```
NDB  START     ndbname
               [ { OPTIMIZE | OPTIMISE }={ YES | NO } ]
(2)            [ KEYSTATS [ ={ START | STOP } ] ]
               [ DEFER | NODEFER ]
               [ SCANDEBUG | NOSCANDEBUG ]
               [ TRACE | NOTRACE ]
               [ LOADMODE ]
               [ UNLOCK ]
               [ INPUT ]
               [ FORCE ]
               [ JOURNAL | NOJOURNAL]
```

Use: This command is used to start a $NDB database, to change the deferred I/O status of an NDB or to start and stop global tracing of NDB activity. An NDB must be started using the START command before it can be accessed by NCL processes (using &NDB*xxx* NCL verbs).

Operands: ***ndbname***

This operand is the required 1 to 8 character name of the database to be started. The database must have been previously created, and a UDBCTL OPEN command executed for it. If an NDB STOP command had been previously executed, but the database had not yet stopped, the STOP status is cancelled.

{ OPTIMIZE | OPTIMISE }={ YES | NO }

This operand allows you to enable (set to YES) or disable (set to NO) the scan optimizer for this NDB. When specified, it overrides the setting of the SYSPARMS NDBSCANO operand.

KEYSTATS [={ START | STOP }]

This operand allows you to start or stop keyed field statistics collection, that is, it is used by the scan optimizer.

Note

This request fails if the INPUT operand is specified.

DEFER | NODEFER

DEFER is an optional keyword, indicating that the NDB is to run in DEFER I/O mode. Normally, every update statement issued against an NDB has a forced buffer flush to ensure integrity. This, however, means that update statements are slow, particularly when bulk adding data. Provided that the database was UDBCTL opened using the DEFER and LSR options, using the DEFER option on the start command instructs the database not to flush buffers. If the database is already started, DEFER option causes an immediate setting of DEFER mode.

DEFER is mutually exclusive with the JOURNAL option.

Warning

A database started in DEFER mode has no transaction integrity. If a system or Management Services failure occurs, the database will be flagged as unusable. You should only use DEFER mode to bulk load data when you can completely rebuild the database as it was (for example, after a full backup).

NODEFER causes the NDB to run in normal, buffer flush mode. This is the default if the NDB being started is not already active. If the NDB is currently active in DEFER mode, this keyword will cause an immediate switch to normal, non-deferred I/O mode, thus reestablishing transaction integrity.

SCANDEBUG | NOSCANDEBUG

SCANDEBUG controls whether &NDBSCAN statements issued by any NCL procedure within this system will generate debugging messages showing the parsed scan-request, and messages showing the record counts passing each level of the scan to the activity log. This operand operates independently of any &NDBCTL SCANDEBUG requests.

NOSCANDEBUG causes no debugging information to be logged. This is the default.

TRACE | NOTRACE

TRACE causes message N87Y01 to be written to the activity log every time an NDB statement or an &NDB*xxx* verb referencing this NDB is executed by an NCL procedure anywhere in the system. The message includes the call type, NDB ID, record ID and user/NCL procedure ID (see the online help for N87Y01 for more information about the content of the message).

NOTRACE stops the trace messages described above being written to the activity log. NOTRACE is the default.

LOADMODE

Causes the database to be placed into load mode. Once in load mode, no keys are maintained and only an ALTER BLDX will reset load mode.

UNLOCK

Allows starting and unlocking a database that has been locked by a prior NDB STOP LOCK command or I/O error, and so on. If this operand is omitted, and the database is locked, it will not be started.

INPUT

Starts the database in READ-ONLY mode. No updating is done. It is mutually exclusive with DEFER, NODEFER, LOADMODE, FORCE, and JOURNAL and not effective if the database is already started.

No integrity checks are performed on input-only databases.

FORCE

Force opens the NDB even if the control record contains a non-blank domain ID that is different from the current domain ID. Doing this could damage the database if it is simultaneously open on another system.

This operand allows you to open an NDB on another system, after a previous system has abnormally terminated.

JOURNAL | NOJOURNAL

Specifies whether NDB journal records are to be written to the Management Services journal dataset. The default for this operand depends on the option that was in effect the last time the NDB was started. Once the JOURNAL operand has been specified for an NDB, journaling can be terminated only with the NOJOURNAL operand.

Examples:

```
NDB START MYNDB         -* Starts an NDB called MYNDB
NDB START MYNDB DEFER   -* Starts MYNDB and sets DEFER I/O mode
NDB START MYNDB UNLOCK  -* Starts MYNDB, even if it is
                        -* currently locked
```

Notes:

Leaving both the DEFER and NODEFER keywords off an NDB START command causes the database to start in NODEFER mode; otherwise the current mode is not changed.

A database activated by a user signing on always starts in NODEFER mode.

Key statistics should be initiated from time to time (for example, weekly on Sunday night), depending on the volatility of the NDB.

NDB ALTER also builds key statistics.

See Also:

The NDB STOP command description.

NDB STOP

Function: Shuts down a $NDB database.

```
NDB  STOP      ndbname
               [ IMMEDIATE ]
(2)            [ LOCK ]
```

Use: This command is used to shut down a $NDB database. By default, the database stops when all currently signed on users sign off. Options allow an immediate shutdown, rather than waiting for all active users to sign off, and/or to lock a database from all access. In all cases, no new users will be permitted to sign on.

Operands:

ndbname

Is the required 1- to 8-character name of the database to be stopped. This name need not be the name of an active, or even created database (although only the STOP LOCK option would be useful in that case). If the database is not active, then an error message is issued. The nominated database is placed in the STOPPING state.

IMMEDIATE

An optional keyword, indicating that the database is to be immediately stopped, that is, all active users are given an error response on the next call. If this is omitted, then the database stops when all active users sign off. This operand can be abbreviated to IMM.

LOCK

An optional keyword, indicating that the database is to be locked from further access once stopped. This LOCKED status can be cleared only by an NDB START UNLOCK command, by an NDB PURGE command, or by Management Services being shut down and restarted. Note that an NDB STOP LOCK command can be executed against a nonexistent database.

Examples:

```
NDB STOP MYNDB
```

Flag MYNDB as stopping. When all users sign off, the database stops.

```
NDB STOP MYNDB IMM LOCK
```

Immediately stop MYNDB. When stopped, lock the database against further access until an NDB START MYNDB UNLOCK command is executed.

NDB STOP

Notes: NDB STOP provides an orderly way to shut down a database.

NDB STOP IMM provides a quick way to shut down a database. NDB STOP LOCK allows a database to be locked (for example, when backups are taking place).

A database that is in STOPPING status can be restarted by an NDB START command. The STOPPING status is removed.

The NDB STOP IMMEDIATE command can be used to halt an NDB UNLOAD or ALTER which is currently in progress. If key statistics are being collected (using NDB START KEYSTATS), the collection is terminated. The next NDB START restarts collection where it ended, unless NDB START KEYSTATS=NO is specified.

See Also: The NDB START and NDB PURGE commands.

NDB UNLOAD

Function: Copies a database and field definition to a VSAM ESDS.

```
NDB  UNLOAD        ndbname
                   TO=esdsid
                 [ ORDER={ PHYSICAL | RID } |
                                FIELD=fieldname ]
                 [ ERROR={ CONTINUE | STOP } ]
                 [ LOCK ]
                 [ UNLOCK ]
                 [ REPORT={ NO | number } ]
                 [ LIMIT={ NO | 0 | nrecs } ]
                 [ DATECONV ]
(2)
```

Use: This command is used to unload an NDB database definition, field definitions and data to a nominated VSAM ESDS, where it can be accessed by the NCL &FILE*xxx* verbs. This command can also be used to copy a database between two systems.

Operands:

ndbname

This is the mandatory 1 to 8 character name of the database to be unloaded. The database must have been previously CREATEd, and a UDBCTL OPEN command executed for it.

TO=*esdsid*

Is the required 1 to 7 character name of the ESDS to unload the NDB to. The ESDS must be UDB opened and reset. No other users of the ESDS are permitted. The minimum record length for the ESDS is 300 bytes.

ORDER=PHYSICAL | RID

An optional keyword, indicating the order in which records are to be unloaded. ORDER=PHYSICAL (the default) unloads the NDB in the order that the records appear on disk. ORDER=RID unloads the NDB in ascending record number.

FIELD=*fieldname*

An optional keyword, mutually exclusive with the ORDER keyword, indicating that the records are to be unloaded in ascending sequence, based on the (keyed) fieldname.

Note

Only records containing the nominated keyed field are unloaded.

ERROR=CONTINUE | STOP

An optional keyword, indicating the course of action to perform if an error is encountered in reading an NDB record. ERROR=CONTINUE (the default) indicates that the unload is to continue, but a message is to be displayed whenever a record in error is encountered. ERROR=STOP indicates that the unload is to stop at the first error.

LOCK
An optional keyword indicating that the NDB is to be locked when the unload is complete.

UNLOCK
An optional keyword, indicating that the unload is to continue regardless of whether the NDB is locked or not. If not specified then locked NDBs cannot be unloaded. This is not mutually exclusive with the LOCK option.

REPORT=NO | *number*
An optional keyword, indicating how often (if ever) to display messages reporting on the status of the unload. NO (the default) indicates that no messages are to be produced, and *number* indicates that messages are to be produced every number records. The *number* must be greater than or equal to 100.

LIMIT={ NO | 0 | *nrecs* }
An optional keyword, indicating the maximum number of records to unload from the NDB. If a number is specified, the unload of the data is limited to the number of NDB records indicated. Specifying LIMIT=0 will unload only the field definitions from the NDB. Not specifying any limit will unload the entire NDB. Field definitions are always unloaded.

DATECONV
An optional keyword, indicating that the records are to be changed from DATE format to CDATE format during the unload.

Examples:

```
NDB UNLOAD MYNDB TO=THISUDB
```

Unload MYNDB into the ESDS identified by THISUDB.

```
NDB UNLOAD MYNDB TO=THISUDB FIELD=NAME UNLOCK LIMIT=1000
```

Unload MYNDB into the ESDS THISUDB, ordering the records by the NAME field. Unlock the database to perform the unload and unload only the first 1,000 records.

Notes:

NDB UNLOAD provides a means to transport NDBs and their definitions or to backup NDBs.

NDBs to be unloaded must not be started.

To reload an NDB dumped from an unload command use the supplied NCL procedure $NDBRELD. The documentation is provided by the procedure.

The unload can be terminated by the NDB STOP IMMEDIATE command.

While the unload is in progress, the NDB is effectively locked. Any NDB commands (except STOP IMMEDIATE) are rejected with an RSP 251—long-running NDB command in progress.

See Also:

The section on unloading of NDBs in the *Network Control Language User's Guide*.

NETM

Function: Passes a command to NETM for execution.

NETM (2)	*xxxxxxxxxxxx*

Use: In a Hitachi VOS3 environment, commands can be routed to NETM for execution. This is done using the NETM command.

Operands: ***xxxxxxxxxxxx***

Text as it is to be passed to NETM.

Examples:

```
NETM #COM ALL,'THIS IS A TEST MESSAGE'
NETM #COMMAND $D A
```

Notes: For the NETM command to be used, a NETM/OP Connection Path must be available, defined previously by a NETMCNTL ADD= command.

Responses to commands passed to NETM will be received at the terminal or NCL procedure from which the command was entered.

The NETM command applies only to systems operating in a VOS3 operating system environment, but is valid when a SOLVE system is using ECS/VTAM or XNF communications access methods.

NETM passes commands to various target environments, including VOS3, JSS3, VTAM, and XNF.

See Also: NETMCNTL command description.

NETMCNTL

Function: Controls communication between SOLVE and NETM systems.

```
NETMCNTL     ADD=xxxxxxxx
                [ TYPE={ CMD | UNSOL } ]
                [ CAT=xx ]

NETMCNTL     DELETE=xxxxxxxx

NETMCNTL     START= { CMD | UNSOL | ALL }

NETMCNTL     STOP = {CMD | UNSOL | ALL }
(2)
```

Use: The NETMCNTL command is the means by which communication between SOLVE and NETM systems can be controlled by the operator. The ADD and DELETE operands are used to add or delete NETM/OP *Connection Path* definitions. Each Connection Path definition must include a NETM UAP name and UAP category that the SOLVE system will use to communicate with NETM/OP.

The two types of connection path are as follows:

- TYPE=CMD—used for command/response flows
- TYPE=UNSOL—used to receive unsolicited messages from NETM

At least one TYPE=CMD connection path must be added before the SOLVE system can send commands to NETM using the NETM command. A TYPE=UNSOL path must be added before unsolicited messages can flow from NETM to the SOLVE system.

The START and STOP operands are used to control the flow of command traffic and unsolicited message traffic between NETM and the SOLVE system.

Operands:

ADD=*xxxxxxxx*

This operand adds a NETM/OP Connection Path. *xxxxxxxx* specifies the name of a UAP to be used for connection to NETM/OP.

TYPE={ CMD | UNSOL }

This operand defines the type of messages (command/response or unsolicited) that will flow across this connection path.

CAT=*xx*

This operand defines the NETM UAP category that will be used with this connection path. The default is CAT=F1 for CMD paths, and CAT=CT for the UNSOL path.

DELETE=*xxxxxxxx*

This operand deletes the entry *xxxxxxxx* from the SOLVE list of connection paths defined for NETM/OP communication. Deletion will be scheduled immediately, but might not be completed for a few seconds if the path is busy servicing a command at the time the delete request is entered.

START={ CMD | UNSOL | ALL }

START=CMD enables command/response flow between SOLVE and NETM systems. By default, command/response flow is enabled. However, it can be disabled using a NETMCNTL STOP=CMD command.

START=UNSOL starts the flow of unsolicited messages from NETM. Prior to issuing a START=UNSOL command, a TYPE=UNSOL connection path must be defined using a NETMCNTL ADD= command.

STOP={ CMD | UNSOL | ALL }

STOP=CMD disables command/response flows between SOLVE and NETM systems. STOP=UNSOL stops the flow of unsolicited messages from NETM.

Examples:

```
NETMCNTL ADD=NETMSOLV1
NETMCNTL ADD=NETMSOLV2 TYPE=UNSOL CAT=FF
NETMCNTL START=UNSOL
```

Notes:

This command only applies to systems executing in a VOS3 operating system environment.

The SHOW NETMCNTL displays the status of SOLVE connections to NETM/OP.

When communicating with a NETM/OP version 07-02 system, a single TYPE=CMD connection path should be defined.

When communicating with a NETM/OP version 07-01 or earlier system, if multiple SOLVE users are to send commands to NETM/OP, it will be necessary to define several TYPE=CMD connection paths to allow the passage of concurrent commands. If insufficient paths are defined, commands will be queued for service. A message will be issued indicating that the command has been queued. In this case, consider increasing the number of connection paths to allow more commands to be executed concurrently.

Only one TYPE=UNSOL connection path can be defined.

The UAP names specified in the NETMCNTL ADD commands must be defined as UAPs to NETM/OP, otherwise NETM/OP will reject the attempts of the SOLVE system to communicate with NETM/OP.

See Also:

The NETM and SHOW NETMCNTL command descriptions.

NETSTAT (IBM)

Function: Invokes the IBM TCP/IP NETSTAT command—available in NetMaster for TCP/IP only.

```
NETSTAT    { ? | netstat_operands }
           [ ADDRESS=ip_address_mask ]
           [ LU=lu_name_mask ]
           [ APPL=application_name_mask ]
           [ STACK=stackname ]
```

Use: The NETSTAT command is used to display the current status of IBM TCP/IP resources.

Operands:

?

Displays help in the form of the NETSTAT command syntax. For a list of the NETSTAT command-dependent operands and an explanation of each, see the IBM TCP/IP documentation.

netstat_operands

The NETSTAT command parameters as used in the TSO NETSTAT command.

> **Note**
> The following TSO options are not supported: REPORT, STACK, and INTERVAL.

ADDRESS=*ip_address_mask*

The IP address of the remote host you are seeking information about. This address can be specified as a mask where * matches zero or more characters and ? matches exactly one character. IP Addresses are in the form *nnn.nnn.nnn.nnn* where *nnn* is a number in the range 0 to 255.

This operand applies only to the output of the NETSTAT TELNET option.

LU=*lu_name_mask*

The LU name of the IBM TCP/IP LU being used for a TN3270 connection. This can be specified as a mask, where * matches zero or more characters and ? matches exactly one character.

This operand applies only to the output of the NETSTAT TELNET option.

APPL=*application_name_mask*

The name of the SNA application associated with the TN3270 connection. This can be specified as a mask, where * matches zero or more characters and ? matches exactly one character.

This operand applies only to the output of the NETSTAT TELNET option.

STACK=*stackname*

The name of the stack you are seeking information about. The default is the stack on which you are operating.

This operand applies only if connection awareness has been enabled.

Examples:

```
NETSTAT TELNET
NETSTAT TELNET ADDRESS=192.168.2.*
NETSTAT TELNET ADDRESS=192.168.2.* APPL=CICS
NETSTAT GATE
```

Notes:

Some of the NETSTAT command options produce a large amount of output which can be reviewed in the SOLVE log (displayed with the $LOG command).

Filtering performed by the ADDRESS, LU, and APPL operands is in addition to any already specified by the native NETSTAT command options.

NETSTAT (TCPaccess)

Function: Invokes the TCP/IP for TCPaccess NETSTAT command—available in NetMaster for TCP/IP only.

```
NETSTAT      { ? | netstat_operands }
             [ STACK=stackname ]
```

Use: The NETSTAT command is used to display the current status of TCPaccess resources.

Operands:

?

Displays help in the form of the NETSTAT command syntax. For a list of the NETSTAT command-dependent operands and an explanation of each, see the *TCPaccess System Management Guide*.

netstat_operands

The NETSTAT command parameters as used in the TSO NETSTAT command.

STACK=*stackname*

The name of the stack you are seeking information about. The default is the stack on which you are operating.

This operand applies only if connection awareness has been enabled.

Examples:

```
NETSTAT TELNET
```

NOTIFY

Function: Sends a message to TSO/TSS users (z/OS, OS/390, MSP, or VOS3) or CMS users (VM/SP).

```
NOTIFY        { 'message-text',USER=(tso-userid) [ options ] |
                  CMS-userid 'message-text' }
(0)
```

Use: This command applies only in z/OS, OS/390, MSP, VOS3, or VM/SP systems and differs depending upon the operating system.

In z/OS, OS/390, MSP, and VOS3 systems the NOTIFY command is entered in exactly the same format as the system console SEND command. It results in the message text being sent to the nominated TSO/TSS user(s).

In VM/SP systems the NOTIFY command sends the message text to a nominated CMS user.

Operands:

message-text
The text for the message to be sent to the TSO or CMS user ID.

USER=(*tso-userid*)
The user ID to whom the text is to be sent.

options
As for the IBM SEND command (see the IBM manual *OS/390 System Command Reference*).

CMS-*userid*
The ID of the CMS user.

Examples:

```
NOTIFY 'Please call Support Center',USER=(TSXAC)
NOTIFY 'Please call Support Center',USER=(TSXAC),LOGON
NOTIFY CMXAC 'Please call Support Center'
```

See Also: The MSG command and the IBM *OS/390 System Command Reference*.

NOTRACE

Function: Terminates a VTAM network trace.

NOTRACE { *name* | *operand,operand,...operand* } ,TYPE=*type* (1)

Use: This command will build and issue a VTAM MODIFY command with the format:

```
F NET,NOTRACE,ID=name,TYPE=type or

F NET,NOTRACE,operand,operand,TYPE=VTAM
```

This command can be used to terminate a VTAM network trace.

Operands:

name
The VTAM name for the resource to have the trace terminated.

type
The type of VTAM trace to be terminated.

operands
Operands specified when terminating a TYPE=VTAM trace.

Examples:

```
NOTRACE LU01,TYPE=BUF
NOTRACE LINE5,TYPE=LINE
NOTRACE OPTION=(PIU,API),TYPE=VTAM
```

Notes: Alternatives to this command are:

```
F NET,ID=LU01,NOTRACE,TYPE=BUF
F NET,ID=LINE5,NOTRACE,TYPE=LINE
F NET,NOTRACE,TYPE=VTAM,OPTION=(PIU,API)
```

See Also: The F and TRACE commands.

NPTAB

Function: Changes the status of a user's RESOURCE LIST table entry or adds, replaces, or deletes a resource table.

```
NPTAB        [ resource-group ]
             { ACT=qualifier.tablename | INACT=qualifier.tablename |
               ADD=tablename | REP=tablename | DEL=tablename }
(0/4)
```

Use: Resource tables can be defined for a user in the RESOURCE LIST member name. Access to resources within a resource group can be granted in the specified resource table members. Inactive tables are bypassed when determining if an operator has access to a resource group. This can be useful in controlling which VTAM messages operators receive. By modifying the status of the $NMMSG resource group, operators have control over overlapping zones of responsibility in different time zones or geographical boundaries.

The NPTAB command can also be used to add new resource tables and replace or delete existing tables.

Operands: ***resource-group***

This is the resource-group name to which all subsequent operands are to apply. If this operand is not specified, the default of $NMMSG applies.

ACT=*qualifier.tablename*

The resource table specified is to be given ACTIVE status for this operator. Resources specified within this table will be made available to the operator.

INACT=*qualified.tablename*

The resource table specified is to be given INACTIVE status for this operator. Resources specified in this table will not be available to this operator.

ADD=*tablename*, REP=*tablename*, DEL=*tablename*

These facilities are reserved for the systems programmer and require an authority level of 4. They are fully documented in the *Management Services Administrator Guide*.

Examples:

```
NPTAB ACT=ZONE3
NPTAB INACT=NET5.SANFRAN ACT=NET3.SEATTLE
NPTAB USERDB INACT=SOCKS
```

Notes: Multiple operands can be specified on the one NPTAB command. The SHOW NPF command can be used to display the status of the resource tables for a user. The SHOW NPTAB command can be used to display contents of a particular resource table.

See Also: The SHOW NPF and SHOW NPTAB command descriptions.

NRDRET

Function: Restores all Non-Roll Delete messages for an OCS window.

```
NRDRET
(0)
```

Use: If you have performed cursor removal of non-roll delete messages from your OCS window, NRDRET will restore the hidden messages in your non-roll delete display area. Also, on entry to OCS mode, NRDRET will refresh your non-roll delete display area with all NRD messages that are currently in the system and that you are profiled to receive. This allows you to establish the current status of all non-roll delete messages in the system whenever OCS mode is entered.

NRDRET can also be issued from NCL procedures and from procedures running in a dependent processing environment.

See Also: The description of non-roll delete message handling in the chapter about OCS in the *Management Services User's Guide*.

Function: Creates and sends a general or specific broadcast from an OCS mode panel.

```
NSBRO        { GB1=text | GB2=text | GB3=text | GB4=text |
               SB1=text | SB2=text | SB3=text | SB4=text |
               PM1=text |
               GSAVE | PMSAVE |
                 { GSEND | SSEND=mask |
                     { GASEND | SASEND=mask | USEND=mask
                     [ IMM={ YES | NO } ] }
             [ ID=nsbroid ] } }
(3)
```

Use: Broadcast Services makes it possible to send messages to terminals under the control of EASINET and, under certain circumstances, to Management Services users. The NSBRO command allows an authorized OCS operator to create and send a general or specific terminal/user broadcast from an OCS screen as an alternative to using the fullscreen panels provided by Broadcast Services.

This command can also be used from within an NCL procedure, although the Broadcast Services NCL interface $BSCALL allows more flexibility. See the chapter, *The Broadcast Services NCL Interface*, in the *Network Control Language Reference* for more information on $BSCALL.

Operands:

GB1=*text* | GB2=*text* | GB3=*text* | GB4=*text*
The text for lines 1, 2, 3, and 4 of a general broadcast. If no text is specified, then any existing text is deleted.

SB1=*text* | SB2=*text* | SB3=*text* | SB4=*text*
The text for lines 1, 2, 3, and 4 of a specific broadcast. If no text is specified, then any existing text is deleted.

PM1=*text*
The text for the Primary Menu broadcast. If no text is specified, the existing text will be deleted. The appearance of this text on a panel is determined by the use of the &ZPMTEXT1 system variable.

GSAVE
Saves the current general broadcast text lines to the VFS file. This enables their values to be restored during Management Services initialization.

PMSAVE
Saves the current Primary Menu text line to the VFS file. This enables the value to be restored during Management Services initialization.

GSEND
Requests sending of the general broadcast text set by the GB1=, GB2=, GB3= and GB4= operands to all EASINET terminal displaying a panel with broadcast lines.

NSBRO

SSEND=*mask*

Requests sending of the specific broadcast text to EASINET terminals selected using the specified *wildcard* mask. Use of the SSEND= operand without setting the broadcast text with previous SB*n*= commands is invalid. After using the SSEND= operand any previously set text is no longer available and new text must be set before the SSEND= operand can again be used.

GASEND

As for GSEND, except that all Management Services users will also be *shoulder-tapped* at the next screen interaction. An NCL procedure named $NMBRO is then executed, which can display the broadcast lines.

SASEND=*mask*

As for SSEND, except that all SOLVE users will also be *shoulder-tapped* at the next screen interaction. An NCL procedure named $NMBRO is then executed, which can perform any required processing. The broadcast lines themselves are not made available to this procedure automatically. Specific action must be taken by your installation to implement the desired action.

USEND=*mask*

Requests *shoulder-tapping* of all Management Services users whose user IDs match the specified wildcard mask. An NCL procedure named $NMBRO is executed at the next screen interaction, and can then perform any required processing. No broadcast lines are set in conjunction with USEND. Note that $NMBRO requires global variables to be set up to contain the broadcast information that is to be issued and it is recommended that you use the Broadcast Services function (selected from the primary menu) to do this. (Set global variables as &&000$NMUBRO1 if using an NCL procedure and NSBRO.)

IMM=YES | <u>NO</u>

Indicates whether the broadcast is to be sent immediately to users (IMM=YES) or to be sent the next time that they take action, for example, pressing the ENTER key.

ID=*nsbroid*

This operand specifies a variable to be passed to the $NMBRO NCL procedure, available in &ZBROID. If not specified, Management Services generates a sequential number.

Examples:

```
NSBRO GB1=Time: 14:40 IMS will be down for 25 minutes.
NSBRO GASEND
NSBRO SB1=Time: 20:40 Perth link down tomorrow for maint.
NSBRO SB2= Please use dial-backup link till further notice.
NSBRO SSEND=P*
NSBRO GB1=
NSBRO GB2=
NSBRO GB3=
NSBRO GB4=
NSBRO GSEND
NSBRO USEND=*
```

Notes:

A general broadcast is sent to all terminals and/or Management Services users. A specific broadcast will only be sent to those terminals or users whose name matches that specified in the *mask* specified with the SSEND= or SASEND= operand. An asterisk in any position of the mask means any character in that position of the name will be accepted. Where characters other than an asterisk are specified, a precise match is required. For example:

- SSEND=P* means all terminals whose name starts with P.
- SSEND=L5**9 means all terminals whose names start with L5 end in a 9 and are exactly five characters long.

The NSBRO command only accepts one of the above operands for each use of the command. To set and send a broadcast, multiple NSBRO commands might be required as in the examples. The GSEND/GASEND operand can be used at any time to send the set general broadcast. The SSEND=/SASEND= operand is only valid after some broadcast text has been set using the SB*n*= operands.

All text following the GB*n*= or SB*n*= operands is used. Blanks after the equals sign will be retained. The maximum length text for each line is 78 characters. If truncation occurs a warning message will be issued. The case of the entered text will be maintained when the broadcast is sent.

An existing broadcast can be deleted by entering the GB*n*= or SB*n*= operands without text. To delete a broadcast from terminals, delete the text and then use the GSEND or SSEND= operand to delete the message from the required terminals by sending a blank message.

NSBRO commands can be used to automate the setting of broadcasts with NCL procedures (although the Broadcast Services exit $BSCALL allows more flexibility—see the *Network Control Language Reference*). Broadcasts can then be set and issued from timer commands. MAI-OC sessions can also be used to send the broadcasts through subsystems such as IMS or CICS. This also makes it possible to append static information such as date and time to broadcast messages using the system variables &DATE*n* and &TIME as provided by the NCL facilities of Management Services.

See Also:

The SHOW BRO command description.
The chapter titled *The Broadcast Services NCL Interface* in the *Network Control Language Reference* for more information on $BSCALL.
The chapter on Broadcast Services in the *Management Services User's Guide* for information on broadcasting to EASINET terminals.

NSLOOKUP

Function: Uses the name service to find the name or address of a host.

NSLOOKUP { ? | *host_name* | *ip_address* }

Use: The NSLOOKUP command is used to find the name or the address of a remote host.

Operands:

?
Displays help in the form of the NSLOOKUP command syntax.

host_name* | *ip_address
The name or IP address of the TCP/IP host about which information is required. (*Mandatory*)

Examples:

```
NSLOOKUP venus
NSLOOKUP billing
NSLOOKUP 192.168.1.87
NSLOOKUP X'C0A80157'
```

Notes: The NSLOOKUP command returns the full name of a host and, if known, its address. This command can take either a real or alias name, or an IP address as input.

In some locations the name service might be configured such that the user can enter the name of the application to which they want to connect (for example, BILLING) and be connected to the appropriate host (for example, MVSSY3). In this case NSLOOKUP BILLING returns information about the current host that is running the BILLING application.

The IP address can be entered an 8-character hex number: *x'nnnnnnnn'*.

NSPCONN DEFINE

Function: The NSPCONN DEFINE command is used to define a NetSpy connection, but not to start it.

NSPCONN DEFINE *conn-name* [{APPLID | APPL}=*appl-name*] (4)

Operands:

conn-name

A 1- to 12-character connection name (PDSNAME format). A connection name must be unique.

appl-name

An optional VTAM application name. This must correspond to the ACB name in the NetSpy configuration file. If this operand is omitted, the application name defaults to the first eight characters of the connection name.

Examples:

```
NSPCONN DEFINE nspconn1
NSPCONN DEFINE nspconn1 appl2
```

See Also: The NSPCONN DELETE, NSPCONN START, and NSPCONN STOP command descriptions.

NSPCONN DELETE

Function: Deletes a connection to a NetSpy system. If the connection name is not known, or is not stopped, then an error message is produced.

NSPCONN DELETE *conn-name* (4)

Operands: ***conn-name***

A 1- to 12-character connection name (PDSNAME format).

Examples:

```
NSPCONN DELETE nspconn1
```

See Also: The NSPCONN DEFINE, NSPCONN START, and NSPCONN STOP command descriptions.

NSPCONN START

Function: Starts a connection to a NetSpy system. If the connection has not been defined, it also defines it.

NSPCONN { START | CONNECT } *conn-name* [{APPLID | APPL}=*appl-name*] (4)

Operands:

conn-name

A 1- to 12-character connection name (PDSNAME format). A connection name must be unique.

appl-name

An optional VTAM application name. This must correspond to the ACB name in the NetSpy configuration file. If this operand is omitted, the application name defaults to the first eight characters of the connection name.

Examples:

```
NSPCONN START nspconn1
NSPCONN START nspconn1 appl2
NSPCONN CONNECT nspconn2
NSPCONN CONNECT nspconn1 app23
```

See Also: The NSPCONN DEFINE, NSPCONN DELETE, and NSPCONN STOP command descriptions.

NSPCONN STOP

Function: Stops a connection to a NetSpy system. An error message is produced if the connection is not known, or is not starting or active.

NSPCONN (4)	{ STOP | DISCONNECT } *conn-name*

Operands: ***conn-name***

A 1- to 12-character connection name (PDSNAME format). A connection name must be unique.

Examples:

```
NSPCONN STOP nspconn1
NSPCONN STOP nspconn1 appl2
NSPCONN DISCONNECT nspconn2
NSPCONN DISCONNECT nspconn1 app23
```

See Also: The NSPCONN DEFINE, NSPCONN DELETE, and NSPCONN START command descriptions.

NTSDBMOD

Function: Alters database records or session keep counts on the NTS database. Controls execution of previously issued NTSDBMOD and SHOW SKEEP commands.

```
NTSDBMOD    { [ PRINAME=priname [ PRINET=prinet ] ] |
                [ SECNAME=secname [ SECNET=secnet ] ] |
                [ REFNET=refnet ] |
                [ SKEEP=nnn ] |
                [ KEEPDATE=YY/MM/DD[.HH.MM.SS ] ] |
                [ CANCEL=nnn ] }
(4)
```

Use: Use the NTSDBMOD command to delete records or alter the *session keep* counts for sessions already on the NTS database. The current session keep values can be displayed through the SHOW SKEEP command (see SHOW command below). NTSDBMOD can also be used to cancel the execution of any previously issued NTSDBMOD or SHOW SKEEP commands.

Operands:

PRINAME=*priname*
Provides the full or partial primary name of the session (or sessions) which require modification. An asterisk as the last character of a name (for example, TSO*) indicates a partial name specification. Either or both PRINAME and SECNAME must be supplied.

PRINET=*prinetid*
Optionally provides the NETID of the network in which the real primary name *priname* is known. If omitted, and *priname* is present, a *prinetid* of the network in which NTS is executing will be assumed.

SECNAME=*secname*
Provides the full or partial secondary name of the session (or sessions) which require modification. An asterisk as the last character of a name (for example, LU*) indicates a partial name specification. Either or both PRINAME and SECNAME must be supplied.

SECNET=*secnetid*
Optionally provides the NETID of the network in which the real secondary name *secname* is known. If omitted, and *secname* is present, a *secnetid* of the network in which NTS is executing will be assumed.

REFNET=*refnet*
Provides the name of the reference network for a session incidence. In an NTS Single Image environment where session awareness data is being received from NTS systems in other networks, it is possible that multiple session incidence records will be logged. Each represents the session from the point of view of a different network. If omitted, the network in which NTS is operating will be assumed.

SKEEP=*nnn*

Specifies the new session keep count for the selected sessions. The value *nnn* must be in the range 0 to 255. If the current number of session incidence records kept for any selected session name pair exceeds the new session keep count provided, then the oldest session information is deleted from the database until the number of session incidences retained equals the new session keep count. If a SKEEP value of 0 is specified, then all selected sessions are deleted in entirety from the database, including all master and cross-reference records.

KEEPDATE=YY/MM/DD[.HH.MM.SS]

Specifies the date and, optionally, the time up to which session records are purged from the NTS Database. The date is entered in the format YY/MM/DD. If the time is specified, it must be specified in the format HH.MM.SS preceded by a full stop (.). If not specified the time will be defaulted to 00.00.00 on the date specified. All session records logged in the database before the specified date and time will be purged by this command.

CANCEL=*nnn*

Specifies that the NTSDBMOD or SHOW SKEEP process with *nnn* as its identifier is to be terminated. Because of the asynchronous and possibly long running nature of the NTSDBMOD and SHOW SKEEP commands, it might be desirable to cancel a previous invocation of the command. When these commands are issued, a unique identifier is allocated to the request. This identifier can be used to specify to NTS which invocation of the NTSDBMOD or SHOW SKEEP command is to be cancelled. Use the SHOW NTSDBMOD command to determine which identifiers are currently in use.

Examples:

```
NTSDBMOD PRINAME=TSO* SKEEP=0
NTSDBMOD SECNAME=LU23 SKEEP=20 REFNET=NETB
NTSDBMOD KEEPDATE=88/12/25.12.00.00
NTSDBMOD CANCEL=5
```

See Also: The SHOW SKEEP and SHOW NTSDBMOD command descriptions.

Function: Alters session trace and logging parameters of active sessions.

```
NTSMOD      NAME={ [ netid.] name | [ netid1.] name1 |
                [ netid2.] name2 }
            [ REFNET=refnet  ]
            [ PCID=nnnn ]
            [ CLOSE [ =ALL ] ]
            [ LOG={FORCE | ALL | EXIT | SUMMARY | DATA |
                       ERROR | NO } ]
            [ RECORD={ ACC | RTM } ]
            [ RESET | NORESET ]
            [ TRACE=(i,f) ]
            [ EXIT={ YES | NO } ]
            [ FORCE ]
(4)
```

Use: The NTSMOD command allows logging and trace options for an active session to be modified, thus overriding any options for the session which might have been extracted from the relevant session class definition.

Operands:

NAME=[*netid*.] *name* | [*netid*1.] *name*1 | [*netid*2.] *name*2

Identifies a particular session, or a particular resource for which all sessions are to be modified. If two names are specified, then only the single session (or multiple sessions for parallel sessions only) between the two resources is the subject of the NTSMOD. If a single name is specified, then all sessions with that resource are affected. A resource name can be network qualified by preceding the resource name with the *netid* and a period. A slash separates two resource names identifying a session pair.

REFNET=*refnet*

Provides the name of a reference network for a session incidence. In an NTS Single Image environment, where session awareness data is being received from NTS systems in other networks, it is possible that multiple session incidence records are in storage, each representing the session from the point of view of a different network. If omitted, the network in which NTS is operating will be assumed.

PCID=*nnnn*

Where *nnnn* is the Procedure Correlation Identifier allocated by VTAM to identify a specific session. Using this operand, it is possible to specify a unique session on which NTSMOD functions can be performed. Use of this operand might be necessary where parallel sessions exist.

CLOSE [=ALL]

This operand is used to remove selected sessions from the NTS environment. Once a session has been closed, NTS will no longer be aware of the existence of the session. Any data associated with the session can be logged to the NTS Database as part of the CLOSE processing. Specifying the CLOSE operand with no associated value will cause NTS to examine the log options current for the selected session(s) and process accordingly. Specifying CLOSE=ALL will cause NTS to override the log options for the session class and log all data associated with the session as part of the CLOSE processing. The CLOSE and LOG operands are mutually exclusive.

LOG={ FORCE | ALL | EXIT | SUMMARY | DATA | ERROR | NO }

This operand is used to specify a new logging disposition for the sessions affected.

LOG=FORCE means the session data is immediately queued for output logging even while the session remains active. Once logging is complete normal session processing again resumes.

LOG=ALL means no immediate activity takes place, but all session data will be written to the NTS database on session end, overriding any class definition options associated with the session.

LOG=FORCE and LOG=ALL will also cause NTS to present any data to be logged in the NTS Database to the NTS User Exit.

LOG=EXIT will cause all session data to be immediately presented to the NTS User Exit without being queued to be logged to the NTS Database.

LOG=SUMMARY causes only accounting and RTM to be logged with the session.

LOG=DATA causes the session to be logged only if some data was accumulated for it.

LOG=ERROR causes the session to be logged only if it ended in error.

LOG=NO causes the session not to be logged at all.

Note

The LOG and CLOSE operands are mutually exclusive.

RECORD={ ACC | RTM }

This operand causes NTS to flag the type of NTS System Management Facility (SMF) record(s) for the specified session(s) as either accounting or response time measurement (RTM) data before it is passed immediately to the NTS User Exit.

If RECORD=ACC was specified, the accounting data held by NTS in the session record will be reset by this command. If RECORD=RTM was specified, the RTM data held by NTS will be reset by this command if the RESET operand was specified. If NOREST is specified then the RTM counts will not be reset. This operand can be used to analyze session accounting and RTM data on an interval basis.

RESET | NORESET

Specify NORESET if you do not want the cluster controllers RTM count to be reset. Specify RESET to reset the RTM counts. Note, this operand is only valid if RECORD=RTM is specified.

TRACE=(*i*,*f*)

Allows the trace queue depths for selected sessions to be modified. The new values, which specify the initial *i* and final *f* queue depths respectively, apply immediately. If the new initial trace queue depth is greater than the current number of PIUs on this queue then the new value is ignored. Any new final value that is within the acceptable limits (defined by the SYSPARMS NTSMAXTP operand) is actioned, and if the queue depth is reduced the oldest PIUs on the final queue are discarded until the new queue depth value is reached.

EXIT={ YES | NO }

EXIT=YES causes all session data to be immediately presented to the NTS User Exit without being queued to be logged to the NTS database.

FORCE

Specifies that the nominated session(s) will be forced to the NTS database and User Exit immediately. All data accumulated to the session will be logged to the database. The subtype of the SMF type 39 record logged to the database will be 04.

Examples:

```
NTSMOD NAME=CICS TRACE=(4,50) LOG=ALL
NTSMOD NAME=NET001.IMS/LU23 LOG=FORCE
NTSMOD NAME=IMS01 RECORD=ACC
```

Notes:

If neither the LOG nor TRACE operands are specified the number of sessions is still reported which can be effectively used to determine the scope of the command before actually modifying any sessions.

When NTS Single Image session awareness or session data sharing is in effect, entering an NTSMOD RECORD= command will cause NTS to route the command to the NTS system in the domain in which the resource is known (the resource's owning domain). The command will be executed and the results routed back to the originating NTS system. This process is transparent.

See Also:

The DEFCLASS command.

OBEYFILE

Function: Invokes the IBM TCP/IP OBEYFILE command—available in NetMaster for TCP/IP only.

OBEYFILE	{ ? | *dataset_name* }

Use: The OBEYFILE command is used to execute the IBM TCP/IP configuration commands.

Operands:

?
Displays help in the form of the OBEYFILE command syntax.

dataset_name
The fully qualified dataset name containing the configuration commands to be executed. (*Mandatory*)

Examples:

```
OBEYFILE SYST.USER0.OBEYFILE
OBEYFILE SYST.OBEYFILE(TRACEON)
```

Notes: Obeyfile processing can be used to start and stop traces and to alter the state of the interface devices as well as altering some configuration parameters.

The appropriate level of access authority is required for NetMaster for TCP/IP to issue the OBEYFILE command.

The contents of the Obeyfile are logged in addition to the command and its results.

OCSID

Function: Sets or resets OCS window identifiers.

OCSID (0)	[*textid*]

Use: When operating with multiple OCS windows, perhaps profiled to perform different functions, it can be difficult to identify which window is currently in use when swapping between windows.

The OCSID is a 1- to 8-character name, set by this command, that will be displayed at the right side of the bottom window framing line.

A different ID should be set for each window to ensure that identification is possible.

Operands: ***textid***

The 1- to 8-character text to be set in the ID area and in the &OCSID NCL system variable. See the discussion on the &OCSID variable below. No restrictions are placed on the characters contained in the selected ID.

Examples:

```
OCSID PRODSYS
```

Notes: Issuing the OCSID command without any text will delete any current setting. Text in excess of the maximum of 8 characters will be truncated.

The OCSID command would normally be issued by an NCL procedure executed as the initial command on entry to OCS mode. As the same procedure is executed on entry to both OCS windows a pair of NCL system variables has been provided (&OCSID and &OCSIDO) to make it possible to test the setting of any current OCSIDs and set the alternative OCSID accordingly.

The order of entry and exit of OCS windows is completely random. An NCL procedure which is to set the OCSID and possibly perform associated functions (for example, TEST and PROD) must be able to determine the OCSID of a new window regardless of when it is entered or how frequently it is entered. The &OCSID system variable supplies the OCSID of this window or null if none has been set. The &OCSIDO supplies the OCSID of any other window that might already be operational if an OCSID has been set for that window.

By testing these variables an NCL procedure can determine exactly what ID is to be set for a new window.

For example:

```
&IF .&OCSIDO NE .PROD OCSID PROD
&IF .&OCSIDO EQ .PROD OCSID TEST
```

This example if executed on entry to OCS will always set the first window to PROD and the second to TEST and maintain this situation regardless of the order of entry and exiting of windows.

See Also: The *Network Control Language User's Guide*.

OPNDST

Function: Connects a specific terminal to SOLVE by using VTAM.

```
OPNDST      name
            [ LOGMODE=logmode ]
            [ RELREQ={ YES | NO } ]
            [ Q={ YES | NO } ]
            [ RELEASE={ YES | NO } ]
            [ DATA=userdata ]
            [ PRINTER={ YES | NO } ]
(1/0)
```

Use: This command can be used to start a session with the specified terminal. For display terminals, unless user data is specified the terminal will display the EASINET logo if that facility is active, otherwise the logon screen will be displayed. For LU-1 terminals the device will be placed under EASINET control only.

Operands:

name

VTAM node name of terminal with which session is to be started.

LOGMODE=*logmode*

Specifies the name of an entry in the LU's log mode table which is to be used for the session. If this operand is omitted, the LU's default log mode entry is used.

RELREQ={ YES | NO }

Specifies that if the terminal is currently logged on to another application, that application's VTAM RELREQ exit is to be driven, if it has one. This notifies the application that the SOLVE region requires the terminal, and it might then choose to disconnect the terminal, in which case it will be connected to the SOLVE region. If the application does not have a RELREQ exit, or chooses not to disconnect the terminal, the logon queues until the application eventually disconnects it.

Q={ YES | NO }

Q=YES specifies that a logon request be queued if the terminal is currently connected to another application. When that application eventually disconnects the terminal, it will be connected to the SOLVE region.

RELEASE={ YES | NO }

Specifies if the terminal is to be released if the SOLVE VTAM RELREQ exit is driven due to another application requiring the terminal. RELEASE=YES will cause the terminal to be released by the SOLVE region. Any terminal with a signed-on user will not be released regardless of the use of this operand. RELEASE=YES is the default.

DATA=*userdata*

If entered, this must be the last operand. All text following the operand will appear as user data when the terminal is connected to the SOLVE region.

PRINTER={ YES | NO }

This operand can be used to connect a printer, without requiring changes to EASINET. It enables printing support in SOLVE systems that are not licensed for EASINET.

If PRINTER=YES is specified, an LU1 device which is connected to the SOLVE region is not passed to EASINET, and is not presented with the logo message. Any user data specified on the OPNDST command is ignored, and the resulting session is treated as a single bracket in SNA terms.

Note

If PRINTER=YES is specified, a command authority of 0 is required. Otherwise, a command authority of 1 is required.

The &WRITE verb can be used to send output to the device while it is connected to the SOLVE region. The device can be released from the SOLVE region by the CLSDST command.

This operand is ignored if the device is not an LU1.

Examples:

```
OPNDST LU01 LOGMODE=M43278
OPNDST TERM3 RELREQ=YES
OPNDST TERM1 LOGMODE=M2SNAQ Q=YES DATA=MYUSRID MYPSWRD U.A
OPNDST TERM6 DATA=EASINET QUERY ABR54S
```

See Also: The CLSDST command description.

OPSYS

Function: Passes a command to the operating system for execution.

OPSYS (2)	[CP]	*text of command*

Use: The system allows an authorized user to pass a command to the operating system.

Operands:

CP
The command is passed to CP for execution (for VM only).

text of command
Text as it is to be passed to the operating system.

Examples:

```
OPSYS CANCEL U=TS022
OPSYS $AJ54
OPSYS CP cp-command
```

Notes: The OPSYS command is supported on the following operating systems:

- MVS/XA, MVS/ESA, OS/390, and z/OS
- MSP, MSP/AE, MSP/EX
- XSP
- VOS3
- VM/GCS (including XA and ESA variants)

To aid in the discussion below:

- z/OS, OS/390, and VOS3 are regarded as MVS
- VM/GCS is regarded as VM.

Under MVS, an indication of the acceptance of the command can be given. In VM systems the command is either passed to GCS for execution within the VM machine in which the SOLVE system is executing, or, if CP is specified, the command is passed to CP for execution.

For the OPSYS command to be used the system must be APF authorized. This subject is discussed in your *Management Services Administrator Guide*.

OPSYS commands can be included in the INIT and READY procedures executed during system initialization.

In MVS or VM systems when CP is not specified, responses to commands passed to the operating system will not be received at the SOLVE terminal. In a VM system, if OPSYS with the CP operand is used to pass a command to CP for execution, and the command is one which returns a response, then the results of the command will be returned to the OCS window as a series of variable length messages.

ORDER

Function: Reorders your OCS window display from top to bottom.

```
ORDER
(0)
```

Use: During OCS operation it is possible that a group of messages start part way down a window and 'roll' on, continuing at the top and being followed by the roll-delimiter line. The ORDER command can then be used to reformat the window so that the screen is filled (where enough messages have been received) from the top to bottom with the most recent message appearing last (that is, in chronological order).

Notes: This is a useful technique for re-sequencing commands before printing a screen image of, for example, a VTAM display.

The ORDER command is assigned by default to F12.

The ORDER command will also cause reformatting of the Non-Roll-Delete area if non-roll-delete messages are present.

PAGE

Function: Ensures that any following output messages start at the top of the OCS window, and any existing display lines there are erased.

```
PAGE
(0)
```

Use: When processing NCL procedures that contain many lines of OCS output, it is better to divide the output into pages and to commence each page at the top of the OCS window. The PAGE command allows the synchronization of output with the erasing of the screen. When this is combined with the &PAUSE command full control of display output is possible.

Notes: Operating the PAGE command is similar to the CLEAR command except that the screen clearing function will occur at the precise point at which it is processed. The CLEAR command is an expedited command and will be processed ahead of other commands that are queued for output.

Another major difference is found when using the PAGE command when AUTOHOLD is set ON. In this case the operator will be prompted with the normal AUTOHOLD signal before the screen is erased, to ensure any previous messages have been observed. This makes the PAGE command ideal for concatenation in timer commands to ensure that the operator is prompted before the screen is erased.

See Also: The &PAUSE verb description in the *Network Control Language Reference*.

The K and CLEAR command descriptions.

PF

Function: Displays and sets terminal function keys.

PF	[LIST | SLIST]
PF*nn*	[IMM | [IMM,] *text* | RESET | CONV,*text* | PREF,*text* | SUFF,*text*]

Use: This command lets you display and set any one of up to 24 function keys.

Four types of function keys are supported:

- **IMM -** immediate. Text is executed immediately the function key is pressed. If no text is entered the key acts as an ENTER key.
- **CONV -** conversational. Text is displayed in the command line for modification and only executes once the ENTER key is pressed.
- **PREF -** prefix. The text associated with the function key will be prefixed to any data in the line occupied by the cursor when the key is pressed.
- **SUFF -** suffix. The text associated with the function key is added to the text in the line occupied by the cursor when the key is pressed.

Prefix and suffix keys take as their source data the contents of the line on the screen where the cursor is positioned when the function key is pressed. This means that if the cursor is in the command line, the contents of the command line will be used and if the cursor is anywhere else in the OCS display area the contents of that line will be used.

The number of the function key to be set is specified as *nn* following the PF command keyword (for example, PF04). If the CONV, PREF, or SUFF operands are not specified with a text, an immediate function key will be set.

Operands:

LIST
List all current function key settings. No function key number (*nn*) should be specified for the list function.

SLIST
List all current settings for function keys 1 to 12 only (short list). No function key number (*nn*) should be specified for the list function.

nn
Number of function key that is to be set, in the range 1 to 24.

IMM | IMM
Specifies that an immediate function key is being set. This is the default. If you do not enter any text the key will act as an ENTER key, that is, it will cause input activity with no associated text.

text
Text to be associated with PF*nn*.

RESET
If RESET is specified (or no other operands are specified), then the PF key setting is reset.

CONV
Specifies that a conversational function key is being set.

PREF
Specifies that a prefix function key is being set.

SUFF
Specifies that a suffix function key is being set.

Examples:

```
PF
PF LIST
PF SLIST
PF4 D BFRUSE
PF04 D BFRSHOW USERS
PF24 CONV,D NET,ID= PF12 IMM
PF10 SUFF,+Production
PF13 PREF,MSG USER1+
```

Notes:

When setting a prefix key you might want to have a blank at the end of the function key text so that when it is appended to the text from the command line the correct format results. This can be achieved by the addition of a plus sign to the end of the function key text. This plus sign will be interpreted as a blank when the function key is processed. For example:

```
PF5 PREF,MSG USER2+
```

When setting a suffix key, a plus sign at the front of the suffix text will be interpreted as a blank when the suffix text is appended to the end of the function key text. For example:

```
PF5 SUFF,+FROM USER2
```

Concatenated commands can be set on a function key by specification of a colon (:) as the concatenation character rather than a semicolon (;) which is the normal concatenation character. The colons will be translated to semicolons when the function key text is stored. For example:

```
PF7 CLEAR:D BFRUSE:D IMS,E
```

When a conversational function key is pressed the cursor is normally positioned at the end of the command text in the command line. Precise positioning of the cursor is possible by placing the cursor in the exact position it is required when the PF*nn* command is entered. Thus, if you require the cursor to be positioned part way along the text, then key in the text and backspace the cursor to the required position prior to pressing the enter key to set the new function key value. For example:

```
PF8 CONV,D NET,ID=_:D IMS,E
```

When an IMMEDIATE function key is defined without text it becomes the equivalent of another ENTER key. When it is pressed, no text value is associated with the key but an input operation occurs which is the same as pressing the ENTER key. See the ENTER command for details of ENTER key reassignment.

When a user first enters Operator Console Services (OCS) mode the global function key values are in effect, until changed by the user. This change can be performed either by individually entering the PF*nn* commands or by bulk entry from an NCL procedure using the EXEC command. Any changes made by a user do not affect the global function key values and the changes only remain in effect until the user exits from OCS mode.

The global function key values are changed by including the appropriate PF*nn* commands in the NMINIT procedure which is automatically executed during system initialization. When setting global function keys it is suggested that the default values associated with function keys 2, 9, 14, and 21 are not changed as they will impact the use of split screen operation and will not be consistent with split screen operation outside of OCS mode.

The specialized form of the function command PFKS, will give a short list of the function key settings (1 to 12 only). It is synonymous with PF SLIST.

An alternative to the use of function keys is the EQUATE command.

Function key settings always execute in accordance with the user's CMDSEP PROFILE option, as at the time the function key value was set. See the description of command concatenation in the chapter *The Operator Console (OCS) Environment* in the *Management Services User's Guide*.

When setting a function key, the profile of the user is retained when the function key is used. That is, if CMDSEP was set to YES when the function key was set, it will remain as YES when the function key is used - even if the user has changed his profile in the meantime.

See Also: The EQUATE and ENTER command descriptions.

The chapter about OCS in the *Management Services User's Guide*.

PING

Function: Sends echo requests to a remote host.

```
PING        { ? | host_name | ip_address }
            [ COUNT={ 3 | count } ]
            [ SIZE={ 256 | packetsize } ]
            [ WAIT={ 5 | seconds }]
            [ VERBOSE={NO | YES}]
```

Use: The PING command is used primarily to test whether a host is reachable through the network.

Operands:

?
Displays help in the form of the PING command syntax.

host_name* | *ip_address
The name or IP address of the TCP/IP host to which the test is to be applied. (*Mandatory*)

COUNT={ 3 | *count* }
The number of echo requests to send to the TCP/IP host. If no count value is specified, then the default value of 3 is assumed. If a value is supplied, it must be in the range 1to 999.

SIZE={ 256 | *packetsize* }
The size of each echo request message to be sent to the TCP/IP host. If no packet size value is specified, then the default value of 256 is assumed. If a value is supplied, it must be in the range 16 to 2048.

WAIT={ 5 | *seconds* }
The number of seconds to wait for a response to each echo request. If no wait value is specified, then the default value of 5 is assumed.

VERBOSE={ NO | YES }
The kind of output required. If YES is specified, then full details of the ping response are returned. If this operand is allowed to default to NO, then only summary results are returned.

Examples:

```
PING venus
PING 192.168.8.127 COUNT=5 VERBOSE=YES
```

Notes: The PING command is a simple and effective means of diagnosing connectivity and performance related problems on a TCP/IP network.

PPO

Function: Starts and stops the VTAM PPO interface.

```
PPO          { START | STOP }
(3)
```

Use: SOLVE regions support the VTAM PPO interface to receive unsolicited messages about network events. These messages would normally be directed by VTAM to the operating system console where they could be lost amongst the multitude of other messages. Entering the PPO START command determines that the SOLVE region is authorized to take control of these messages.

Operator Console Services (OCS) operators will only receive PPO messages if authorized to do so. This authorization is a function of the user ID definition. After the PPO START command has been entered, the SOLVE region automatically directs any PPO messages to those OCS terminals that are authorized to receive them. These messages are presented to OCS terminals timestamped and highlighted, and the terminal alarm rings. The messages can optionally be delivered in a particular color. They will also be recorded in the activity log.

The delivery of unsolicited messages can also be controlled by the Network Partitioning Facility (NPF).

Operands:

START
Commence receiving PPO messages.

STOP
Stop receiving PPO messages.

Examples:

```
PPO START
PPO STOP
```

Notes: The PPO START command is normally placed in the NMREADY procedure that is automatically executed during system initialization. If VTAM has been terminated and restarted without Management Services being recycled, a PPO START command must be issued to restart the VTAM PPO interface.

Receipt of PPO messages at the system console can still be achieved by giving the SYSOPER user ID definition (allocated specifically for the console interface) the authority to receive PPO messages. Removal of this authority will suppress the PPO messages from the system console. (Your installation might have chosen an alternative name for the system console). If the SOLVE region receives PPO messages from VTAM and has no one else to deliver them to they will be sent to the console as a last resort.

Individual OCS operators that are authorized to receive PPO messages can control the receipt of such messages using the PROFILE command. In some cases it might be desirable for an operator to temporarily suppress the receipt of PPO messages where they might interfere with other work. The PROFILE command PPO= operand is provided for this purpose.

The PROFILE command also supports the specification of a severity level to provide further flexibility in the control of message delivery.

See Also: The PROFILE command PPO operand.

The chapter on OCS in the *Management Services User's Guide*.

PROFILE

Function: Displays or modifies a user profile.

```
PROFILE     [ AOMFMTRC={ YES | NO } ]
            [ AOMMSG={ YES | NO } ]
            [ AOMMSGLV={ ALL | NONE |* | list } ]
            [ AOMPRFJI={ YES | NO } ]
            [ AOMPRFJN={ YES | NO } ]
            [ AOMPRFMN={ YES | NO } ]
            [ AOMPRFSN={ YES | NO } ]
            [ AOMPRFTM={ YES | NO } ]
            [ AOMPRFUI={ YES | NO } ]
            [ AOMPRFUN={ YES | NO } ]
            [ CMDKEEP={ YES | NO } ]
            [ CMDSEP={ YES | NO } ]
            [ CMDSTACK=n ]
            [ DELCHAR=c ]
            [ EDS= ]
            [ ENV={ CURRENT | PRIMARY } ]
            [ EXIT=YES | NO | WARN ]
            [ FTSMSG={ YES | NO } ]
            [ HOLD=nnn ]
            [ INDENT=nnn ]
            [ INITCMD=cmdstring ]
            [ MONMSG={ YES | NO } ]
            [ MSG={ YES | NO } ]
            [ MSGALARM=n ]
            [ MSGCODE=xx ]
            [ MSGPROC={ name | FLUSH } ]
            [ NCLTEST={ YES | NO } ]
            [ NCLUCASE={ YES | NO } ]
            [ NRDELCH=c ]
            [ PPO={ YES | NO | INFO | WARN | NORM | SER } ]
            [ PREFSYS={ YES | NO } ]
            [ RDMIN=nn ]
            [ ROLL=nn ]
            [ ROUTCDE={ ALL | NONE | list | * } ]
            [ UNSOL={ YES | NO } ]
(0/2)       [ VTAMNUM={ YES | NO } ]
```

Use: Each OCS window and dependent (&INTCMD) environment has associated with it a profile, which determines the basic parameters that control the way in which the OCS operator or dependent environment functions. The profile is established by a combination of the details specified when the USERID definition is created and by variables that are modified by the user with the PROFILE command during execution.

The current profile can be displayed by entering the PROFILE command without any update operands. Modifications are made to a profile by entering the PROFILE command with one or more operands.

The operands can be entered in any combination and in any order. Modifications made to a profile are effective only until the user exits from OCS. An exception to this is the value associated with the INITCMD operand. This operand can be used to set other normally desired profile operands automatically on entry to OCS.

For dependent environments, profiling is valid until an &INTCLEAR TYPE=ALL is issued.

Operands:

AOMFMTRC={ YES | NO }

Specifies if AOM messages delivered to this OCS window are to have the internal SOLVE Routing Code (NMROUTCDE) prefixed to the message in the form RC=list.

This operand is useful when determining which message routing codes are of interest. By viewing the routing codes assigned to various messages, a list of those of interest can be determined.

AOMMSG={ YES | NO }

Used to control the delivery of AOM messages to an OCS window.

A user ID with AOM authority can be profiled to receive AOM messages when in OCS mode. A user, so profiled, can stop these messages being sent to the OCS window by specifying AOMMSG=NO.

AOMMSGLV={ ALL | NONE | * | *list* }

Specifies the MVS message level for messages delivered to this OCS window. If NONE is specified then no messages are delivered to this window.

The highest classification of message level that a user can receive is defined as part of the user ID definition. The AOMMSGLV operand can be used to set a subset of that highest classification. An attempt to profile for a value that the user is not authorized for is rejected.

Specifying an asterisk (*) results in the message levels being reinstated to the default user ID values.

Specifying ALL sets all message levels on. This operand is valid only if the UAMS definition also specified ALL for message levels.

Specifying NONE sets all message levels off. This blocks receipt of all messages containing AOM information.

The individual message level values that can be specified are described in the appendix *Message Levels* in the *Automation Services Common User Guide*. If more than one value is desired, then a list of values, surrounded by parentheses and separated by commas, can be specified.

PROFILE

AOMPRFJI={ YES | NO }

Specifies if the JES job ID of the job that this message relates to (if available) is to be prefixed to the message when it is delivered to this OCS window. If no ID is available, eight blanks are used instead. The prefixed ID is in the format:

`JOB` *nnnn*
`STC` *nnnn*
`TSU` *nnnn*

Leading zeroes are suppressed.

This operand applies to AOM messages sourced from z/OS and OS/390 systems only.

AOMPRFJN={ YES | NO }

Specifies whether the job name that issued a message, is to be prefixed to the message when it is delivered to this OCS window.

This operand applies to AOM messages sourced from z/OS, OS/390, MSP, and VOS3 systems only.

AOMPRFMN={ YES | NO }

Specifies whether the minor lines of MVS-sourced multi-line WTO messages are to have AOM prefixes inserted when displayed.

AOMPRFSN={ YES | NO }

Specifies whether the originating system name is to be prefixed to the message when it is delivered to this OCS window.

This operand applies to AOM messages sourced from z/OS and OS/390 systems only.

AOMPRFTM={ YES | NO }

Specifies whether the time a message was issued is to be prefixed to the message when it is delivered to this OCS window.

YES or HMS indicates that the time, accurate to 1 second, is to be displayed in the format HH:MM:SS.

HM indicates that the time, accurate to 1 minute, is to be displayed in the format HH:MM.

AOMPRFUI={ YES | NO }

Specifies whether VM-sourced AOM messages are to have the VM user ID prefixed.

This operand applies to AOM messages sourced from VM systems only.

AOMPRFUN={ YES | NO }

Specifies whether VM-sourced AOM messages are to have the VM RSCS user node prefixed. The eight-character node value has a colon (:) suffixed.

This operand applies to AOM messages sourced from VM systems only.

CMDKEEP={ YES | NO }

By default the system maintains the last command entered in OCS mode on the command line. This is designed to allow rapid re-entry of a command. Under some circumstances it might be desirable for the command to be erased from the command line and the cursor repositioned. The CMDKEEP=NO operand can be used to set this.

CMDSEP={ YES | NO }

Specifies whether semicolons (;) are to be recognized as command separators on input. See the chapter on OCS in the *Management Services User's Guide* for further information.

CMDSTACK=*n* (number in range 1 to 32)

Commands entered from an OCS screen are automatically maintained in a command stack that allows easy recall of commands into the command line using the CS+ and CS- commands. The default size of the command stack is 6. This operand allows the user to adjust the size of the command stack within a range 1 to 32 inclusive.

DELCHAR=*c* (any 1 character)

This operand determines the character that is to be used to construct the delimiter line that is used to separate new output from previous output as the screen display rolls from the top to the bottom of the screen. Any single display character can be used, including a blank. When operating in either horizontal or vertical split screen mode it might be desirable to use different delimiter characters in each of the OCS windows to distinguish output in the different windows.

EDS=(see PROFILE EDS command for options)

The EDS operand is used to define an EDS profile within the CURRENT executing environment. It is mutually exclusive with all other operands described here for the PROFILE command. For a complete description of PROFILE EDS, see that command in this manual.

ENV={ CURRENT | PRIMARY }

Specifies whether the profile command applies to the current executing environment (this is the default) or to the OCS window (PRIMARY). It is applicable only in NCL procedures that issue PROFILE via &INTCMD. ENV=CURRENT allows the internal command environment to be profiled differently from the real OCS window. Profile privileges cannot supersede the user ID's initial values. Screen formatting operands are rejected in an internal command environment; for example, CMDKEEP,ROLL,RDMIN.

EXIT={ YES | NO | WARN }

Specifies the action to be taken when OCS termination is requested and NCL processes (other than scripts) are active. If EXIT=NO is set, then OCS termination is prevented and a message is issued. EXIT=WARN issues a warning message on the first attempt and a second exit request continues the termination. EXIT=YES is the default and allows termination to proceed.

FTSMSG={ YES | NO }

Specifies whether the user wishes to receive notification of the commencement and completion of transmissions by File Transmission Services. Notification is for all transmissions not just those initiated by the user.

HOLD=*nnn* (number in the range 10 to 999)

Determines the maximum number of messages that are to be held when the screen is in either AUTOHOLD or HOLDING state. The system default is normally 200. The operator is warned when the number of held messages reaches 75% of the value specified in the HOLD operand, and again at the 100% mark. When the value specified in the HOLD operand is exceeded, a MSGS LOST warning is given to the operator. At this point the oldest held message is discarded and the new message held. The operator is then re-warned every 10 messages. The held messages can be released by the entry of any input from the keyboard. See the AUTOHOLD command for more information.

For dependent environments, this operand can be used to modify the response and request queue limits. The default queueing limit set by NMIQLIM system parameter can be overridden by this means for a particular environment.

Note

The range of the HOLD operand might be restricted in your installation by the SYSPARMS HELDMSG command. See your *Management Services Administrator Guide* for details.

INDENT=*nnn* (number in the range 0 to 999)

OCS messages which occupy more than one line can be more easily identified by operators if all lines other than the first are indented. This operand defines the number of columns which these messages are to be indented. The default value is 0. The maximum value is the number of columns available minus 10.

If the screen is split vertically, and the OCS window is not wide enough to perform the required indentation, the indentation is reduced automatically, and is restored if the window width is increased again.

INITCMD=(*command-string* | ‘*command string*’)

On entry to Operator Console Services, the system automatically executes an ‘initial command’ if one has been defined. This command can be specified either when the USERID is defined or by using the INITCMD operand. The information entered in the INITCMD operand is retained and is not reset on exit from OCS. Up to 40 characters of command string can be entered. If the string being entered contains any blanks, then the entire string must be enclosed in single quotes (’). Concatenated commands can be entered, but the concatenation character must be entered as a colon (:). This is translated to a semicolon (;) when the command is stored. This operand always applies to the primary environment, regardless of the ENV operand.

MONMSG={ YES | NO }

Specifies whether the user wishes to receive Monitor class messages. The user ID must have the monitor privilege initially.

MSG={ <u>YES</u> | NO }

Specifies whether the environment is to receive general broadcast messages (for example, &WRITE ALL=YES) and messages generated via the MSG command. MSG=NO blocks the receipt of broadcast messages and prevents other users from sending messages using MSG ALL or MSG user ID to this environment.

MSGALARM=*n* (number in the range 0 to 9)

Specifies whether the terminal alarm is to sound when data arrives at an OCS window that is closed. The terminal alarm can sound from 1 to 9 times to alert the operator about the arrival of data at the closed window. A value of 0 (the default) specifies that this facility is not to be used.

MSGCODE=*xx* (two hexadecimal digits in the range 00 to FF)

Specifies an installation-defined mask that can be used to control the receipt of messages from NCL procedures generated by the &WRITE MSGCODE=*xx* format statement. The default value is set in the security definition for the user ID. The user can change this value to a lower subset to further limit messages received. To obtain a higher level of authorization the user's security definition must be altered.

For further information on the use of this facility, see the &WRITE verb description in the *Network Control Language Reference* manual.

MSGPROC={ *name* | FLUSH }

Specifies whether the user's current message-processing procedure (MSGPROC) is to be terminated or, if no MSGPROC is currently active, whether the specified procedure is to be started as the new MSGPROC. This operand of the PROFILE command requires a command authority of 2, by default (or as set by your installation).

NCLTEST={ YES | NO }

This operand determines if an NCL procedure, executed in the current OCS window, is to be a copy of a PRELOADed or AUTOSHAREd procedure, or if a private copy is to be used. Specify NCLTEST=YES to use a private copy, or NCLTEST=NO to use any copies already loaded. The NCLTEST command can also be used to set/reset the NCL TEST mode.

NCLUCASE={ YES | NO }

When a START, FSPROC, or EXEC command is entered from OCS, the parameters passed to the procedure are converted to uppercase. The PROFILE NCLUCASE=NO command can be used to prevent this happening.

NRDELCH=*c* (any 1 character)

This operand determines the character that is to be used to construct the delimiter line. The delimiter line is used to separate NRD messages that have made it to the top of the screen, from messages in the roll area. It might be desirable to use a different character from that used for the roll-delimiter-line.

PPO={ YES | NO | INFO | WARN | NORM | SER }

Determines whether the OCS Operator is to receive unsolicited messages from VTAM about events in the network (YES or NO). The INFO, WARN, NORM, and SER values allow the user to select the severity level of the unsolicited messages that are to be received. Only those messages of the selected level or of a higher level are displayed. For example, if PPO=NORM is specified, only messages of normal or serious severity level are displayed.

The severity levels of the individual messages are as documented in the *VTAM Messages and Codes* manual. Regardless of the selected level, any messages classified as *unsuppressable* by VTAM are still displayed. The severity level selected also applies to those messages that are delivered as 'undeliverable messages' (UDMs).

If NO is specified, the current severity level is preserved and then is reinstated when PPO=YES is specified. The default value for the PPO operand is specified in the user ID definition, defined by the installation. The PPO operand is valid only for users whose user ID has been authorized to receive PPO messages.

PREFSYS={ YES | NO }

Specifies whether all messages are to be prefixed with the SOLVE domain ID where the message originated.

Note

When changing this operand, messages already displayed or already pending display are not reformatted.

RDMIN=*nn* (number in range 1 to 99)

Specifies the minimum number of lines within which the rolling messages area is to operate. This command is used to limit the size of the NRD message area to such a size that the rolling message area of the screen can be deemed functionally acceptable by the user.

If the NRD message area is not large enough to accommodate all NRD messages, then an NRD PEND caption appears, indicating that it is not possible to display all NRD messages.

ROLL=*nn* (number in the range 1 to 25)

Determines the period in seconds of the pause to be used when the output display wraps from the bottom display line back to the top line. The initial default value for this operand is 1 second. The roll delay period operates independently of the AUTOHOLD facility.

ROUTCDE={ ALL | NONE | *list* | * }

If a user is authorized to receive AOM messages, then a list of console routing codes is also specified, when the user ID is defined, to control which messages are to be delivered to the user's OCS window.

By using the ROUTCDE operand, a user can specify a subset of the allowed list of routing codes to further control the AOM messages that are to be delivered to the OCS window. Specifying NONE effectively turns off the delivery of messages, while specifying a subset list reduces the number of messages being delivered. For a full description of the allowable values, see the appendix *Message Routing Codes* of the *Automation Services Common User Guide*.

The valid format of a list of routing codes is:

n
(*n*,*n*,*n*)
(*n*-*m*,*n*) and so on

where *n* is from 1 to 128, and *n*-*m* sets all routing codes from *n* to *m*; for example, 3-7 sets 3, 4, 5, 6, and 7.

Note

You can subset your initial routing codes only; for example, if your user ID is profiled for 1-5, you cannot set 6. A message must also pass the AOMMSGLV screen to be delivered (except a console broadcast, which needs to pass message level screening only).

Specifying ALL or a list of Routing Codes while AOM message receipt is turned off turns AOM message receipt on again.

Specifying an asterisk (*) results in the routing codes being reinstated to their originally defined values.

Messages with the BROADCAST AOM message level are not subject to ROUTCDE filtering. They are delivered if the user profile indicates broadcast message level messages are wanted.

Messages containing no routing codes themselves are not subject to ROUTCDE filtering.

UNSOL={ YES | NO }

Specifies whether the environment is to receive any unsolicited messages at all. This provides an overriding control for all unsolicited message delivery from all sources, such as PPO, AOM, monitor messages, and broadcasts. By setting UNSOL=NO, an environment can block unsolicited message receipt, without altering individual message receipt profiling options. Unsolicited message flow can be restored by setting UNSOL=YES.

VTAMNUM={ <u>YES</u> | NO }

Determines if message numbers for VTAM messages are to be displayed. All solicited VTAM displays contain a message number on each line. The number is often a large proportion of the data displayed, and is normally not required.

Suppression of the numbers can speed up displays on remote terminals, and is ideal when operating the screen in vertical split mode, as it significantly reduces the width of each line of data being displayed.

Unsolicited messages received from VTAM do not have their messages suppressed, regardless of the setting of this operand.

PROFILE

Examples:

```
PROFILE
PROFILE CMDKEEP=NO CMDSTACK=9
PROFILE HOLD=2 INITCMD=‘PR VTAMNUM=NO DELCHAR=*:SHOW USERS’
PROFILE INITCMD=‘EXEC MYPFK’
PROFILE ENV=CURRENT FTSMSG=YES
PROFILE PPO=SER FTSMSG=NO
PROFILE AOMPRFJN=Y AOMPRFTM=HM ROUCDE=*
PROFILE ROUTCDE=(1-10,12-128) AOMMSGLV=(CE,I,WTOR) AOMPRFTM=YES
```

Notes:

Although a user’s MONITOR status and time zone details are displayed in the profile, they cannot be changed by using the PROFILE command. They can be changed only by updating the user’s user ID definition.

Certain operands might not be accepted if your system is not authorized for all features. Additional PROFILE command operands associated with optional SOLVE features are described in the relevant feature-related manuals.

Caution should be used when profiling internal command environments that are running under message-generating environments such as the Background monitor or AOMPROC. It is possible by profiling these environments to receive the same class of message as the primary environment generates to produce recursive message generation. For example, profiling an internal command environment under the background monitor to receive monitor messages and for it to then issue an &WRITE would result in this message being received by the sending environment, a potentially self-perpetuating situation.

The AOMFMTRC, AOMPRFJI, AOMPRFJN, AOMPRFMN, AOMPRFSN, AOMPRFUI, AOMPRFUN, and AOMPRFTM operands affect the content of the message as displayed at the terminal. If the user is running a MSGPROC, however, the message text that MSGPROC sees is not affected, as the prefixing is done after MSGPROC processes the message. If MSGPROC requires the values of these prefixes, they are available in system variables.

The initial value of the AOMPRF*xx* operands is the same as the current SYSPARMS of the same name at the time a user enters OCS.

Dependent processing environments can independently set profile options for AOM messages.

See Also:

The AUTOHOLD, NCLTEST, and PROFILE EDS command descriptions.

PROFILE EDS

Function: Declares or revokes interest in event notification.

```
PROFILE EDS     { ENABLE | DISABLE }=event profile name
                [ NAME=name ]
                [ SCOPE={ SYSTEM | REGION } ]
                [ TYPE={ ALL | [ ( ] [ APPLICATION ]
                     [ ,SERVICEABILITY ]
                     [ ,CONFIGURATION ]
                     [ ,ACCESS ]
                     [ ,PROCEDURAL ]
                     [ ,UTILIZATION ] [ ) ] } ]
                [ OBJECT=object ]
                [ RESOURCE=resource ]
                [ REFERENCE=refcde ]
                [ ROUTCDE=routcde ]
(0)
```

Use: Profiles a process for event receipt. Event notification can result from a system event or from an &EVENT NCL statement. The PROFILE EDS command establishes filters for the event attributes. By careful use of event profiles, a process can restrict its view of external events to only those in which it is directly interested, thus minimizing processing overheads associated with redundant messages within the NCL procedure.

The PROFILE EDS command enables event notification to the *current* environment only.

When an NCL procedure terminates, all event profiles associated with it are deleted.

Operands:

ENABLE=*event profile name*

Each EDS profile must be associated with a user-defined name specified on the ENABLE operand. This name is used only to allow the profile to be removed from an environment by a subsequent PROFILE EDS command with the DISABLE operand.

The *event profile name* can have between 1 and 32 characters. Allowable characters are alphanumerics (including national characters), period (.), and underscore (_). Asterisks (*) are not allowed.

DISABLE=*event profile name*

Removes a previously ENABLED event profile definition from the *current* environment. The profile name specified on the DISABLE operand can be generic. This allows groups of profiles to be disabled, or allows all event profiling to be stopped by using the DISABLE=* operand.

The *event profile name* can have between 1 and 32 characters. Allowable characters are alphanumerics (including national characters), period (.), and underscore (_). Asterisks (*) are not allowed, except as a wild character in a generic DISABLE command.

NAME=*name*

Specifies a 1- to 32-byte event name mask.

SCOPE={ SYSTEM | REGION }

The default scope of SYSTEM means that only events with a scope of SYSTEM are received. REGION indicates that only REGION events are received.

TYPE=*event-type*

This operand is a high-level event category which describes the nature of the event.

Valid types are:

APPLICATION

User defined. (This is the default.)

SERVICEABILITY

Faults, errors, availability, degradation, recovery

CONFIGURATION

Object definition, relationship notifies

ACCESS

Security alarms

PROCEDURAL

Scheduling, process control

UTILIZATION

Statistics, raw performance and accounting data, RTM

ALL

All event types will be notified or revoked.

OBJECT=*object*

Specifies a 1- to 32-byte object class. Object classes categorize resources associated with an event.

RESOURCE=*resource*

Specifies a mask for a resource instance, or a pair of resource instances, separated by a comma. If a resource instance pair is specified, then both partners must be present in the event object pair, but they could be present in any order.

REFERENCE=*event reference code*

Specifies a 1- to 32-byte event code mask.

ROUTCDE=*routcde*

Specifies a list of values from 1 to 128, representing event route codes. If *routcde* is specified, then route codes for an incoming event, the profile route codes, and match have an OR operation performed on them. This happens if at least one of the codes is set in each route code string.

Examples:

```
&INTCMD PROFILE EDS DISABLE=*     -* disable all event profiles
&INTCMD PROFILE EDS ENA=EVPROF01 NAME=USERSRCE +
    REF=&UCODE TYPE=APPLICATION
```

Notes:

Operands of the PROFILE EDS command should not appear together with the other operands of the PROFILE command.

The PROFILE EDS command should be issued by the &INTCMD statement to ensure that the message mode environment is established.

NAME, RESOURCE, REF, and OBJECT can have between 1 and 32 characters. Allowable characters are alphanumerics (including national characters), period (.), and underscore (_). Asterisks (*) can be used as wild characters to specify a generic string. An asterisk appearing at the end of a string truncates the attribute mask to the string length preceding the asterisk.

Mask operands that are not specified default to asterisk.

See Also:

The SHOW EDS command.

The &EVENT verb description in the *Network Control Language Reference*.

PURGE

Function: Purges and deletes current timer-initiated commands or NCL locks, or retained Non-Roll-Delete (NRD) messages.

```
PURGE        { TIMER={ nn | OWN | ALL | tid } |
               LOCK=nn |
               NRD={ domainid/n | domainid/* | n | nH | nM | nD | nN } |
               APPLSTAT=applname }
(0/2)
```

Use: Timer initiated commands created by AT and EVERY commands can be purged or deleted in one of three ways. If when the timer command was established the KEEP operand was not specified the timer command will automatically be deleted at the first attempt at execution if the responsible user ID is no longer logged on. If the KEEP operand was specified the command will be kept and the results redirected accordingly.

If the LIMIT operand was specified the command will be automatically deleted when the count of the number of times the command has been executed reaches the specified limit. The third means of deleting or purging a timer command is by the PURGE command.

The PURGE command can also be used to release an NCL process which has been forced to wait during the execution of an &LOCK verb.

NRDs retained for redisplay, can also be deleted by the PURGE command.

Operands: **TIMER={ *nn* | OWN | ALL | *tid* }**

Timer=*nn* can be used to specify the system assigned timer ID number. Specifying this operand will purge the specified timer command.

TIMER=OWN will purge all of the requesting user's timer commands.

TIMER=ALL will purge ALL timer commands in the system. This form of the command requires by default authority level 2.

TIMER=*tid* will purge the timer command assigned the specific identifier *tid*. This is a user-assigned identifier for the timer. (Timer IDs can be displayed using the SHOW TIMER command.)

LOCK=*nn*

Purge the nominated request for a resource. The &LOCK NCL statement can be used to request access to a resource. If the resource is unavailable, the requestor can elect to wait for an interval, or forever, for the resource access to be granted. If this causes conflicts or hang conditions between other NCL processes the PURGE command will cancel the &LOCK statement and the waiting NCL process will resume processing with a return code indicating that the request for access was cancelled. Specify the unique identifier of the resource request to indicate which request is to be cancelled.

NRD={ *domainid/n* | *domainid/ | *n* | *n*H | *n*M | *n*D | *n*N }**

Purge the nominated NRD message(s). The messages to purge can be identified in several ways:

domainid/n

Delete the NRD message(s) having the nominated identifier (shown in the format *domainid/id*). The identifier is the ID shown in a SHOW NRD display, or provided in the &ZDOMID or &ZMDOMID NCL system variables. This option is similar to the &NRDDEL NCL verb.

domainid/*

Delete all NRD messages that originated from the nominated domain (NMDID JCL parm or first 4 characters of the primary ACB name).

***n* or *n*H**

Delete all NRD messages that are older than *n* hours (in the range 1 to 999999).

***n*M**

Delete all NRD messages older than *n* minutes.

***n*D**

Delete all NRD messages older than *n* days.

***n*N**

Delete all NRD messages produced by the NCL process with NCLID *n*.

APPLSTAT=*applname*

Terminate INQUIRE APPSTAT processing for the specified application. The global variable which represents the status of the application will be deleted. Any MAI or EASINET functions which are suspended waiting for an outstanding INQUIRE will be notified of the failure of the INQUIRE to complete.

Examples:

```
PURGE TIMER=OWN
PURGE T=5 T=35 T=2
PURGE TIMER=ALL
PURGE LOCK=4
PURGE NRD=1H
PURGE APPLSTAT=CICS
```

Notes: The TIMER operand can be abbreviated to T and multiple entries made in the one command.

Use the SHOW LOCKS command to identify &LOCK occurrences that are waiting to be satisfied. The display will indicate the unique identifier of each waiting request.

A PURGE NRD=*time* command issued from a timer command. For example:

```
EVERY .15 PURGE NRD=1H
```

can keep the NRD retention list clean.

See Also: The AT, EVERY, SHOW NRD, SHOW LOCKS, and SHOW TIMER command descriptions.

The SYSPARMS APPLSTAT command in the *Management Services Administrator Guide*.

RECONN

Function: Connects the current terminal session with a previously disconnected processing region.

```
RECONN      [ ID=nnnnn ]
            [ REGION={ CANCEL | DISC } ]
(0)
```

Use: This command can be used to connect the terminal session with a previously disconnected processing region.

Operands:

ID=*nnnnn*

Specifies a nominated disconnect region that is to be reconnected to. The valid range is 1 to 65,535. If this operand is omitted then the oldest disconnected region for the user is selected.

REGION={ CANCEL | DISC }

Specifies the action to be taken for the current region. CANCEL terminates the requesting region and is the default. DISC performs a disconnection of the requesting processing region.

Examples:

`RECONN`

reconnect to any available region

`RECONN ID=834`

reconnect to disconnect region 834

`RECONN ID=834 REGION=DISC`

reconnect and leave the current region disconnected

Notes: Disconnected regions are assigned a numeric reconnection identifier. This identifier is displayed by the SHOW USERS command output and can be supplied on the ID operand to select a specific target for reconnection.

A reconnect region can be automatically cancelled after an installation specified wait time.

See Also: DISCONN, CANCEL, and SHOW USERS command descriptions.

REPALIAS

Function: Replaces an alias name entry used by the Alias Name Translation Facility of NEWS.

```
REPALIAS      NAME=alias
              NET=aliasnet
              RNAME=realname
              RNET=realnet
            [ RCDRM=cdrm ]
            [ TYPE={ LU | COS | MODE } ]
(4)
```

Use: This command is used to replace an existing alias name definition in NEWS which is used by the Alias Name Translation Facility.

Operands:

NAME=*alias*
The required 1- to 8-character *alias* or *origin* name of the type specified by the TYPE operand, which together with the network name provided by the NET operand uniquely identifies the definition being replaced.

NET=*aliasnet*
The required 1- to 8-character network identifier, which qualifies the alias name and identifies the definition being replaced.

RNAME=*realname*
The required 1- to 8-character name of the type specified by the TYPE operand which defines the *real* or *target* name of the resource as it is known in the target network (specified by the RNET operand). This value replaces the existing *realname* entry for this definition.

RNET=*realnet*
The required 1- to 8-character network identifier, known to VTAM, in which the resource name specified by the RNAME operand can be used. This value replaces the existing *realnet* entry for this definition.

RCDRM=*cdrm*
Can specify the 1- to 8-character name of the owning CDRM (Cross Domain Resource Manager) for TYPE=LU definitions only. If the *cdrm* name is specified, then this must be the LU's owning CDRM, or session establishment will fail.

TYPE=LU
Indicates that the *alias* name being replaced is an LU name. This is the default.

TYPE=COS
Indicates that the *alias* name being replaced is a COS name.

TYPE=MODE
Indicates that the *alias* name being replaced is a LOGMODE name.

Examples:

```
REPALIAS NAME=IMSTEST NET=NET* RNAME=IMS01 RNET=A1NET
REPALIAS NAME=BATCH NET=CTY3 RNAME=PRTY0 RNET=CTY2 TYPE=MODE
```

Notes:

See the description of the DEFALIAS command for full details on coding all operands and any restrictions that apply.

See Also:

The DEFALIAS, DELALIAS, and SHOW DEFALIAS command descriptions.

The *Unicenter NetMaster Network Management for SNA Implementation and Administration Guide*.

REPCLASS

Function: Replaces an existing SAW, RTM, Session or Resource class definition defined with a previous DEFCLASS command.

```
REPCLASS     SESSION=sessclass
            { [ RTMCLASS=rtmclass ] [ SAWCLASS=sawclass ] }
            { [ COS=cosname ] [ ER=er ] [ PRI=name ]
                [ SEC=name ] [ TP=tp ] [ VR=vr ] }
            [ SOURCE={ ALL | LOCAL | REMOTE } ]
            [ SSCP=sscpname ]
            [ SCLASS={ SD | XD | XN } ]
            [ STYPE={ LL | SL | SP | SS | MAI } ]

REPCLASS     SAW=sawclass
            [ ACCT={ YES | NO } ]
            [ EVENT={ YES | NO | ERROR } ]
            [ KEEP={ YES | NO | LOCAL } ]
            [ LOG={ SUMMARY | DATA | ERROR | ALL | NO } ]
            [ TRACE=(i,f) ]

REPCLASS     RTM=rtmclass
             BOUNDS=(value1,value2 ...  value4) OBJPC=objpc
             OBJTIME=objtime
                [ RTMDEF={ FIRST | CDEB | KEYBD | LAST } ]

REPCLASS     RESOURCE=rscclass
            [ RTMCLASS=rtmclass ]
            [ STATS= NO | STATS= YES [ LIMIT=nnn ] ]
            { [ LINK=name ] [ PU=name ] [ LU=name ] }
(4)
```

Use: This command is used to replace an existing SAW, RTM, Session or Resource class definition that has been defined using a previous DEFCLASS command.

Operands: **SESSION=*sessclass***

Identifies this class as a session class and provides the unique *sessclass* name of up to 8 characters which identifies this class. Use this name in the REPCLASS and DELCLASS commands to replace or delete this session class respectively.

The DEFCLASS operands specific to the definition of a SESSION class are:

RTMCLASS=*rtmclass*

Specifies the *rtmclass* name which is to be used by sessions that are associated with this session class. If no RTM class is found for a session then no RTM data is collected for that session.

SAWCLASS=*sawclass*

Specifies the *sawclass* name from which sessions falling within the session class being defined are to take their SAW class attributes. If not specified then the system SAW class defaults will apply to sessions in this session class.

COS=*cosname*

Optionally specifies the *cosname* of sessions which can be considered as within this session class. Generic names are not supported.

ER=*er*

Optionally provides the Explicit Route number *er* (0-7) that sessions must have to be considered in this session class.

PRI=*name*

Optionally provides the full or partial name of the primary resources considered to be in this session class. A generic name can include an asterisk (*) as a wild character in any position in the name. This represents a single wild character only. The last character can be specified as a trailing arrow (>) which indicates all trailing character positions are wild. Alternatively the last character can be specified as a trailing hyphen (-) which has the same meaning as > but in addition only those characters in the name up to the - are used when logging the session to the database, the rest of the name considered padded with blanks. (This allows, for example, by specifying PRI=TSO- all sessions between a particular terminal and primary partners such as TSO0001, TSO0014, TSO0055 and so on, to be logged in the database as different session incidences of the same session name pairs).

SEC=*name*

Optionally provides the full or generic name of the secondary session partner for sessions within this session class. Generic names as described for primary names above are allowed.

TP=*tp*

Optionally provides the Transmission Priority number *tp* (0-2) that sessions must have to be considered in this session class.

VR=*vr*

Optionally provides the Virtual Route number *vr* (0-7) that sessions must have to be considered in this session class.

SOURCE={ <u>ALL</u> | LOCAL | REMOTE }

Selects sessions on the basis of their origin. If SOURCE=LOCAL is specified then only sessions whose data is derived from the local VTAM are selected. If SOURCE=REMOTE then only sessions whose data is derived from other NTS systems are selected (that is, data is collected from NTS systems that are not in session with the local VTAM). ALL is the default and selects both types of session.

SSCP=*sscp name*

Selects sessions derived from the domain defined by the SSCP named.

SCLASS={ SD | XD | XN }

SD specifies sessions in which both session partners are in the same domain, XD specifies cross domain sessions and XN specifies cross network sessions. A session must match this parameter to be associated with this session class. If this parameter is not specified, the class of session is not used to match the session class.

STYPE={ LL | SL | SP | SS | MAI }

LL specifies LU-LU sessions, SL specifies SSCP-LU sessions, SP specifies SSCP-PU sessions and SS specifies SSCP-SSCP sessions. STYPE=MAI selects MAI sessions. A session must match this parameter to be associated with this session class. If this parameter is not specified, the type of session is not used to match the session class.

SAW=*sawclass*

Identifies a SAW class and provides the unique *sawclass* name of up to 8 characters which identifies the class. Use this name in the REPCLASS and DELCLASS commands to replace or delete this SAW class respectively. Use this name also on the SAWCLASS operand of the DEFCLASS command when defining a *Session class* to provide the name of the SAW class applicable to the particular session class (see below).

The DEFCLASS operands specific to the definition of a SAW class are:

ACCT={ YES | <u>NO</u> }

Indicates whether or not accounting data is to be accumulated for sessions using this SAW class. The collection of accounting data firstly depends upon the SYSPARM NTSACCT operand value. If SYSPARM NTSACCT=NO is in effect then no accounting information is collected regardless of the ACCT operand value. If SYSPARM NTSACCT=ALL is operative then accounting data is collected for all sessions again regardless of this ACCT operand. However, if the SYSPARM NTSACCT=SELECTIVE (default) option is in effect then the ACCT operand of the SAW class definition determines whether or not accounting data is to be collected for those sessions using this SAW class.

EVENT={ YES | <u>NO</u> | ERROR }

Indicates whether events are to be generated for sessions matching this class. The events that can be generated are as follows:

- Session start ($$NTS.SESSION.START)
- Session end ($$NTS.SESSION.END)
- Session failures ($$NTS.SESSION.FAIL)
- RTM objective not met ($$NTS.RTM.OBJ.EXCEEDED)

When EVENT=YES is specified, all these events are generated and sent to profiled users. When EVENT=NO is specified or defaulted, none of these events are generated. When EVENT=ERROR is specified, only the session failure event is generated.

Note

The NTS SYSPARMS NTSEVENT operand must be set to YES to cause NTS events to be generated

KEEP={ YES | NO | LOCAL }

Specifies whether or not session (or SAW) data in this class is to be retained by NTS. If KEEP=NO is specified then NTS discards session awareness data for any sessions mapped into this SAW class. No information on these sessions is subsequently available to the user, and no other NTS information can be collected for such sessions. Hence when KEEP=NO is specified all other SAW class operands are meaningless. If KEEP=LOCAL is specified then data relating to any session in this class will not be forwarded (via an ISR link) to any other NTS.

LOG={ SUMMARY | DATA | ERROR | ALL | NO }

Determines what logging options are to be applied to data in this SAW class. The default LOG=SUMMARY option means all data except trace is logged at session end if the session ends normally, however all data including trace data is logged if the session ends in error. The LOG=DATA option means that if there is any session data (such as trace data, accounting data and so on) available for the session then all data is logged otherwise no logging takes place. The LOG=ERROR option means that all data will be logged but only if the session ends due to an error, otherwise no logging takes place. The LOG=ALL option indicates that all data is to be unconditionally logged, while LOG=NO indicates no data should be logged at all.

TRACE=(*i,f*)

Provides the depth *i* of the initial trace queue, and the depth *f* of the final trace queue for this SAW class. Both *i* and *f* can be in the range 0 to 999, the default values being 4 and 20 respectively. When session establishment trace data arrives for a session it will be placed on the initial queue until the queue depth is reached. Other trace data is placed on the final queue and when the queue limit is reached the oldest data is overwritten with the most recent. Trace data continues to wrap on the final queue until the session ends or tracing is terminated.

RTM=*rtmclass*

Identifies an RTM class and provides the unique *rtmclass* name of up to 8 characters which identifies this class. Use this name on the REPCLASS and DELCLASS commands to replace or delete this RTM class respectively. Use this name also on the RTMCLASS operand of the DEFCLASS command when defining a session class (see below) to associate sessions in the session class with this RTM class.

The DEFCLASS operands specific to the definition of an RTM class are as follows:

BOUNDS=(*value1,value2...value4)*

Specifies up to 4 boundary values which are to be set in the control unit and used to accumulate RTM data for this session. Each of these values must be a time specified in the same manner as for *objtime* above. Each subsequent value must be greater than the previous (that is, *value1* < *value2* < ... *value4*). One of the values specified must be the same as the *objtime* specified below.

OBJPC=*objpc*

Defines the objective percentage for this RTM class as a value in the range 1 to 100. If the percentage of total transactions which complete within the *objtime* specified equals or exceeds the *objpc* value specified then the performance objectives for this RTM class are considered to be met.

OBJTIME=*objtime*

Defines the objective response time for the session. This is a time in the range 0.1 seconds to 30 minutes and can be specified as *mm:ss* (minutes and seconds), *mm:ss.t* (minutes, seconds and tenths of a second), *ss* (seconds), or *ss.t* (seconds and tenths of a second). This value must correspond to one of the boundary values (see BOUNDS operand below) or the DEFCLASS command will be rejected.

RTMDEF={ FIRST | KEYBD | CDEB | LAST }

Defines the RTM definition criteria used. If FIRST is specified (or defaulted) the response time is measured until the first character of the host datastream is received. For KEYBD it is measured until the keyboard is unlocked, for CDEB it is measured until an SNA Change Direction or End Bracket order is received, while for LAST it is measured until the last character of the host datastream is received. See the relevant component description manual of devices which support RTM (such as 3274 and 3174) for complete details on the operation and collection of RTM data in the control unit.

RESOURCE=*rscclass*

Identifies this class as a resource class and provides the unique *rscclass* name of up to 8 characters which identifies this class. Use this name in the REPCLASS and DELCLASS commands to replace or delete this resource class respectively.

The DEFCLASS operands specific to the definition of a Resource class are:

RTMCLASS=*rtmclass*

Optionally specifies the *rtmclass* name which is to be used for resources that are associated with this resource class. The format of RTM responses received will be compared with the format defined in the RTM class definition. Statistics will be kept for matching responses. This parameter is only relevant if statistics are being collected for resources associated with this class. If this parameter is not specified, no RTM statistics will be collected for resources associated with this resource class.

STATS={YES | NO }

Indicates whether or not statistics are to be collected for resources associated with this class. The collection of resource statistics firstly depends on the SYSPARMS NTSRSTAT operand value. If SYSPARMS NTSRSTAT=NO is in effect, no resource statistics are collected, irrespective of the value of this operand. Resource accounting statistics are derived from session accounting data. If no session accounting information is being collected for sessions in which a resource participates, no statistics will be available for the resource. It is strongly recommended that SYSPARM NTSACCT=ALL be in effect if resource statistics collection is required.

LIMIT=*nnn*

nnn is minutes, ranging from 0 to 255. This parameter specifies the number of intervals to occur before the statistics for the oldest interval are overwritten. This parameter is only relevant if STATS=YES is specified. The operand for this parameter will override any value specified for the NTSRLIM operand of the SYSPARMS command.

Note

High settings of this operand can consume large amounts of storage.

LINK=*name*

Optionally provides the full or partial name that must be used by links to be considered associated with this class. See the *Generic Name* description below for valid formats.

PU=*name*

Optionally provides the full or partial name that must be used by PUs to be considered associated with this class. See the *Generic Names* description below for valid formats.

LU=*name*

Optionally provides the full or partial name that must be used by LUs to be considered associated with this class. See the *Generic Names* description below for valid formats.

Note

Generic Names

The generic format uses an asterisk (*) as a single *wild* character and a trailing arrow (>) as an indicator that all trailing characters are *wild*. For LU operand values a trailing hyphen (-) can be used. This is treated in the same way as the arrow with the additional function of collecting all the statistics under the single name. This name is made up of all the characters of the LU operand value excluding the hyphen.

Examples:

```
REPCLASS SESSION=TSO  RTMCLASS=TSORTM1 SESSION=TSO>
REPCLASS RTMCICS OBJTIME=1 OBJPC=80 BOUNDS=(0.5,1,3,10)
REPCLASS RESOURCE=ALU LINK=L080BF PU=PUA>> LU=LUA>
          RTMCLASS=TSORTM2 STATS=YES LIMIT=30
```

Notes:

All non-default values must be specified on the REPCLASS command even if there is no change from the existing class definition. Sessions not mapped by a Session class receive the system SAW class default values (see the underscored SAW class operand values above).

If any session class selection operands are omitted all values of the type omitted are considered to be valid for the class. For example if no PRI= operand is supplied all primary names are considered valid (that is, omitting say the PRI operand is equivalent to specifying PRI=>, omitting the ER operand is equivalent to specifying ER=* and so on). Where selection options are provided sessions must match all such options to be considered in the session class.

The order of the operands as shown above for a Session class definition is the order of precedence for class selection where a session could otherwise exist in more than one class. Further, a more specific name will be matched before a less specific one (for example, SEC=LU25 will be found before SEC=LU**).

If a session falls within a particular Session class and either the SAWCLASS or RTMCLASS is not defined for that Session class, the search will continue for a session class matching the session attributes to determine the missing SAW or RTM class. When both the SAWCLASS and RTMCLASS are known, or all session classes exhausted, the search ends and default SAW attributes assigned if necessary.

At least one of the LINK, PU or LU parameters must be specified in a Resource class definition. If STATS=YES is specified and no LIMIT parameter is present, the number of intervals is taken from the NTSRSINT operand of the SYSPARMS command. If no RTM Class is specified, no RTM statistics will be collected for any resource associated with the resource class.

See Also: The DEFCLASS and DELCLASS command descriptions.

Function: Replaces or defines a DLT definition.

```
REPLINK      TYPE=INMC
           [ LUNAME=luname LINK=link |
                 MASK=mask PREFIX=prefix |
                 LUNAME=luname PREFIX=prefix |
           [ AM=amname ]
           [ { COLOR | COLOUR }=color ]
           [ { HLIGHT | HLITE }=hlite ]
           [ RT=nnnnn ]
           [ MSGID=id ]
           [ DEFNAME=inmcefname ]

REPLINK      TYPE=NVC
             LUNAME=luname  LINK=link
           [ SSID=NetView_ssid ]
           [ { COLOR | COLOUR }=color ]
           [ { HLIGHT | HLITE }=hlite ]
           [ MSGID=id ]

REPLINK      TYPE=APPC
           { LUNAME=luname LINK=link |
                 LUMASK=lumask PREFIX=pref |
                 LUNAME=luname PREFIX=pref |
           [ AM=amname ]
           [ QUAL=qualifier ]
           [ PASSWORD=password ]
           [ OPSET=opset ]
           [ { COLOR | COLOUR }=color ]
           [ { HLIGHT | HLITE }=hlite ]
           [ MON={ YES | NO } ]
           [ USERENV={ GLOBAL | LINK } ]
           [ TRACE={ YES | NO } ]
(3)
```

Use: INMC Dynamic links are created on request for a remote system provided the request matches the criteria set up in a dynamic link definition on this system. Use the REPLINK command to define or build a dynamic link.

See the chapter about INMC and APPC in the *Management Services Administrator Guide* for details of dynamic link concepts.

APPC Dynamic Link Table, or DLT, entries are used to start APPC links in response to an &APPC ALLOCATE verb or a LINK START command. Use the REPLINK command and the related DEFLINK and DELLINK commands to maintain the DLT.

See the chapters on APPC in your *Management Services Administrator Guide* for further details.

Operands: **TYPE=INMC**

Indicates that this DEFLINK command defines an INMC Dynamic Link Table entry. TYPE=INMC is the default, so this operand is optional when defining INMC links.

INMC operands are:

LUNAME=*luname*

Provides the LU name that corresponds to the link name described below.

LINK=*link*

Provides a specific LINK name, and requires that the LU name parameter is used to provide the corresponding LU name of the remote system. When the LINK and LUNAME operands are used, the MASK, LUMASK and PREFIX operands are not allowed. If a LINK START command is issued for this link name and type, then the DEFLINK definition can be used to complete the link details.

MASK=*mask*

This operand is mandatory. It defines a mask against which any user data on an incoming BIND (from another SOLVE system) or logon request (from an NMPC system) will be tested. The mask value can be a generic string with asterisks as wild characters. If the user data matches the mask, a dynamic link will be built. The link name assigned is the actual LU name, prefixed by the 4-character PREFIX value. The use of MASK and PREFIX is mutually exclusive with the use of LINK and LUNAME.

PREFIX=*prefix*

This operand is required when MASK is used. When a dynamic link is built as a result of this definition, its linkname will consist of this prefix concatenated in front of the VTAM nodename for the remote system. The prefix must be four (4) characters long.

AM=*amname*

Specifies the access method to be used for the link. Can be VTAM or EPS (End Point Services). If this is omitted, the primary access method is used.

{ COLOR | COLOUR }=*color*
{ HLIGHT | HLITE }=*highlight*

Provides a means of identifying messages originating from this set of dynamic links through extended highlighting or color displays. Color can be:

```
RED, YELLOW, WHITE, BLUE, GREEN, TURQUOISE, or PINK
```

Highlighting can be:

```
REVERSE, BLINK, USCORE
```

Both FTS and ROF messages received from these links are highlighted as requested for terminals supporting that type of data stream.

RT=*nnnnn*
Retry interval. The default is 60 seconds.

MSGID=*id*
A 1- to 4-character string that is to be used as a prefix to any message received by a ROF user across any of these dynamic links. The default MSGID is the prefix value. Individual OCS users that use ROF to receive messages from remote systems might choose their own remote system prefix via the SIGNON command, overriding the default value set on the DEFLINK command.

DEFNAME=*inmcefname*
Specifies the name of an INMC multipath link definition. This is the only way that a multi-path link can be established when using a REPLINK.

If specified, when an INMC link is dynamically started on this system as a result of matching this REPLINK, then the named INMC multipath link definition record is used as a model for the session definitions. The domain name in the definition is ignored.

A LINK START command that matches a REPLINK definition also uses the DEFNAME, unless overridden on the LINK START.

TYPE=NVC
Indicates that this DEFLINK command defines an NVC Dynamic Link Table entry. The default is TYPE=INMC so this operand is mandatory when defining NVC links.

NVC operands are:

LUNAME=*luname*
Provides the LU name that corresponds to the link name described below.

LINK=*link*
Provides a specific LINK name, and requires that the LU name parameter is used to provide the corresponding LU name of the remote system. If a LINK START command is issued for this link name and type, then the DEFLINK definition can be used to complete the link details.

SSID=*NetView-ssid*
A 4-character string that specifies the first 4 characters of the NetView jobname. If this operand is omitted, it must be specified on the LINK START command.

{ COLOR | COLOUR }=*color*
{ HLIGHT | HLITE }=*highlight*
Provides a means of identifying messages originating from this set of dynamic links through extended highlighting or color displays. Color can be:

```
RED, YELLOW, WHITE, BLUE, GREEN, TURQUOISE, or PINK
```

Highlighting can be:

```
REVERSE, BLINK, USCORE
```

Both FTS and ROF messages received from these links are highlighted as requested for terminals supporting that type of data stream.

MSGID=*id*

A 1- to 4-character string that is to be used as a prefix to any message received by a ROF user across any of these dynamic links. The default MSGID is the prefix value. Individual OCS users that use ROF to receive messages from remote systems can choose their own remote system prefix via the SIGNON command, overriding the default value set on the DEFLINK command.

TYPE=APPC

Indicates that this DEFLINK command defines an APPC Dynamic Link Table entry, and pertains to LU6.2 session connections only. The default is TYPE=INMC so this operand is required for APPC DLT definitions.

APPC operands are:

LUNAME=*luname*

Provides a specific network LU name (of up to 8 characters) that, when connected to this system by an LU6.2 session, will be known by the *link* name as described below. By using the LINK and LUNAME parameters a fixed link name can be associated with a target LU, such as another LU6.2 capable application system. When an LU6.2 session starts between this system and the target *luname*, by whatever means, this DEFLINK entry is used to complete the link details.

LINK=*link*

Provides a specific *link* name (of up to 12 characters), and can only be used when the LUNAME parameter is used to supply a specific *luname*. Both the *link* and *luname* supplied must be unique amongst all APPC DLT entries. The use of the LINK parameter is mutually exclusive with the use of the PREFIX parameter.

LUMASK=*lumask*

Provides an LU name mask that must be unique amongst APPC DLT definitions. The *lumask* can use an asterisk (*) to denote a wild character in any position, or a trailing * to denote the remainder of the name as wild. When matching an LU name against the *lumask* a more specific mask is selected over a more generic one, with leftmost characters having precedence over those occurring to the right (for example, the name N12P3L15 matches the masks, N*, N****L, and ***P3L15, but selects the second as its leftmost character takes precedence over the third mask, and it is a more specific match than the first mask). This parameter is mutually exclusive with the LUNAME parameter.

PREFIX=*pref*

Provides a 4-character string that, when prefixed to the actual LU name, creates a link name, and is used when a DEFLINK entry is matched using the *lumask* described above. This parameter is required when LUMASK is used, but is mutually exclusive with the LINK parameter.

AM=*amname*

Specifies the access method to be used for the link. Can be VTAM, EPS (End Point Services), or INMC. If omitted, the primary access method is used.

QUAL=*qualifier*

Provides the name of qualifier with up to 4 characters to be used for this destination to qualify transaction identifiers (see DEFTRANS). This operand is optional, and if omitted then transactions to this remote LU are unqualified.

PASSWORD=*password*

Provides a 16-hexadecimal character string, representing 8 bytes of data, to be used as a password for the link. A password is required only if LU-LU session verification is required, and can be supplied in the DEFLINK definition, or on the LINK START command.

OPSET=*opset*

Nominates the name of the OSCT entry to be used to determine the communication options supported by this remote LU.

{ COLOR | COLOUR }=*color*
{ HLIGHT | HLITE }=*highlight*

Provides a means of identifying messages originating from this set of dynamic links through extended highlighting or color displays. Color can be:

```
RED, YELLOW, WHITE, BLUE, GREEN, TURQUOISE, or PINK
```

Highlighting can be:

```
REVERSE, BLINK, USCORE
```

Both FTS and ROF messages received from these links are highlighted as requested for terminals supporting that type of data stream.

MON={ YES | <u>NO</u> }

Specifies whether or not link messages for those connected systems using this definition are to be displayed as monitor messages. If MON=NO is specified, or defaulted, then link messages are directed to the activity log only. If MON=YES is specified then link messages are sent to all OCS monitor users.

USERENV={ <u>GLOBAL</u> | LINK }

The USERENV operand is used to indicate whether or not an APPC user region created to service an attach request for a given user ID and received from the remote network node defined by the DEFLINK definition, will also service subsequent requests (for the same user ID) received from different remote network nodes.

GLOBAL indicates that the APPC region built will service attach requests from different remote nodes. LINK indicates that the APPC region will only service attach requests received from the remote node defined by this DEFLINK definition.

TRACE={ YES | <u>NO</u> }

Specifies whether LU6.2 session level tracing will be performed for all sessions in this link.

REPLINK

Examples:

```
REPLINK MASK=IMPC PREFIX=NMPC
REPLINK MASK=N*P* PREFIX=DYNL COLOR=RED HLITE=REVERSE
REPLINK MASK=* PREFIX=LINK MSG=INMC
REPLINK TYPE=APPC LU=NMA LINK=NMPROD OPSET=NM MON=YES
REPLINK TYPE=APPC LU=CICS1 LINK=TESTCICSREPLINK TYPE=APPC
        LUMASK=OS2* PREFIX=WSTN OPSET=WSOPSET
REPLINK TYPE=APPC LU=NM1 PREFIX=NETM OPSET=NM
```

Notes:

REPLINK commands will usually be included in the system initialization procedures.

Regardless of the REPLINK commands in use, a remote link request does not result in a dynamic link being established if the SYSPARMS DYNLMAX limit has been reached. The default for SYSPARMS DYNLMAX is zero.

When an LU6.2 session setup request for a remote LU is first processed the DLT is searched to locate an entry that will be used to initialize the APPC link. Those entries defined with specific LINK and LUNAME parameters are examined first. If a DLT entry with a matching LU name is found the APPC link is initialized using the corresponding LINK name.

If no specific entry is found, the remaining entries in the DLT are examined to locate the most appropriate LUMASK match, as described for the LUMASK parameter above. If found APPC link initialization proceeds using a link name built as described for the PREFIX parameter above. Otherwise the session setup request is denied with a *permission rejected* sense code.

See Also:

The DEFLINK (INMC), LINK (INMC), SHOW DEFLINK, and SHOW LINK command descriptions.

REPLOGON

Function: Replaces an existing application logon entry as defined by a previous DEFLOGON statement, or adding a new one.

```
REPLOGON    { INPUT=text | IN=text }
            [ IGE | IEQ ]
            [ APPL={ vtamappl | * } ]
            [ DATA={ userdata | INPUT | INPUT1 | NOEASINET } ]
            [ LOGMODE=table entry ]
            [ MID=MAI-id ]
            [ MSHR={ YES | NO } ]
            [ APPSTAT={ YES | NO } ]
            [ MRESP={ AFTER | BEFORE } ]
            [ MPREF=prefix ]
            [ MTLIMIT=minutes ]
            [ MTACTION={ CANCEL | CC | SCRIPT | EVENT } ]
            [ NOTIFY={ YES | NO } ]
            [ REFRESH=key ]
            [ DESC=session description ]
(4)
```

Use: This command is normally reserved for Systems Programmer use and is therefore described fully in the *Management Services Administrator Guide*.

If only the INPUT operand is specified the DEFLOGON entry that matches the text in the input operand will be fetched and displayed in the command line for modification. The existing entry will be replaced by the updated entry when the enter key is pressed.

Operands:

INPUT=*text* | IN=*text*

The input string entered by the operator, or the leading characters of the string to be entered.

IGE | IEQ

Identifies the type of match to be made against the input string. IGE (the default) specifies that a logon string will match on leading characters of the logon path defined using the INPUT operand (see above). Alternatively, IEQ can be specified which indicates that an exact match must be made.

APPL={ *vtamappl* | * }

VTAM APPL name of target application or * for Management Services' own APPL. If omitted an application name matching the INPUT text will be assumed.

DATA={ *userdata* | INPUT | INPUT1 | NOEASINET }

The user data to be passed to the target application. Specify INPUT if text as entered by the user is to be passed, or INPUT1 if text as entered minus the first word is to be passed. Specify NOEASINET if the target application is another SOLVE system and its EASINET facility is to be bypassed. If the *userdata* format is used and the data contains blanks, the string must be enclosed in quotes (").

REPLOGON

LOGMODE=*table entry*
VTAM log mode table entry to be used when the session is established.

MID=*MAI-id*
Default MAI session ID if one is not supplied during creation of an MAI-FS session. If not supplied the APPL name is used as the default.

MSHR={ YES | NO }
Determines if MAI sessions created using this entry will use ACB sharing.

APPSTAT={ YES | NO }
Determines whether the application status should be queried before establishing a session.

MRESP={ AFTER | BEFORE }
Specifies whether MAI-FS sessions using this definition are to respond to the application before or after the data has been sent to the terminal. The AFTER setting (default) provides synchronous session control, with the terminal's responses being echoed to the application.

The BEFORE setting indicates that MAI-FS is to always respond positively upon receipt of data. It also provides an asychronous mode of operation.

MPREF=*prefix*
Specifies an ACB prefix for MAI to use as a logical terminal pool. If not specified, it defaults to the value of the SYSPARMS MAIPREF operand.

MTLIMIT=*minutes*
Specifies a timeout limit that applies to MAI sessions established using this DEFLOGON entry.

MTACTION={ CANCEL | CC | SCRIPT | EVENT }
Specifies the action to be taken when an MAI session is inactive for the time specified by MTLIMIT.

CANCEL will cancel a session, CC will cause a conditional cancel, SCRIPT will cause a session script to be initiated, and EVENT (the default) will cause an event to be issued.

NOTIFY={ YES | NO }
Specifies whether or not EASINET should use VTAM's Third Party Notify capability when establishing a session.

REFRESH=*key*
Supplies the name of the application's refresh key. MAI-FS will include this key as part of the *lost screen image* message (N54201).

DESC=*session description*

Enter a text description of the application identified by this DEFLOGON entry. If specified, this must be the last operand. It can consist of any characters (including blanks) without the need for enclosing the data in quotes. If your system is configured with the MAI feature, this description will be displayed beside the MAI menu entry for this application.

If present, the DESC= operand must be the last specified.

Examples:

```
REPLOGON INPUT=IMS
REPLOGON INPUT=OP IGE APPL=TSO DATA='OPTSO ACCT(123,245,678)'
REPLOGON INPUT=CICS APPL=CICSPROD MSHR=YES  MID=CICS DESC=
Production CICS
```

Notes:

The REPLOGON command will add an entry if it does not already exist.

An easy way to add a new DEFLOGON entry that is similar to an existing entry is to enter the REPLOGON command for the existing similar entry. This entry will then be displayed in the command line.

See Also:

The DEFLOGON, DELLOGON, and SHOW DEFLOGON command descriptions.

REPLY

Function: Issues a VTAM REPLY command to respond to an outstanding prompt from VTAM.

REPLY (1)	*lnn,reply text*

Use: In some critical circumstances VTAM will issue messages that require responses from the OCS operator. These are allocated a unique identification that must be specified in the REPLY command. This identification is in the format *lnn*. Unsolicited messages that require a reply will be displayed only on those terminals that have PPO authority. The messages will be non-roll delete and will therefore remain on the screen until a reply is made. The operator must enter the REPLY command with the corresponding reply identification as displayed in the message, and the required reply text.

Operands: ***lnn,reply text***

The reply identification of the outstanding message that is to be responded to, and the text that forms the reply. *l* can be either the letter P or S and *nn* is a two-digit number in the range 0 to 99.

Examples:

```
REPLY P00,DUMP
REPLY S01,YES
```

Notes: The letter P or S that forms part of the reply identification number is used to control the routing of the reply to VTAM. Any REPLY command must use the exact form of the reply identification as displayed with the message that requires the reply.

The SHOW REPLY command can be used to display the most recent messages from VTAM that require a reply. This display includes the identification number associated with the reply.

See Also: The SHOW REPLY command description.

REPMODE

Function: Replaces or defines an APPC Mode Control Table (MCT) entry.

```
REPMODE      MODE=mode
             [ MODENAME=modename ]
             [ LOGMODE=logmode ]
             [ SESSLIM=nn ]
             [ MINWIN=nn ]
             [ MINLOSE=nn ]
(4)
```

Use: The REPMODE, DEFMODE and DELMODE commands are used to maintain the MCT. These commands are normally reserved for Systems Programmer use and are further described in the *Management Services Administrator Guide*.

Operands:

MODE=*mode*

Identifies the unique name (of up to 8 characters) for this mode control table entry. This name can be used to relate an OSCT definition to this MCT definition by including the *mode* name in the list of the parameters following the MODE operand for the DEFOPSET command. This operand is required.

MODENAME=*modename*

Provides the actual LU6.2 *modename* for this mode definition. This *modename* can be used to relate a TCT definition to this MCT definition by specifying the *modename* on the MODENAME operand for the DEFTRANS command. This *modename* can also be specified on the MODENAME parameter of an &APPC ALLOCATE verb if necessary to override any system defaults. If this operand is omitted, then the *modenam*e defaults to the *mode* name specified on the MODE parameter.

LOGMODE=*logmode*

Provides the actual logmode name to be used on session initiation requests for this mode set definition. This can be used to assign certain network parameters such as RU sizes, and Class of Service (COS) values. If this operand is omitted, then no VTAM *logmode* is used for the session initiation request.

SESSLIM=*nn*

Specifies the session limit (up to 99) for this mode name definition (corresponding to the LU6.2 architected LU_MODE_SESSION_LIMIT parameter of the CNOS verbs) and is applicable where parallel sessions are supported. If this operand is omitted, then a limit of 1 is assumed.

MINWIN=*nn*

Specifies the minimum number of contention winner sessions for this mode name definition (corresponding to the LU6.2 architected MIN_CONWINNERS_SOURCE parameter of the CNOS verbs). If this operand is omitted, then a minimum of 1 is assumed.

MINLOSE=*nn*

Specifies the minimum number of contention loser sessions for this mode name definition (corresponding to the LU6.2 architected MIN_CONWINNERS_TARGET parameter of the CNOS verbs). If this operand is omitted, then a minimum of 0 is assumed.

Examples:

```
REPMODE MODE=WSMODE LOGMODE=LU62MODE SESSL=5 MINW=2 MINL=2
REPMODE MODE=NMMODE LOGMODE=LU62PARL SESSL=20 MINW=5 MINL=5
REPMODE MODE=NMSNGL LOGMODE=LU62SNGL
REPMODE MODE=LU62SNGL
```

Notes: The MCT is used by LU6.2 services.

See Also: The DEFMODE and SHOW DEFMODE command descriptions.

REPOPSET

Function: Replaces or defines an APPC Option Set Control Table (OSCT) entry.

```
REPOPSET     OPSET=opset
             MODE=(mode,...,mode)
           [ MAP={ NO | YES } ]
           [ LOG={ NO | YES } ]
           [ SECURITY={ NONE | USER | USERPSWD } ]
           [ PARSESS={ NO | YES } ]
           [ PV={YES|NO}]
(4)
```

Use: The REPOPSET, DEFOPSET, and DELOPSET commands are used to maintain the OSCT. These commands are normally reserved for Systems Programmer use and are further described in the *Management Services Administrator Guide*.

Operands: As per DEFOPSET. See the DEFOPSET command for all operand descriptions.

Examples:

```
REPOPSET  OPSET=NMOPSET MODE=NMMODE MAP=Y LOG=Y PARS=Y +
          SEC=USER
REPOPSET  OPSET=WSOPSET MODE=WSMODE MAP=YES
REPOPSET  OPSET=CICSOPST MODE=(CICSMODA,CICSMODB) +
          PARSESS=Y SEC=USER
```

Notes: The OSCT is used by LU6.2 services.

See Also: The DEFOPSET, DELOPSET, and SHOW DEFOPSET command descriptions.

REPTRANS

Function: Replaces or defines an APPC Transaction Control Table (TCT) entry.

```
REPTRANS      TRANSID=transid*
            [ QUAL=qualifier ]
            [ TPN=tpn* ]
            [ ENV={ CURRENT | DEPENDENT } |
              LUNAME=luname | LINK=linkname | DOMAIN= domain ]
            [ MODENAME=modename ]
            [ PROC=procname* [ SERVER= servername
                [ SCOPE={ REGION | USER | SYSTEM } ] ] ]
            [ SYNC={ NONE | CONFIRM } ]
            [ SECURITY={ NONE | SAME | USER | USERPSWD } ]
            [ CMDAUTH=nn ]
            [ CONVTYPE={ MAPPED | BASIC } ]
            [ SIGNON={ YES | NO } ]
(4)
```

Use: The REPTRANS, DEFTRANS and DELTRANS commands are used to maintain the TCT. These commands are normally reserved for Systems Programmer use and are further described in the *Management Services Administrator Guide*.

Operands: **TRANSID=*transid** (mandatory)**

This operand identifies the locally known transaction identifier (of up to 12 characters) and must be unique for the qualifier nominated by the QUAL operand on this definition. The *transid* has meaning for both allocation requests and attach requests (see Notes below for an explanation of these terms). The *transid* corresponds to the locally known name of the LU6.2 architected TPN parameter of the ALLOCATE verb.

When an NCL procedure issues an allocation request for a conversation, it must supply the *transid*. The appropriate TCT entry is then located using this information to complete the allocation request.

When an attach request is received from the remote node to start a conversation, the TCT entry is located by *tpn* but is locally known by *transid*.

Thus the *transid* specifies the locally known identifier only and is the transaction identifier returned by a query operation against the current conversation, regardless of whether it was allocated locally or remotely.

A generic value can be specified for *transid*. A transaction identifier with no exact match in the TCT will be compared to any defined generic entries. When a matching entry is found it is used as the entry for that transaction identifier. A generic *transid* is usually used in conjunction with a generic *tpn* and *procname*. See details of these operands for an explanation.

Note

Transaction IDs starting with a dollar sign ($) are reserved for future system use.

QUAL=*qualifier*

This operand provides the transaction qualifier (of up to 4 characters) for this transaction identifier. If this parameter is omitted, the *transid* being defined is unqualified. Use of a transaction qualifier allows the definition of multiple transactions with the same identifier, but which are distinguished by different qualifiers. This is most useful where a given transaction has different requirements according to the conversation destination. During an allocation request the *qualifier* is derived from the DLT entry for the transaction destination, and if it is set, the TCT scanned for a *transid* with that qualification. If not found, an unqualified *transid* is accepted.

TPN=*tpn**

Applies to both locally and remotely allocated conversations, and specifies the globally known Transaction Program Name (TPN), of up to 64 characters, unique for the qualification specified, that corresponds to the locally known identifier *transid*. When completing a local allocation request the *tpn*, not the *transid*, is passed to the remote end as the LU6.2 architected TPN parameter on the ALLOCATE request.

Conversely, when a remote attach request is received by the local system, the incoming TPN is matched against this *tpn* value to select the TCT entry to complete the remote allocation request. Hence for any conversation to successfully initiate, both the local and remote ends need only agree upon the *tpn* value for any particular transaction, while local transaction identifiers can be defined in any convenient manner. If this parameter is omitted it defaults to the *transid* value.

A generic value can be specified for *tpn*. A *tpn* name with no exact match in the TCT will be compared to any defined generic entries. When a matching entry is found it is used as the entry for that *tpn* name. A generic *tpn* name is usually used in conjunction with a generic *procname*.

ENV={ <u>CURRENT</u> | DEPENDENT }

This operand applies to locally allocated conversations only. Its presence indicates that, when no destination parameters are specified in the allocation request, the attached process will be started in the local SOLVE system, in the same region as the requesting process.

If ENV=CURRENT is specified, or defaulted, the attached process will be started as a peer of the requesting process. If ENV=DEPENDENT is specified, the attached process will be started as a dependent of the requesting process.

LUNAME=*luname*

Applies to locally allocated conversations only, and by its presence indicates the default target system to be used for this transaction. The LUNAME operand is mutually exclusive with the ENV, LINK and DOMAIN operands.

LINK=*linkname*

Applies to locally allocated conversations only, and by its presence indicates the default APPC link to be used for this transaction. The LINK operand is mutually exclusive with the ENV, LUNAME and DOMAIN operands.

DOMAIN=*domain*

This operand applies to locally allocated conversations only. Its presence indicates that if no destination parameters are specified in the allocation request, the default target system is an SOLVE/APPC system with a domain ID of *domain*. If the SOLVE/APPC system cannot be located by domain ID (for example, where an INMC link has not been activated to the system, and hence the domain ID is unknown), the request will fail.

MODENAME=*modename*

Applies to locally allocated conversations only, and nominates the default mode name to be used for the conversation. It corresponds to the LU6.2 architected MODE parameter of the ALLOCATE verb. Together with the actual destination *luname*, the *modename* forms a unique session selection criteria. Unless a session to the remote LU is active using this *modename*, then no current session can be used for the conversation. Depending upon allocation options, a new session can be opened to the remote LU using this *modename*, otherwise the allocation will fail. If omitted a default *modename* for the selected destination will apply.

PROC=*procname**

Applies to remotely allocated conversations only, and by its presence indicates that this TCT entry can be used to complete remote allocation requests. The nominated *procname* is the procedure name that is to service this transaction when an allocation request is received. If omitted, this TCT entry cannot be the subject of an attach request.

A generic value can be specified for *procname*. It consists of a prefix which is used, in conjunction with a generic *tpn*, to construct the name of the NCL procedure to be invoked. See the notes below for an explanation.

SERVER=*servername*

This operand is a 1 to 32 character name. It applies to remotely allocated conversations only.

It indicates that the NCL process that is to service this transaction, as indicated by the PROC operand, is to be registered as a server with the name *servername* In the case where *servername* is already a registered name, attach processing will attempt to connect the attach request to the active server.

SCOPE={ REGION |USER | <u>SYSTEM</u> }

This operand is used to provide the scope for the registration of the *servername*, and is valid only if the SERVER operand is specified. Valid scopes are REGION, USER and SYSTEM which indicate (respectively) region-wide, user-wide and SOLVE system-wide scoping.

SYNC={ NONE | <u>CONFIRM</u> }

Applies to locally allocated conversations only, and nominates the default conversation synchronization options. It corresponds to the LU6.2 architected SYNC-LEVEL parameter of the ALLOCATE verb. A value of SYNC=NONE means no synchronization is used, and the &APPC CONFIRM and CONFIRMED options cannot be used, nor can the &APPC PREPARE-TO-RECEIVE TYPE=CONFIRM and &APPC DEALLOCATE TYPE=CONFIRM options.

The default value of SYNC=CONFIRM indicates that both the &APPC CONFIRM and CONFIRMED options can be used, as well as the &APPC PREPARE-TO-RECEIVE TYPE=CONFIRM and &APPC DEALLOCATE TYPE=CONFIRM options.

This operand can be overridden by the SYNC value specified on the conversation allocation.

SECURITY={ NONE | SAME | USER | USERPSWD }

Applies to both locally and remotely allocated conversations, and specifies the security level to be implemented for this transaction. It corresponds to the LU6.2 architected SECURITY parameter of the ALLOCATE verb.

If SECURITY=NONE is specified (or defaulted) then no security checks are requested. For a local allocation the security access information fields are omitted from the allocation attach header. For a remote allocation request the security access information fields in the allocation attach header (if present) are ignored.

If SECURITY=SAME is specified, then the allocation will assume the security profile of the request source. For a local allocation from a procedure executing in any environment where the user is not signed on to the SOLVE system, the result is as if SECURITY=NONE is specified, and where the user is signed on as if SECURITY=USER is specified.

For an attach request, if the security access fields are omitted from the attach header, then SECURITY=NONE is assumed. If a user ID is supplied with the already verified indicator set, then the allocation is accepted on behalf of that user, unless rejected by the security exit. If both the user ID and password are supplied, then they must be verified by the security exit before execution is accepted on behalf of that user.

If SECURITY=USER is specified, then for a local allocation request where the user has been verified by the local security exit, the user ID is passed to the remote end with the *already verified* indicator set. If the user ID is not known (that is, the user is not signed on to the SOLVE system), the allocation request will fail. For an attach request, the attach header must carry either a user ID with the *already verified* indicator set, or a user ID and password, and be approved by the local security exit, otherwise the request is rejected.

If SECURITY=USERPSWD is specified, then for a local allocation request where the user has been verified by the local security exit, both the user ID and password will be passed to the remote system if known, otherwise the allocation request will fail. For an attach request, the attach header must carry the user ID and password, and be approved by the local security exit, otherwise the allocation request is rejected.

CMDAUTH=*nn*

This operand applies to attach requests only, and specifies the command authority with which the nominated NCL procedure is to be started.

The command authority of the NCL procedure will be determined in the following way:

- It the attached transaction is a secure transaction, an APPC user region is signed on and the NCL procedure executes with the user's defined command authority, and this operand is ignored.
- If the attached transaction is unsecured, it executes in the BSYS region with a command authority that is the minimum of the BSYS user command authority or the CMDAUTH specified by this operand.
- If this operand is omitted, CMDAUTH defaults to zero.

CONVTYPE={ MAPPED | BASIC }

Applies to locally allocated conversations only, and specifies whether the LU6.2 conversation is to be allocated as a mapped or basic conversation type. NCL processing is designed primarily for mapped conversations and the NCL &APPC verb implements the LU6.2 Mapped Conversation verb set, so the CONVTYPE=MAPPED option is strongly recommended, and is the default. The CONVTYPE=BASIC option is only used to set an indication in the attach header that the conversation is basic where the parameter expects a basic conversation to be allocated. Note that regardless of the CONVTYPE setting, the conversation is operated according to the OPERTYPE option.

SIGNON={ YES | NO }

Specifies the persistent verification option to be sent to the remote end on an allocation request. If SIGNON=NO is specified or defaulted, the attach header carries no persistent verification indicators.

If SIGNON=YES is specified, the persistent signon indicator is sent to the remote LU, unless a previous transaction for the same (USER, LU) pair specified SIGNON=YES. In this case, the already signed on indicator is sent.

This operand allows procedures to establish a persistent signon environment in the remote system in which future transactions will be targeted. This will avoid the overhead of signon and signoff with each conversation when it is known that multiple conversations will be conducted with the same system and user ID.

Examples:

```
REPTRANS TRAN=DBQUERY TPN=DB01 LINK=NMB MODENAME=NMMODE
REPTRANS TRAN=GETREC PROC=APPC01 SEC=USER
```

Notes: The TCT is used by LU6.2 services.

See Also: The DEFTRANS, DELTRANS, and SHOW DEFTRANS command description.

REQMS

Function: Sends data across the CNM interface.

REQMS (2)	*data*

Use: This command is used to send a message across the CNM interface.

Operands: ***data***

The data to be sent across the CNM interface, expressed in hexadecimal, in the format of a Forward RU. The data must contain only hexadecimal characters (0 to 9, A to F), and must be an even number of characters, as each two characters represent one hexadecimal byte.

Examples: `REQMS 810810000000000841030400000000081F103D7E4C1F103D7E4C1`

This command shows how to solicit link test statistics from a PU called PUA. The content of the data is broken down as follows:

`810810`	this is a forward RU, format 0
`00`	embedded RU solicits a reply
`00`	embedded RU contains a CNM header
`00`	reserved
`0008`	length of embedded NS RU

The embedded NS RU starts:

`410304`	NS RU is REQMS (request maintenance statistics)

The CNM header starts:

`0000`	CNM target ID - supplied by SSCP
`0000`	CNM target ID descriptor and reply correlator descriptor supplied by SSCP, correlator supplied by NEWS when data is sent to SSCP
`81`	X’80’ bit indicates to reset statistics in the target node. X’01’ indicates type 1 REQMS, soliciting link test statistics
`F1`	destination is a PU
`03`	length of destination PU name
`D7E4C1`	name of the destination PU in hexadecimal (PUA)

`F1`	target is a PU
`03`	length of target PU name
`D7E4C1`	name of the target PU in hexadecimal (PUA)

Notes:

It is the responsibility of the issuer of the command to ensure that the data is of the correct format. The appropriate *VTAM Programming Guide* and the *SNA Format and Protocol Reference* manual contain information on the data formats expected by SSCP and PUs. NEWS performs only the following checks on data content before forwarding it to SSCP:

- All data is valid hexadecimal.
- The data is at least 32 characters (16 hexadecimal bytes) in length. This is because NEWS must be able to supply the reply correlator in the applicable field so that the reply to the REQMS can be routed to the issuer of the command.

The reply to the REQMS command can be an error message if, for instance, the PU name supplied is unknown, the PU is inactive, or the data forwarded is invalid. The PU itself can reject the request, or it can be rejected by the SSCP. If, on the other hand, a Deliver RU is received in reply, it is broken up into 120-character (60 hexadecimal byte) lines, each prefixed by a NEWS message number, which are then sent to the issuer of the command. If the example shown above were entered from an OCS window, the reply might be:

```
N39701 810812000000001241038 44A36100281017000000000000A
000AF104D5C3D7C1F103D7E4C1F904D3D5C5C1F104D5C3D7C100
N39703 *END*
```

The data has the following meaning:

`81081200`	this is a deliver RU, format 0
`00`	embedded RU contains a CNM header
`00`	reserved
`0012`	length of embedded NS RU

The embedded NS RU starts:

`410384`	NS RU is RECFMS (record formatted maintenance statistics)

The CNM header starts:

`4A36`	CNM target ID - supplied by SSCP
`1002`	CNM target ID descriptor and reply correlator -descriptor supplied by SSCP, correlator supplied by NEWS when data is sent to SSCP

`81`	X'80' bit indicates this is a solicited reply. X'01' indicates this is type 1 data (link test statistics)
`01700000`	block and ID number
`0000`	reserved
`000A`	count of test frames received
`000A`	count of test frames responded to
`F1`	originator is a PU
`04`	length of originator's name
`D5C3D7C1`	name of originator (NCPA)
`F1`	target is a PU
`03`	length of target PU name
`D7E4C1`	name of the target PU in hex (PUA)

Network hierarchy follows -

`F9`	name is a link
`04`	length of link name
`D3D5C5C1`	name of the link (LNEA)
`F1`	name is a PU
`04`	length of PU name
`D5C3D7C1`	name of PU (NCPA)
`00`	no more name fields

The REQMS command is not designed to be used directly from an OCS window. Rather, it is used from an internal environment (&INTCMD) from within an NCL procedure. Having issued the command, the NCL procedure then issues one or a number of &INTREAD statements to retrieve the results. These which would normally be the reply sent by the target PU.

It is recommended that the NCL statements &CNMSEND and &CNMREAD be used in preference to the REQMS command as they provide greater flexibility.

See Also: The &CNMSEND and &CNMREAD verb descriptions in the *Network Control Language Reference*.

RETURN

Function: Exits from any mode or function and returns to the primary menu.

```
RETURN
(0)
```

Use: The RETURN command is used to exit from any mode or menu and immediately return to the primary menu.

By default, the RETURN command is associated with the F4 and F16 function keys. Functions other than OCS always have the RETURN command assigned to F4 and F16.

If the RETURN command is executed from the Primary Menu the current operational window is terminated. If no other operational window remains, the user is logged off.

The RETURN key can also be used to skip straight to a desired function. Enter =*n.n* in any input field of a panel and press the RETURN or ENTER key to skip to menu *n.n*.

Notes: When exiting OCS, current PROFILE values will be reset. Any function key values will be reset to the global values and any NCL process invoked from the window by EXEC or START commands will be flushed. Any local equates defined will also be deleted.

ROUTE

Function: Sends a command string to a remote system.

```
ROUTE        { linkname | DOMAIN=domain-id | SSCP=sscp-name }
               text
(0)
```

Use: Users of the Remote Operator Facility (ROF) who have a ROF session with a remote system can send OCS commands to the remote system for execution, and receive the results of those commands at their local terminal. The ROUTE command identifies the system to which the command string is to be sent, and the text that is to be transmitted.

The results of a ROUTE command issued from an OCS window are returned to the window as conventional line-by-line messages. If an NCL process executing in the NCL processing environment associated with an OCS window issues a ROUTE command, the results return to the OCS window. If the NCL procedure executes a ROUTE command under its dependent processing environment (&INTCMD ROUTE), then the results of the command return to the procedure, which can issue an &INTREAD statement to retrieve them. Only the results of the ROUTE command issued from the internal command environment are returned; the results of other ROUTE commands issued concurrently from the OCS window return to the window.

Any number of NCL processes can issue &INTCMD ROUTE commands concurrently.

Operands:

linkname
The symbolic name for the remote system to which the command string is to be sent.

DOMAIN=*domain-id*
The domain ID of the remote system to which the command string is to be sent.

SSCP=*sscp-name*
The symbolic name of the SSCP which the target remote system is running under. This operand would normally be used to issue VTAM commands, and so, if more than one remote system is running under the specified SSCP, preference is given to a system with its PPO interface active.

text
The command string that is to be sent to the remote system. The contents of the command string are not validated before transmission; invalid commands are rejected by the remote system. Multiple commands can be concatenated in the command string provided that they are separated by a colon.

The ROUTE command does not allow for a single embedded colon (:). When the ROUTE command processor encounters a single colon in the string, it substitutes a semicolon (;). If the processor encounters two colons, it replaces the first with a blank and sends the second to the target system as part of the transmitted command.

For example, the command ROUTE ORLEANS PRO INITCMD='PRO AOMMSG=NO::START USERMON' results in the remote profile initial command being set to 'PRO AOMMSG=NO ;START USERMON'.

ROUTE

Examples:

```
ROUTE NEWYORK SHOW USERS
ROUTE LA TEST D BFRUSE
```

Notes:

Results returned will usually be prefixed by the MSGID allocated to the remote system when the link was defined by the LINK command (the default) or will be prefixed by the MSGID chosen by the user when the SIGNON command was issued. Alternatively, the user can specify a private prefix that is to be used instead of the link MSGID by nominating an alternative prefix on the SIGNON command used to establish the ROF session.

Use of the ROUTE command will invoke automatic ROF signon if no ROF session exists with the target system. If the user is not authorized to have a ROF session the command is rejected by the remote system.

It is recommended that the string ROUTE linkname be abbreviated by use of the EQUATE command to simplify command routing.

If the following command is issued by a user on SYS-A:

```
ROUTE SYS-B ROUTE SYS-C START PROC1
```

The command `START PROC1` is routed to SYS-C via SYS-B, with an automatic signon implied at SYS-C if required, and if the originating user is entitled to log on to SYS-C. This allows indirect access to SYS-C from SYS-A even if no direct path exists from SYS-A to SYS-C.

See Also:

The LINK (INMC), SIGNON, and EQUATE command descriptions.

The *Network Control Language User's Guide* describes internal command environments and concepts of NCL processing environments.

SAWARE

Function: To control NTS activity.

```
SAWARE    START [ ={ WARM | COLD } ]
          [ LOG={ YES | NO } ] [ NOVTAM ] [ QUEUE ]

SAWARE    STOP [ CLOSE | CLOSE=ALL ]
          [ WAIT={ YES | NO } ]

SAWARE    LOG={ YES | NO }
(4)
```

Use: This command is used to control the operation of NTS session awareness processing. Since no other NTS data is collected until session awareness processing is activated, SAWARE is the principal command for controlling NTS activity.

Operands:

START [={ WARM | COLD }]

Requests NTS to start session awareness processing. Unless session awareness processing is active, no other data is collected by NTS. START causes NTS to start its acceptance of session awareness information from VTAM.

Specifying START=COLD causes NTS to present all session start information for already existing sessions to the NTS User Exit in the form of NTS session start SMF records. Specifying START=WARM causes NTS to present session start information to the exit for only those sessions that start after NTS has initialized.

The default for this operand depends on whether this is the first invocation of the SAWARE command since Management Services was started. If so, the default is START=COLD; otherwise it is START=WARM.

NOVTAM

Used in conjunction with the SAWARE START command, this operand specifies that this NTS system only receives data from an ISR conversation with other NTS systems. It does not receive any data directly from VTAM in the domain in which NTS is executing.

QUEUE

Optionally indicates that the SAWARE START request is to be queued if the CNM Interface is not available. When the CNM ACB is opened NTS sends all queued requests to VTAM. By specifying this option on a SAW START command, for example in the NMINIT procedure, session awareness can be automatically started when the CNM ACB becomes available during start-up. Note that this can also be achieved through the appropriate setting of options contained on the NTS System Support panel.

STOP [CLOSE | CLOSE=ALL]

Requests that NTS stop processing session awareness information from VTAM. After all session ended data currently queued for output processing has been serviced the remaining active session data held in storage by NTS is discarded.

If the CLOSE option is specified then NTS considers all currently active sessions as *closed* and queues them for output before terminating. Such sessions are examined and logged according to their previously assigned SAW class log options.

If the CLOSE=ALL option is used then all currently active sessions are closed and queued for output processing where they will be unconditionally logged (that is, all session data is logged regardless of the SAW class options).

When reviewing any sessions which have been closed by NTS they will appear on the NTS session lists flagged with a C indicating that the session was closed by NTS, and the session end time displayed is the time at which the SAW STOP command was entered.

WAIT={ YES | NO }

Specifies whether RTM data will be solicited and collected prior to the completion of the CLOSE processing. This option is only meaningful with the CLOSE option.

Specifying WAIT=YES may cause a significant increase in time for the SAWARE CLOSE processing.

LOG={ YES | NO }

Determines the current logging option required. When a SAW START command is issued the default LOG=YES value specifies that NTS is to attempt to connect to the NTSLOG file ID and commence logging for ended sessions. This operand can also be used in isolation to stop and start logging while session awareness remains active, but is ignored if entered in conjunction with STOP or QUEUE operands.

Examples:

```
SAWARE START LOG=NO
SAWARE STOP CLOSE=ALL
SAWARE LOG=YES
```

SCRIPT

Function: Activates or flushes an MAI-FS script procedure.

```
SCRIPT        { START | START=procname | FLUSH }
              { ID=session-id | RELNUM=nnnn }
              [ DATA=script-parameters ]
(0)
```

Use: To control the processing of MAI-FS script processes.

An MAI-FS session can have an NCL process associated with it, that intercedes on the session flows. The SCRIPT command can be used to start or stop these NCL processes. When starting a script procedure, optional data can be passed as parameters to the procedure.

Operands:

START | START=*procname*
Specifies that the associated script procedure is to be initiated for the supplied session. A class A user can optionally override the defined script procedure name by specifying START=*procname*.

FLUSH
Causes the immediate flushing of an active script process for the specified session.

ID=*session-id*
Identifies the MAI-FS session that is to be affected by the command. Alternatively, a relative session can be used.

RELNUM=*nnnn*
Supplies a relative session number that is to be used to identify the target session.

DATA=*script-parameters*
Supplies optional data to be passed as the parameters on a script procedure initiation. This operand can be used only with the START operand.

Examples:

```
SCRIPT START ID=CICS01
SCRIPT FLUSH ID=TSO2
SCRIPT START ID=IMSPROD DATA=RX00 MENU01 OPT=ALL
```

Notes: The script procedure is initiated as if started via a user request. The script mode is set as SKIP. Scripting is a component of the MAI/EF feature. The SCRIPT command is only available if MAI/EF is configured.

See Also: The SHOW MAI and MAISESS command descriptions.

SECUSER

Function: Issues a command to a disconnected VM service machine.

```
SECUSER      userid
             [ OPT={ CONN | EXCL | REL } ]
             [ DATA=command-text ]
(2)
```

Use: In OCS mode, an operator can use the SECUSER command to send operating system commands to a VM service machine and have the resulting message responses delivered back to the originating OCS window.

Operands:

userid

Specifies the name of the VM service machine.

OPT={ CONN | EXCL | REL }

OPT=CONN (the default) establishes a session with the specified VM service machine and, if the DATA= operand is specified, issues the command provided.

OPT=EXCL establishes an exclusive session with the specified VM service machine and, if the DATA= operand is specified, issues the command provided.

OPT=REL terminates the session with the specified VM service machine. This operand is mutually exclusive with the DATA= operand.

DATA=*command-text*

Specifies the text of the operating system command to issue.

Note

The DATA operand is required if command-text is specified, but not permitted when OPT=REL is specified.

Examples:

```
SECUSER VMUSER2
SECUSER VMUSER6 OPT=REL
```

Notes: Once an OPT=EXCL session has been established between a SOLVE system and a VM service machine, the SOLVE system prevents any further SECUSER sessions being established with the VM service machine.

OPT=EXCL will only be accepted if there are NO current SECUSER sessions active with the same VM service machine.

Once a session has been established with VMOPERATOR it will remain in place until SECUSER OPT=REL is specified or until AOM or VMOPERATOR terminate.

SHOW ALLOC

Function: Displays datasets allocated by the ALLOCATE command.

```
SHOW  ALLOC     [ =ddname ]
(0)
```

Use: Datasets can be dynamically allocated by using the ALLOC command. The SHOW ALLOC command provides details on the allocation, including the DD name, dataset name, current dataset status, the user who performed the allocation, and the time of allocation. If the dataset was a member of a concatenation (z/OS or OS/390), then the CONCATDD column provides the DD name behind which this dataset was concatenated. In VM/SP and VSE/SP systems, only VSAM datasets can be allocated by the ALLOCATE command. The CAT column of the display shows the associated VSAM catalog DLBL name, if applicable.

Operands:

ALLOC

Displays current dynamically allocated dataset entries.

ALLOC=*ddname*

Displays the dataset identified by the specific DD name entered. If *ddname* ends with an asterisk (*), then it is treated as a generic value and all allocated datasets whose DD names match the generic value are listed.

Examples:

```
SHOW ALLOC
SHOW ALLOC=IMNDB
SHOW ALLOC=IM*
```

Notes: If an allocation specifying concatenation was requested and the allocation was successful but the concatenation of DDs failed, then the CONCATDD column displays *ERROR*. Check the log at the time the allocation was requested to find out why the concatenation failed.

See Also: The ALLOCATE (z/OS, OS/390, MSP, and VOS3), ALLOCATE (VM/SP), ALLOCATE (VSE/SP), UNALLOC, and UDBCTL command descriptions.

SHOW AOMABEND

Function: Displays diagnostic information if the AOM z/OS, OS/390, MSP, or VOS3 subsystem interface code ABENDs.

```
SHOW AOMABEND
(0)
```

Use: If the AOM z/OS, OS/390, MSP, or VOS3 subsystem interface (SSI) code or SRB code abends, then this command provides information on the PSW, registers, and abend reason code; this information should be provided to your supplier.

This command is not provided in AOM/VM.

SHOW AOMSTAT

Function: Displays AOM statistics.

```
SHOW AOMSTAT    = { ALL | DELIVERY | PROC | TUNING } ]
(0)
```

Use: Display statistics about the AOM local system interface, or primary AOMPROC queues.

Operands:

Note

If SHOW AOMSTAT is entered without any other operands, then a summary showing date and time started and total message statistics is displayed.

ALL

This operand displays all AOM statistics, as produced by the individual display options. AOM must be started when this operand is used.

DELIVERY

This operand displays statistics about messages and DOMs received and delivered by the local operating system interface. AOM must be started when this operand is used.

PROC

This operand displays statistics about messages queued to the primary AOMPROC, from any source. Since AOM need not be started for ISR traffic and messages from &AOMALERT to be queued, this option is valid at any time.

TUNING

This operand displays statistics related to tuning the local operating system interface; for example, queue sizes, high water marks, and storage utilization statistics.

Examples: `SHOW AOMSTAT=ALL`

Notes: SHOW AOMSTAT=TUNING can be used to fine tune pool and queue sizes.

SHOW APPC

Function: Displays the status of all or selected APPC conversations.

```
SHOW        APPC
            [ TYPE={ USER | SYSTEM } ]
            [ ID=convid | NCLID=nclid ]
            [ USERID=userid*  ]
            [ SERVER=servername*  ]
            [ TRANSID=transid* ]
            [ LINK=link*  | LUNAME=luname*  | LOCAL ]
            [ MODENAME=modename* ]
(0)
```

Use: The SHOW APPC command is used to display the status of APPC conversations. If the ID parameter is not specified, then the display shows the following information for each listed conversation:

- The user ID of the region in which the conversation is running
- The NCLID of the NCL process in which the conversation is running
- The server name, if the process is registered as a server
- How the procedure was invoked: SERVER (allocated) or CLIENT (attached)
- The conversation ID
- The transaction ID
- The conversation state
- The conversation partner information: LUNAME for remote partners; conversation ID and NCLID for local partners

Additional information is displayed for a specific conversation, if the conversation ID is specified.

Operands:

APPC

Indicates that this is an APPC conversation status display.

TYPE={ USER | SYSTEM }

Indicates the class of transactions to be displayed. TYPE=SYSTEM selects system transactions, such as OSCALL, for display.

ID=*convid*

Indicates a specific conversation identifier for the SHOW APPC command. If present, this alters the displays from the usual list of APPC conversations (that satisfy any other operands) to a detailed display for the conversation number requested.

NCLID=*nclid*

Indicates a specific NCL process identifier for which APPC conversation information is to be displayed. No information is displayed if the process has no APPC conversation running.

USERID=*userid**

Optionally restricts the display to those NCL processes running under the user ID provided.

SERVER=*servername**
Optionally restricts the display to those conversations registered as a server with the *servername* specified.

TRANSID=*transid**
Optionally restricts the display to those conversations processing transactions which have the full or generic *transid* specified.

LINK=*link**
Optionally restricts the display to those conversations using the APPC link(s) indicated by the full or generic *link* name specified. This parameter is mutually exclusive with the LUNAME and LOCAL parameters.

LUNAME=*luname**
Optionally restricts the display to those conversations using the remote LU name indicated by the full or generic *luname** provided. This parameter is mutually exclusive with the LINK and LOCAL parameters.

LOCAL
Optionally restricts the display to those conversations running entirely within the local system only. This parameter is mutually exclusive with the LINK and LUNAME parameters.

MODENAME=*modename**
Optionally restricts the display to those conversations using the mode name indicated by the full or generic *modename** specified.

Examples:

```
SHOW APPC
SHOW APPC TRAN=TP*
SHOW APPC LINK=NMA
SHOW APPC LINK=NM*
SHOW APPC LOCAL
SHOW APPC ID=10
SHOW APPC USERID=UI00080
```

Notes:

The NCLID of any procedure participating in an APPC conversation can be obtained from the display shown when the conversation ID parameter is specified in the SHOW APPC command.

If the ID or NCLID operands are not specified, then the display is organized in order of user ID asked for. Each user ID is in order of NCL process ID. For each NCLID, all APPC conversations are divided into CLIENT (allocated) or SERVER (attached) conversations.

The number of processes displayed can be restricted by specifying the USERID and/or SERVER operands. The number of conversations displayed can be further restricted by specifying the TRANSID and/or one of the destination parameters (that is, LINK, LUNAME, or LOCAL).

See Also:

The SHOW NCL, SHOW USERS, and SHOW LINK command descriptions.

SHOW ATF
SHOW ATF TABLE
SHOW ATF ENCODING

Function: These commands display information about the ATF tables.

SHOW ATF { TABLE=*name* | ENCODING } (0)

Use: SHOW ATF displays a summary list of all defined translation names.

SHOW ATF TABLE=*name* provides detailed information on a specific translation name entry.

SHOW ATF ENCODING provides a list of valid encoding rule names. These names must be used as the ENCODING operand of the ATF DEFINE command and the ENCODING statement for source files.

Operands:

Note

The absence of operands indicates that a summary list of defined ATF table names is requested.

TABLE=*name*
Indicates that detailed information about the named ATF table is requested.

ENCODING
Indicates that a list of ATF encoding rule names is requested.

Examples:

```
SHOW ATF TABLE=MYASCII
```

Notes: This command is useful to see what tables are loaded and, when producing new tables, to see the valid encoding names that are unavailable.

See Also: The ATF DEFINE, ATF LOAD, and ATF UNLOAD commands.

SHOW BRO

Function: Displays the current general broadcast messages.

```
SHOW BRO
(0)
```

Use: The system supports broadcast messages that can be sent to EASINET and MAI users' terminals. The SHOW BRO command can be used to display the current broadcast messages.

Notes: When a new broadcast message is set, all terminals in OCS mode are notified of the new text that has been set.

See Also: The NSBRO command description.

SHOW BUFF

Function: Displays the current system storage and buffer utilization.

```
SHOW  BUFF   [ =name ]
(0)
```

Use: The system maintains statistics on its internal buffer utilization, the maximum amount of storage that has been used, and the current amount in use. This information is normally of interest only in problem determination. However, the storage figures can help to determine the region size that is required to support your particular terminal configuration.

Operands: ***=name***

Used to display a specific set of buffers. A generic range of buffer pool names can be displayed by ending *name* with an asterisk.

Examples:

```
SHOW BUFF
SHOW BUFF=ABCD
SHOW BUFF=AB*
```

Notes: More information on the particular fields can be obtained by referencing the message numbers associated with each display line.

SHOW CNMTRACE

Function: Shows active CNM trace requests.

```
SHOW CNMTRACE   [ RU={ rutype | CNMRU | DELIVER |
                     FORWARD | NMVT | RECFMS |
                     RECMS | REQMS | TR-INQ |
                     TR-REPLY | ALIAS } ]
                [ PU=puname ]
                [ IN ] [ OUT ]
                [ SOL ] [ UNSOL ]
                [ LOCAL ] [ REMOTE ]
                [ LINK=linkname ]
(0)
```

Use: The NEWS feature of NetMaster for SNA performs tracing of CNM RUs being delivered and received through the local VTAM CNM interface and from remote systems using ISR. The SHOW CNMTRACE command returns a list of all active CNM trace requests which satisfy all of the supplied screening criteria (such as RU type and link name).

Operands:

RU=*rutype*

Specifies an RU type which is used to restrict the display. Trace requests which do not trace the specified RU type are excluded from the display. Values are:

CNMRU	x’810814’	CNM RU
DELIVER	x’810812’	DELIVER RU
FORWARD	x’810810’	FORWARD RU
NMVT	x’41038D’	NMVT RU
RECFMS	x’410384’	RECFMS RU
RECMS	x’010381’	RECMS RU
REQMS	x’410304’	REQMS RU
TR-INQ	x’3F0814’	Translate-Inquiry RU
TR-REPLY	x’3F0816’	Translate-Reply RU
ALIAS	x’3F08’	TR-INQ plus TR-REPLY

If RU= is not specified, then traces for all RU types are shown.

PU=*puname*

Specifies the name of the resource for which trace requests are to be shown. It can be generic, for example, PU=SDPU*. If PU= is not specified, then traces for all PUs are shown.

IN
Traces RUs which are inbound.

OUT
Traces RUs outbound.

If IN and OUT are both omitted, then traces for both inbound and outbound RUs are shown.

SOL
Traces solicited RUs.

UNSOL
Traces unsolicited RUs.

If SOL and UNSOL are both omitted, then traces for both solicited and unsolicited RUs are shown.

LOCAL
Traces local processing of RUs.

REMOTE
Traces remote (ISR) processing of RUs.

If LOCAL and REMOTE are both omitted, then trace requests for both local and remote RUs are shown.

LINK=*linkname*
Provides a specific link name to trace. It can be generic, for example, LINK=NML* .

If LINK= is not specified, then trace requests for all links are shown, unless LOCAL is specified and REMOTE is not.

Examples:

```
SHOW CNMTRACE IN LINK=NMP*
SHOW CNMTRACE RU=NMVT IN REMOTE UNSOL
SHOW CNMTRACE SOL PU=NCP*
```

SHOW COMMANDS

Function: Displays commands available to the user or the system.

```
SHOW COMMANDS   [ ={ OWN | ALL } ]
(0)
```

Use: Displays a list of commands.

Operands:

OWN

Displays an alphabetical list of the commands that can be executed by this user, honoring the user's command authority. Note that submitting to another environment preserves the user ID and command authority levels.

Note

A displayed command might not be available, as it could require further authorization; for example, the SYSCMD command.

ALL

Displays a list of all commands available in this system.

Examples:

```
SHOW COMMANDS
SHOW COMMANDS=ALL
```

SHOW COMP

Function: Displays datastream compression statistics.

```
SHOW COMP      [ =ALL ]
(0)
```

Use: The SHOW COMP command can be used to display data stream compression statistics. The MS region compresses and optimizes outbound 3270 data streams to ensure efficient use of terminals and lines. Information returned by this command includes the total number of bytes that have been eliminated from outbound data streams, and the compression ratio, expressed as an average percentage of the total number of bytes scanned for compression.

Operands: **=ALL**

Displays compression statistics for both MAI traffic and MS panel traffic.

Examples: `SHOW COMP=ALL`

Notes: Compression is performed only on screen panels that can be improved by optimization.

If datastream compression detects errors in data streams, then messages are written to the activity log pinpointing the location of the errors.

SHOW CONSOLES

Function: Displays the consoles currently allocated to Management Services for use by AOM.

```
SHOW CONSOLES
(0)
```

Use: The SHOW CONSOLES command is used to list all consoles acquired by Management Services, together with their current disposition.

This command is not supported by AOM/VM.

Examples: `SHOW CONSOLES`

produces a response such as:

```
sh consoles
N86E01 CONSOLE INFORMATION FOLLOWS...
N86E10 SYSPARM SETTIONS...
N86E11 CONSOLES AOMCTYPE AOMCMIGI AOMCUTOK AOMCTO1  AOMCTO2  AOMCTO3
N86E12 (5,2)    EXTMCS  Y        Y       10       10      2
N86E13 AOMCMIGX AOMCOPTS AOMJESCH AOMSUBCH
N86E14 NO       81       NO       NO
N86E20 STATISTICS...
N86E21 #-LOG-AQ #-PHY-AQ %-PL-A AV-TM-PA #-LOG-RL #-PHY-RL %-PL-R AV-TM-PR
N86E22       7        5  71.42     1.00        1        0   0.00     0.00
N86E23 #-LA-MIG %-LA-M #-PA-MIG %-PA-M #-LA-FNC %-LA-F #-PA-FNC %-PA-F
N86E24        4  57.14        3  60.00        0   0.00        0   0.00
N86E30 C#  ID  NAME     STATUS   LOCK   AUTH     USERID   ENV      W T/O
N86E31   1 202 ZDPMA001 IN-USE   AOM    INFO     DPMAAOMP AOM      - -
N86E31   2   - ZDPMA002 IN-USE   SYSCMD MASTER   USER01   000020   1 -
N86E31   3 138 ZDPMA003 IN-USE   SYSCMD MASTER   USER01   PRIMARY  2 -
N86E31   4 146 ZDPMA004 POOL     -      MASTER   -        -        - -
N86E31   5   - ZDPMA005 IN-USE   -      MASTER   USER01   000022   1 6
N11907 *END*
```

Notes: Relevant SYSPARM settings are displayed.

Statistics on the number of logical and physical console allocation and release requests are displayed. These statistics include percentages.

Statistics are displayed on the average time taken by a physical allocation or release. These can be used to detect system problems (such as ENQ contention).

Counts and percentages are displayed for failures to obtain consoles, either logical (because the pool was full) or physical (because no consoles were available). These statistics can be used to indicate if the console pool size needs to be increased.

Statistics, including percentages, are displayed on the number of requests (logical and physical) that asked for a migration ID.

SHOW CONSOLES

See Also: SYSCMD and SYSPARMS CONSOLES=*nn* command descriptions.

SHOW DEBUG

Function: Displays NCL debug sessions.

```
SHOW DEBUG    [ ={ ALL | userID } ]
              [ ID=nclid ] [ USER=userID ]
              [ REGION=luname ]
              [ WINDOW=number ]
              [ PROCEDURE=procname ]
              [ NCL={ YES | NO } ]
(0/2)
```

Use: Lists the debug sessions currently active in the system.

Operands:

DEBUG

This keyword requests the display of NCL DEBUG sessions.

={ ALL | *userID* }

The SHOW DEBUG=ALL command displays all the debug sessions for the issuer's user ID. If the issuer is suitably authorized, then all debug sessions for all users are displayed.

The SHOW DEBUG command, by default, displays only debug sessions owned by the current issuing environment. The SHOW DEBUG=*userID* command displays debug sessions owned by the specified user ID. If the user ID specified is not the user's own ID, an authority level of 2 or higher is required.

ID=*nclid*

Indicates that a display of any debug session that has specified an NCL ID as the scope is required.

Note

If ID is specified, then no other operand can be specified.

USER=*userID*

Indicates that any debug session that has specified this user ID in a scope is to be listed.

REGION=*luname*

Indicates that any debug session that has specified this region name in a scope is to be listed.

WINDOW=*number*

Indicates that any debug session that has specified this window number in a scope is to be listed.

PROCEDURE=*procname*

Indicates that any debug session that has specified this procedure name in a scope is to be listed.

SHOW DEBUG

NCL={ YES | <u>NO</u> }

If NCL=NO is specified, or defaulted, then only the scopes of the debug sessions eligible for display are listed. This is the default, except when the ID operand is specified.

If NCL=YES is specified, then a display is included of the NCL processes currently under the control of the debug sessions requested on the SHOW DEBUG command.

Examples:

```
SHOW DEBUG
SHOW DEBUG USER=U01829 NCL=YES
SHOW DEBUG PROCEDURE=USRMENU NCL=YES
SHOW DEBUG ID=123456
```

Notes: If no operands are specified, then all debug sessions are listed for the command issuer's user ID.

See Also: The DEBUG command description.

SHOW DEFALIAS

Function: Displays one or more DEFALIAS entries used by the Alias Name Translation Facility of NEWS.

```
SHOW  DEFALIAS   [ NAME=alias ]
                 [ NET=aliasnet ]
                 [ RNAME=realname ]
                 [ RNET=realnet ]
                 [ RCDRM=cdrm ]
                 [ TYPE={ LU | COS | MODE } ]
(0)
```

Use: This command is used to display one or more alias name entries defined to NEWS to be used by the Alias Name Translation Facility. These entries are created through the DEFALIAS command.

Operands:

NAME=*alias*
Specifies 1 to 8 characters to be used in a generic comparison against defined alias or origin names to restrict the display.

NET=*aliasnet*
Specifies 1 to 8 characters to be used in a generic comparison against defined alias or origin networks to restrict the display.

RNAME=*realname*
Specifies 1 to 8 characters to be used in a generic comparison against defined real or target names to restrict the display.

RNET=*realnet*
Specifies 1 to 8 characters to be used in a generic comparison against defined real or target networks to restrict the display.

RCDRM=*cdrm*
Specifies 1 to 8 characters to be used in a generic comparison against defined CDRM names to restrict the display.

TYPE={ LU | COS | MODE }
Specifies which type of DEFALIAS names are to be displayed, the default being LU names.

Examples:

```
SHOW DEFALIAS
SHOW DEFALIAS NET=MYNET TYPE=COS
SHOW DEFALIAS RNAME=TSO TYPE=LU
SHOW DEFALIAS RCDRM=HOST43
```

SHOW DEFALIAS

Notes: The display can be restricted by specifying one or more of the operands mentioned. Only Alias Name entries which match the operand values specified are displayed.

See Also: The DEFALIAS, DELALIAS, REPALIAS, and XLATE command descriptions.

The *Unicenter NetMaster Network Management for SNA Implementation and Administration Guide*.

SHOW DEFCLASS

Function: Displays an NTS class definition.

SHOW DEFCLASS [{ SESSION | SAW | RTM | RESOURCE } [=*name*]] (0)

Use: The SHOW command is an MS command, but NTS recognizes the DEFCLASS operand and displays NTS class definitions.

Operands: **DEFCLASS [{ SESSION | SAW | RTM | RESOURCE } [=*name*]]**

Results in a display of the DEFCLASS definitions. An optional *name*, or partial name, can be specified after any operand to restrict the classes displayed to those beginning with the partial name only.

If the SESSION operand is specified (or defaulted), then the SHOW DEFCLASS displays all session classes, or those beginning with *name* if SESSION=*name* was specified. The information displayed for each class selected includes the session selection criteria defined for the class, and the SAW and RTM classes associated with each class.

If the SAW operand is specified, then all SAW classes, or those beginning with *name* if SAW=*name* was specified, are selected for display. The information displayed shows all options for the particular SAW class.

If the RTM operand is specified, then all RTM classes, or those beginning with *name* if RTM=*name* was specified, are selected for display. The information displayed shows all options for the particular RTM class.

If the RESOURCE operand is specified, then all resource classes, or those beginning with *name* if RESOURCE=*name* was specified, are selected for display. The information displayed shows all options for the particular RESOURCE class.

Examples:

```
SHOW DEFCLASS
SHOW DEFCLASS RTM=CICS
SHOW DEFCLASS SAW=NM
SHOW DEFCLASS RESOURCE=TSO
```

Notes: For a SHOW DEFCLASS command, the results appear under the following headings:

SESSION

Provides the name of the session class.

PRI-NAME

Provides the name defined using the PRI operand of the DEFCLASS command for this session class, and nominates the primary session partner name required for a session to be selected in this class. Wild characters in the name are displayed as asterisks, and a dash indicates that only the portion of the name up to the dash is to be used when logging the session.

SEC-NAME

Provides the name defined using the SEC operand of the DEFCLASS command for this session class, and nominates the secondary session partner name required for a session to be selected in this class. Wild characters in the name are displayed as asterisks, and a dash indicates that only the portion of the name up to the dash is to be used when logging the session.

COSNAME

Provides the name defined using the COS operand of the DEFCLASS command for this session class, and nominates the class of service name required for a session to be selected in this class. If the name is all asterisks, then any COSNAME is acceptable.

ER

Provides the value defined using the ER operand of the DEFCLASS command for this session class, and nominates the Explicit Route number required for a session to be selected in this class. If the value is an asterisk, then any ER is acceptable.

VR

Provides the value defined using the VR operand of the DEFCLASS command for this session class, and nominates the Virtual Route number required for a session to be selected in this class. If the value is an asterisk, then any VR is acceptable.

TP

Provides the value defined using the TP operand of the DEFCLASS command for this session class, and nominates the Transmission Priority number required for a session to be selected in this class. If the value is an asterisk, then any TP is acceptable.

CL

Provides the value defined using the CL option of the DEFCLASS command for the session class (SD, XD, XN), and nominates the session CLASS required for a session to be selected in this class. If the value is an asterisk, then any CL is acceptable.

TY

Provides the value defined using the TY option of the DEFCLASS command for the session class (LL, SL, SP, SS), and nominates the session CLASS required for a session to be selected in this class. If the value is an asterisk, then any TY is acceptable.

SAWCLASS

Provides the name of a SAW class definition which is to be used to extract the SAW class values for sessions using this session class. A dash indicates that no SAW class name is defined.

RTMCLASS

Provides the name of an RTM class definition which is to be used to extract the RTM class values for sessions using the session class. A dash indicates that no RTM class name is defined.

For a SHOW DEFCLASS SAW command, the results appear under the following headings:

CLASS
The name of the SAW class definition being displayed.

KEEP
YES or NO; indicates whether or not sessions using this SAW class are to be kept by NTS.

LOG
Provides the log option specified on the LOG operand of the DEFCLASS command for the SAW class. The characters SUMM indicate that the SUMMARY option was specified or defaulted. Other values are as for the DEFCLASS command.

ACCT
YES or NO; indicates the selective accounting option for sessions using this SAW class.

ITRC
Provides the initial trace queue depth to be used by sessions with this SAW class.

FTRC
Provides the final trace queue depth to be used by sessions with this SAW class.

EVENT
YES or NO; indicates the generation of system events when the session is started or stopped, fails, or the RTM objectives are exceeded.

For a SHOW DEFCLASS RTM command, the results appear under the following headings:

CLASS
Provides the name of the RTM class being displayed.

OBJTIME
Provides the RTM class objective response time.

OBJ%
Provides the RTM class objective percentage.

RTMDEF
Provides the RTM definition that is used to delimit response time measurements for this class. The values are as supplied on the RTMDEF operand of the DEFCLASS command.

BNDY-1 BNDY-2 BNDY-3 BNDY-4
Provide the boundary time values for the (up to) four boundaries in this class, as supplied by the BOUNDS operand of the DEFCLASS command.

SHOW DEFCLASS

For a SHOW DEFCLASS RESOURCE command, the results appear under the following headings:

CLASS
The name of the resource class definition being displayed.

LINK
Provides the name defined using the LINK operand of the DEFCLASS RESOURCE command, and nominates the link name to be matched for a resource to be selected in this class. If this parameter is not specified in the resource class definition, then a single arrow (>) is displayed. If a partial name is specified, then the required characters are followed by an arrow. Wildcard characters in the name are displayed as asterisks.

PU
Provides the name defined using the PU operand of the DEFCLASS RESOURCE command, and nominates the PU name to be matched for a resource to be selected in this class. If this parameter is not specified in the resource class definition, then a single arrow (>) is displayed. If a partial name is specified, then the required characters are followed by an arrow. Wildcard characters in the name are displayed as asterisks.

LU
Provides the name defined using the LU operand of the DEFCLASS RESOURCE command, and nominates the LU name to be matched for a resource to be selected in this class. If this parameter is not specified in the resource class definition, then a single arrow (>) is displayed. If a partial name is specified, then the required characters are followed by an arrow. Wildcard characters in the name are displayed as asterisks.

STATS
YES or NO; indicates whether NTS is to collect resource statistics for resources selected in this class.

LIMIT
Provides the number of resource statistics collection intervals that can occur before the oldest interval is overwritten, for resources selected in this class.

RTMCLASS
Provides the name of an RTM class definition which is to be used by NTS for response time measurement statistics collection and analysis, for resources selected in this class.

See Also: The DEFCLASS command description.

SHOW DEFLINK

Function: Displays current dynamic link definitions.

```
SHOW        DEFLINK
            [ TYPE={ INMC | APPC | ALL } ]
(0)
```

Use: The SHOW DEFLINK command can be used to display dynamic INMC and APPC link definitions. The DEFLINK command is used to build definitions for dynamic INMC links. Dynamic links allow one MS region to accept a request for an INMC link from a second system even though no specific LINK command has been issued. Dynamic links are generally used where many different MS regions might need to be connected at varying times, particularly in the case of NMPC connections.

Operands: **TYPE={ INMC | APPC | ALL }**

Indicates whether INMC or APPC Dynamic Link Table definitions are to be displayed. The default is INMC link definitions, so TYPE=APPC must be entered to display APPC definitions.

Use TYPE=ALL to display both INMC and APPC link definitions.

Examples:

```
SHOW DEFLINK
SHOW DEFLINK TYPE=APPC
```

See Also: The DEFLINK (INMC), DEFLINK (APPC), DEFLINK (NVC), and DELLINK command descriptions.

SHOW DEFLOGON

Function: Displays the current entries in the DEFLOGON table.

SHOW DEFLOGON [=*name*] (0)

Use: EASINET and MAI features require that target applications be defined using the DEFLOGON command. These DEFLOGON entries are maintained in a table and can be modified using the REPLOGON command or deleted using the DELLOGON command. The full table of DEFLOGON entries can be displayed using the SHOW DEFLOGON command.

Operands: ***=name***

Displays the DEFLOGON entry whose INPUT= field matches the name specified. If name ends with an asterisk (*), then it is treated as a generic name and all DEFLOGON entries whose INPUT= fields match the generic name are displayed.

Examples:

```
SHOW DEFLOGON
SHOW DEFLOGON=IMS
SHOW DEFLOGON=TSO*
```

Notes: If the SUSLOGON command has been used to suspend the logon paths to one or more applications, those logon entries that have been suspended are displayed in high-intensity.

The SHOW DEFLOGON command displays the scanning option that applies when input is matched against the DEFLOGON table entries and determines whether the input must be an exact or generic match against the entry.

See Also: The DEFLOGON, REPLOGON, DELLOGON, SUSLOGON, and ACTLOGON command descriptions.

SHOW DEFMODE

Function: Displays APPC Mode Control Table (MCT) entries.

```
SHOW DEFMODE      [ MODE=mode* ]
(0)
```

Use: The SHOW DEFMODE command is used to display all or a selected number of MCT entries.

Operands: **MODE=*mode*****

Optionally restricts the display to those MCT entries with the mode name indicated by the full or generic *mode** name provided.

Examples:

```
SHOW DEFMODE
SHOW DEFMODE MODE=NM*
SHOW DEFMODE MODE=LU62PARL
```

Notes: The MCT is used by LU6.2 services and is maintained by the Systems Programmer. For more information see the chapters on APPC in the *Network Control Language User's Guide*.

See Also: The DEFMODE, REPMODE, and DELMODE command descriptions.

SHOW DEFMSG

Function: Displays the current DEFMSG delivery options.

```
SHOW        DEFMSG
(0)
```

Use: Displays the DEFMSG delivery options for VTAM PPO/SPO messages by VTAM message number. These are stored in a table from 0001 to 9999.

The delivery options are listed under the following headings:

Heading	Meaning
VTAM MSG#	Message number or range
PPO	YES/NO deliver to PPOPROC
LOC.	YES/NO deliver to LOCAL OCS users
REM.	YES/NO deliver to REMOTE systems
DEF.	YES/NO this is a DEFAULT status message
ACT.	YES/NO has been actioned by DEFMSG command
EVENTNAME	SNA event name suffix for EDS event notification

Notes: The command shows only those table entries for which a delivery option has been set.

See Also: The DEFMSG command description.

SHOW DEFOPSET

Function: Displays APPC Option Set Control Table (OSCT) entries.

```
SHOW DEFOPSET    [ OPSET=opset* ]
(0)
```

Use: The SHOW DEFOPSET command is used to display all or a selected number of OSCT entries.

Operands: **OPSET=*opset****
Optionally restricts the display to those OSCT entries with the *opset* name indicated by the full or generic *opset** name provided.

Examples:

```
SHOW DEFOPSET
SHOW DEFOPSET OPSET=NMOPSET
SHOW DEFOPSET OPSET=OP*
```

Notes: The OSCT is used by LU6.2 services and is maintained by the Systems Programmer. For more information see the chapter on APPC in the *Management Services User's Guide* and the *Network Control Language User's Guide*.

See Also: The DEFOPSET, REPOPSET, and DELOPSET command descriptions.

SHOW DEFTERM

Function: Displays terminal definitions (VOS3 only).

```
SHOW DEFTERM      [ =nnnnnnn ]
(0)
```

Use: The SHOW DEFTERM command is used to display all terminal definitions or a specific terminal definition. (The terminals listed are those defined using the DEFTERM command.)

Operands: ***=nnnnnnn***

Optionally restricts the display to the definition with the name *nnnnnnn*.

Examples:

```
SHOW DEFTERM
SHOW DEFTERM=L3270*
```

Notes: Terminal definitions are used in VOS3 systems to specify terminal types (3270 or 560/20).

See Also: The DEFTERM and DELTERM command definitions.

SHOW DEFTRANS

Function: Displays APPC Transaction Control Table (TCT) entries.

```
SHOW        DEFTRANS
          [ TRANSID=transid* ]
          [ QUAL=qual* ]
          [ TYPE={ USER | SYSTEM } ]
(0)
```

Use: The SHOW DEFTRANS command is used to display all or a selected number of TCT entries.

Operands:

TRANSID=*transid***
Optionally restricts the display to those TCT entries with the transaction name indicated by the full or generic *transid** name provided.

QUAL=*qual***
Optionally restricts the display to those TCT entries with a transaction qualifier indicated by the full or generic *qual** name provided.

TYPE={ USER | SYSTEM }
Indicates whether user or system transactions will be displayed.

Examples:

```
SHOW DEFTRANS
SHOW DEFTRANS TRANSID=TPCICS01
SHOW DEFTRANS TRANSID=TP*
SHOW DEFTRANS TRANSID=TP01 QUAL=TEST
SHOW DEFTRANS TRANSID=TP* QUAL=TEST
SHOW DEFTRANS TYPE=SYSTEM
```

Notes: The TCT is used by LU6.2 services and is maintained by the Systems Programmer. For more information, see the chapter on APPC in the *Management Services User's Guide* and the *Network Control Language User's Guide*.

See Also: The DEFTRANS, REPTRANS, and DELTRANS command descriptions.

SHOW DNR

Function: To display information about the SOLVE Domain Name Resolver (DNR).

```
SHOW DNR     [ PARMS ]
             [ STATS ]
(0)
```

Use: Displays parameters or statistics for the SOLVE DNR.

Operands: If no operands are provided, the DNR parameters are displayed, and, if the TCP/IP interface is active, the statistics are also displayed.

PARMS
Requests a display of the current setting of the DNR parameters.

STATS
Requests a display of the DNR statistics. These statistics include cache statistics, request statistics, and send/receive statistics.

Examples:

```
SHOW DNR
SHOW DNR PARMS
SHOW DNR STATS
```

Notes: If TCP/IP is not active when the STATS operand is used, then an error message is displayed.

All DNR statistics are reset if the NetMaster TCP/IP interface is restarted.

See Also: The DNR command.

SHOW DOMAINS

Function: Displays the domain information of connected or defined MS domains.

```
SHOW DOMAINS      [ =domainid ]
                  [ { TYPE={LINKED | DEFINED } ]
```

Use: When a link is established to another MS domain, the two domains exchange information about their respective environments. This includes the domain ID, which is specified by the NMDID initialization parameter.

The SHOW DOMAIN display lists each MS domain:

- That this domain is or has been connected to
- Defined using the DOMAIN command

The display gives one line of information for each connected domain. Note that this MS system's domain information is always displayed as the top line of the display, unless restricted by the =*domainid* operand and TYPE=LINKED.

Operands: ***=domainid***

Displays the domain information for the MS domain indicated by the nominated *domainid*.

TYPE= { LINKED | DEFINED }

The LINKED setting displays all domains for which a link has been activated since the MS system initialized. This includes those not active when the SHOW DOMAINS command is issued.

The DEFINED setting displays all domains that have been defined:

- Explicitly with the DOMAIN command (including domains for which a link has not been established)
- Implicitly by an inbound link

Examples:

```
SHOW DOMAINS
SHOW DOMAINS=NET1
SHOW DOMAINS=NET1 TYPE=DEFINED
```

Notes: If non-unique domain IDs occur, the links between the two MS domains that have the same domain ID work correctly, but certain optional features that require unique domain IDs can be partially disabled (for example, NTS). If duplicate domain IDs occur, then redefine the domain ID allocations of your MS systems.

See Also: The DOMAIN, LINK (INMC), and SHOW LINK command descriptions.

SHOW EDB

Function: To display information about currently defined external database connections.

SHOW EDB [={ ALL | *dbname* }] (0)

Use: This command is used to display status information about current external database connections.

Operands: **EDB [={ ALL | *dbname* }]**

If ALL is specified, then status and definition information is displayed for all defined connections. If *dbname* is specified, then status and definition information is displayed only for the connection to the database *dbname*. If no operand is specified, then the system displays the number of external database connections defined.

Examples:

```
SHOW EDB
SHOW EDB=ALL
```

See Also: The SHOW EDBSTAT, SHOW EDBUSTAT, and SHOW EDBUSER command descriptions.

The message help for the messages generated by this command.

SHOW EDBSTAT

Function: To display statistical information about currently defined external database connections.

```
SHOW        EDBSTAT
(0)
```

Use: This command is used to display statistical information for currently defined external database connections.

Operands: **EDBSTAT**

Instructs the system to display statistical information for *all* defined external database connections.

Notes: If the STATS operand was not specified on EDB DEFINE or EDB START command, then all statistics displayed are zero.

See Also: The SHOW EDBSTAT, SHOW EDBUSER, SHOW EDB, and EDB command descriptions.

The message help for the messages generated by this command.

SHOW EDBUSER

Function: Displays information about users and NCL procedures that are currently using SQL for DB2 facilities.

```
SHOW        EDBUSER
(0)
```

Use: This command is used to display information about users and NCL procedures that currently have open NCL connections to the external database.

See Also: The SHOW EDBSTAT, SHOW EDBUSTAT, SHOW EDB, and EDB command descriptions.

The message help for the messages generated by this command.

SHOW EDBUSTAT

Function: Displays *statistical* information about users and NCL procedures that currently have open NCL connections to the external database.

```
SHOW        EDBUSTAT
(0)
```

Use: This command is used to display statistical information about users and NCL procedures that are currently accessing an external database.

Notes: If the STATS operand was not specified on the EDB DEFINE or EDB START command, then all statistics displayed are zero.

See Also: The SHOW EDBSTAT, SHOW EDB, SHOW EDBUSER, and EDB command descriptions.

The message help for the messages generated by this command.

SHOW EDS

Function: Displays current Event Distribution Services PROFILE definitions.

```
SHOW   EDS    [ =ENV | =REGION | ID=nclid | LU=luname |
                    USER=userid | ALL ]
              [ STATS ]
(0)
```

Use: Displays event profile information. The display lists the profile attribute masks for each of the event profiles selected.

Operands:

=ENV
This is the default. Only profiles for the current environment are displayed.

=REGION
Displays all event profiles for the user region.

ID=*nclid*
Displays event profiles for the specified NCLID.

LU=*luname*
Specifies the terminal name of the region for which event profiles are to be displayed.

USER=*userid*
Specifies the user ID of the region for which event profiles are to be displayed.

ALL
Displays all event profiles active in the system.

STATS
Displays event notification counts for event profiles and NCL receivers.

Examples:

```
SHOW EDS ALL
SHOW EDS USER=U01829 STATS
```

Notes: Use this command to find out which NCL procedures will receive notification of events.

See Also: The PROFILE EDS command description.

SHOW EPS

Function: Displays a list of all end points known to this MS region.

```
SHOW EPS      [ DETAIL ]
              [ STATS ]
(0)
```

Use: Displays information and statistics about the SOLVE (SSI) Endpoint Services environment. If no operands are provided, then a summary list of endpoints known to the MS region is displayed.

Operands:

DETAIL
Causes the display to be augmented with additional information about each endpoint.

STATS
Provides additional statistics information.

Examples: `SH EPS`

SHOW EQUATES

Function: Displays the current EQUATE strings available to this user.

SHOW	EQUATES
(0)	

Use: Using the EQUATE command, users can equate a 1 to 8 character string to a string of text up to 130 characters long.

The equated text can then be executed as a command by entry of the equate string, or the equate string can be prefixed to other command text in the command line. In this case, the equated text is appended to the other entered command text prior to execution of the command.

The SHOW EQUATES command can be used to display those equated strings and their associated text.

Notes: Equates are defined as either global or local. Global equates are defined with the EQUATE command and are placed in the initialization procedure executed prior to system start-up. On entry to OCS, all global equates are immediately available to a user. Local equates are defined by the user with the EQUATE command and remain in effect only until the user exits OCS. Global equates are displayed with a TYPE of either GLBL or REPL. Local equates have a TYPE of LOCL. Global equates with a TYPE of GLBL can be overridden by the user defining a local equate. Global equates with a TYPE of REPL have been defined to replace a command and cannot be overridden by a local equate.

When the equated text is displayed with a plus sign at the end of the text, it signifies that a blank will be substituted in place of the plus sign, to ensure that any text from the command line will be separated from the prefixed equate text by a blank.

Frequently used local equates are normally set in an NCL procedure that is executed automatically on entry to OCS.

See Also: The EQUATE command description.

SHOW EXEC

Function: Displays the names of members in a procedure library (OS/VS only).

```
SHOW EXEC   [ ,startname ] [ ,endname ]
            [ ,ID=ddname  ]
            [ ,LIST=FIRST | ALL | FULL ]
(0)
```

Use: The SHOW EXEC command can be used to display the names of members in a procedure library. Optionally a range of members can be displayed.

Operands:

startname

A 1 to 8 character name that specifies the starting point from which names are to be displayed. If this operand is specified, only those names that are higher than or equal to the *startname* are displayed. If no *startname* is specified, then the display starts with the first member in the dataset.

endname

A 1 to 8 character name that specifies the ending point for names that are being displayed. If this operand is specified, only those names that are lower than or equal to the *endname* in collating sequence are displayed. If no *endname* is specified, then the display ends with the last member in the dataset.

ID=*ddname*

An optional DD name which identifies the library from which the member list is to be obtained. If specified, this must be the last operand and must follow any range qualifications.

LIST=FIRST | ALL | FULL

This is an optional operand that specifies the type of list to be displayed. If LIST is specified, it must follow any *startname* and *endname* qualifications.

LIST=FIRST (the default) displays the member names in the first dataset of the concatenation.

LIST=ALL displays the member names in all datasets of the concatenation.

LIST=FULL displays the member names in all datasets of the concatenation. Any member statistics are displayed if possible.

Examples:

```
SHOW EXEC
SHOW EXEC,B,C
SHOW EXEC,S
SHOW EXEC,,B
SHOW EXEC,A,B,ID=TESTPROC
SHOW EXEC,ID=VTAMLST
SHOW EXEC,START1,START99
SHOW EXEC,C,D,LIST=ALL
SHOW EXEC,ID=VTAMLST,LIST=FULL
```

SHOW EXEC

Notes: This command is not applicable in VM/SP or VSE/SP systems.

If specified, the end name must be higher in collating sequence than the start name, or the command is rejected.

A count of the number of members displayed follows the display of member names.

See Also: The EXEC, FSPROC, and START commands.

SHOW FTS

Function: Displays the status of file transmissions.

```
SHOW FTS      [ [ STATUS= ] status ] |
              [ =linkname ] |
              [ INIT=destination ]
(0)
```

Use: This command is used to determine the status of one or more FTS links and transmissions.

For SHOW FTS (or SHOW FTS STATUS=) the status of each FTS destination is provided along with statistics on the active transmissions for each destination. The status returned can be one or more from the list described below.

Operands: **[STATUS=] *status*]**

Used to provide a display of FTS destinations with a specified status. Use of the keyword STATUS= is optional. Possible values for the status are as follows:

ACTIVE
The link to the destination is active and a transmission is in progress.

IDLE
Although the link to the destination is active, there is no current FTS activity.

ENABLED
FTS is enabled for this destination. Transmissions will proceed as long as an FTS initiator is open for the class of the FTS request and contact between systems is established.

DISABLED
FTS is disabled for this destination and no transmissions can proceed until enabled by the operator.

PENDING-CONTACT
The link to this destination has been activated, but has not yet contacted the other MS region.

IN-CONTACT
This MS region is in contact with the target region.

LOST-CONTACT
Although contact was once established with the target system as specified in the LINK command, this contact has been lost. It will automatically recover when the physical path between the systems is re-established.

STOPPED
The link to the target system has been stopped by an operator command.

DRAINING

Transmissions are currently active for this destination, but no new requests are to be initiated. After any current transmissions end, the link returns to a DISABLED state.

=*linkname*

Shows statistics for all active transmissions on the specified link. The block count, character count, and elapsed transmission time are displayed for each inbound and outbound request. A SHOW FTS command shows the same information for all active transmissions.

INIT=*destination*

Provides a display of the current status of the initiator set for the destination. The format of the display is similar to that obtained through the initiator supervision panel.

Notes: The status specified for SHOW FTS STATUS=status can be abbreviated to any length which maintains its unique recognition (for example, LOST-CONTACT can be LO, LOST, and so on).

Examples:

```
SHOW FTS                -* Show status of all FTS destinations
                        -* and transmissions
SHOW FTS=SYDNEY         -* Show status of transmissions to SYDNEY
SHOW FTS DIS            -* Show any disabled links
SHOW FTS STATUS=IDLE    -* Show any idle links
SHOW FTS INIT=PERTH     -* Show status of initiators to PERTH
```

See Also: The SHOW LINK and LINK (INMC) command descriptions.

SHOW ISR

Function: Displays ISR status information.

```
SHOW   ISR     [ =ALL | =class | =(class,class...) ]
               [ ALL | DEFAULT | LINK={ name | name* | * } ]
               [ STATS ]
               [ OPTIONS ]
(0)
```

Use: Displays information about ISR status and definitions.

Operands: If all operands are omitted, all classes are assumed.

=ALL
Displays information about all classes.

=*class* | =(*class,class...*)
Displays information only for one class or a list of classes.

The following are valid classes, although some are valid only if your MS region has the corresponding feature installed:

PPO	PPO message traffic
NEWS	NEWS (CNM) traffic
NTS	NTS traffic
AOM	AOM status, unsolicited traffic, and DOM flow

ALL
Displays information about the default ISR definition and all specific ISR definitions.

DEFAULT
Displays information about the default ISR definition.

LINK={ *name* | *name | * }**
Displays ISR information for all active links and specific ISR definitions with linknames equal to, or starting with, the indicated generic name.

STATS
Displays details about data traffic flow and compression statistics.

OPTIONS
Displays class-specific options as well. For AOM, this includes the local, remote, and effective values (for active links) of all the operands that can be specified on the ISR command for AOM.

SHOW ISR

Examples:

```
SHOW ISR
SHOW ISR=ALL ALL OPTIONS
SHOW ISR=AOM LINK=NEWYORK OPTIONS
```

SHOW ISRSTATS

Function: Displays ISR statistical information.

```
SHOW ISRSTATS    [ =ALL | =class | =(class,class...) ]
                 [ LINK={ name | name* | * } ]
(0)
```

Use: Displays ISR statistics on conversion traffic.

Operands: If all operands are omitted, all classes are assumed.

=ALL
Displays information about all classes.

=class
Displays information only for the listed classes.

LINK={ *name* | *name | *}**
Displays ISR statistics for one or more active links, where the linkname is equal to the specific name entered, or starts with the generic name entered.

Notes: The resultant messages display the date and time when the conversation started on the link shown, the individual message inbound and outbound counts, and the number of bytes transmitted inbound and outbound since the start time.

Examples:

```
SHOW ISRSTATS
SHOW ISRSTATS=PPO LINK=APPL*
```

See Also: The ISR command description.

SHOW LIB

Function: Displays libraries which have been defined to the system.

SHOW LIB (0)	[=*name*]

Use: Panel libraries must be defined using the LIBRARY DEFINE command. The libraries so defined can be replaced using the LIBRARY REPLACE command, or deleted using the LIBRARY DELETE command. The library definitions can be displayed using the SHOW LIB command.

Operands: If no operands are entered, then all current library definitions are displayed.

=name
The display is restricted to the library name specified. If this value ends with an asterisk, then all library definitions that begin with the prefix are displayed.

Notes: SHOW LIB displays information under the following headings:

NAME
The name of the library being displayed.

ID
The logical file ID of the file assigned to this library. The file ID must be assigned to the file when it is opened, using the UDBCTL OPEN command...

EDIT
Indicates whether or not the library can be edited...

STATUS
Indicates the UDB status of the library. Possible values are as follows:

OPEN	The file is currently open and available.
UNDEF	The file is undefined (the UDBCTL command can be used to open this file).
DESCRIPTION	The description assigned to the library by the LIBRARY DEFINE command

Examples:

```
SHOW LIB           -* List all library definitions
SHOW LIB=PANELS    -* List only the library named PANELS
SHOW LIB=P*        -* List all libraries that begin with the
                   -* letter P
```

See Also: The LIBRARY DEFINE, LIBRARY REPLACE, LIBRARY DELETE, UDBCTL, and SHOW PATH command descriptions.

SHOW LICENSE

Function: Displays the active licensing details and expiry dates

```
SHOW LICENSE      [ =ID | =EXPIRY ]

(0)
```

Use: The SHOW LICENSE command can be used to display the system identification information and the current feature expiry dates for active features.

Operands: If no operands are entered, then all active license information is displayed.

=ID
Restricts the display to only the system identification information.

=EXPIRY
Restricts the display to a list of active feature expiry dates.

Examples:

```
SHOW LICENSE=ID
SHOW LICENSE=EXPIRY
```

Notes: Features licensed, but not active due to JCL, INC=, or EXC= parameters, are not displayed in the active feature expiry date list.

Any active feature within thirty (30) days of its expiry date appears highlighted in the expiry date list. If the feature has passed its expiry date, then the caption EXPIRED appears in the *Days Left* column.

SHOW LINK

Function: Displays the status of INMC links. The SHOW LINK command is also used to display the status of all or selected APPC links.

```
SHOW LINK | LINK   ={ STATIC | DYNAMIC | ALL | name* |
                        SINGLE | PARALLEL }
                   [ DETAIL ]
                   [ TYPE={ INMC | APPC } ]
                   [ MODENAME | MODENAME=modename* ]
(0)
```

Use: This command provides a display of the status of all or selected INMC or APPC links to remote systems. Remote systems are always referred to by their linkname, the 1- to 12-character symbolic name allocated when the link is defined.

The status of a link can be ACTIVE, PEND-ACT, or INACTIVE.

- ACTIVE indicates that the link to the remote system is open.
- PEND-ACT indicates that the system is attempting to open the connection to the remote system.
- INACTIVE shows that an earlier LINK command was used to stop the link.

Following the title line displayed by this command is one display line for each VTAM session maintained with the remote system that is being displayed. The status of each session is shown as ACTIVE, PEND-ACT, DISABLED, STARTING, STOPPED, or FAILED.

- ACTIVE means the session is open and working
- PEND-ACT means that the session is not yet open but that the system is trying to open it.
- DISABLED means that an unrecoverable error occurred when trying to open the session and no more attempts will be made for this session. This condition should be brought to the attention of your systems programmer.
- STARTING means that the two systems have made contact and are initializing the INMC sessions. This involves a protocol exchange that includes an authorization check on the product license at both ends.
- STOPPED means that the link has been deactivated by use of the LINK STOP=*linkname* command.
- FAILED means that an unrecoverable error occurred when trying to open the session, and no more attempts will be made for this session. This condition should be brought to the attention of your systems programmer.

The link display also indicates whether the link is static (that is, a permanent link definition invoked by a LINK START command) or dynamic (that is, a temporary link definition that will last only as long as the remote system maintains it).

The SHOW LINK command is also used to display information about APPC links. When the MODENAME operand is not specified, the display shows general status information about APPC links.

This status information includes the following:

- Link status (see the *Notes* section below)
- Link type (parallel or single session)
- Remote LU name
- LU6.2 options supported (for example, mapping)
- Link and conversation level security supported
- The session limit for the link

The MODENAME operand is used to display status information for each mode for all links selected. This includes the current number of contention winner and contention loser sessions active in each mode.

Operands: INMC/NVC operands are as follows:

DETAIL
Displays the access method used and the number of kilobytes transmitted in the session.

LINK
Indicates that this is a link status display of all links.

LINK=STATIC
Displays all permanent links, that is, those links defined using the LINK START command. NVC links are always static. When TYPE=INMC is set, this is the default.

LINK=DYNAMIC
Displays all temporary links, that is, those links that last only as long as the remote system maintains them.

LINK=ALL
Displays all links, that is, all STATIC and all DYNAMIC. When TYPE=APPC is set, this is the default.

LINK=*name**
Displays the status of the single nominated link. If the name entered ends with an asterisk (*), then all links whose linknames start with the nominated prefix are listed.

TYPE=INMC
Indicates that INMC links are to be displayed. This is the default, so it is not necessary to specify this operand for INMC links.

Note
TYPE=INMC also displays information about links defined as TYPE=NVC.

MODENAME | MODENAME=*modename**

The MODENAME operand is used to display status information about each LU6.2 mode for all links selected for display. MODENAME=*modename** is used to restrict the display to a specific modename. If the name entered ends with an asterisk (*), then all modes with names starting with the nominated prefix are displayed.

APPC operands are as follows:

LINK

Indicates that this is a link status display of all links.

LINK=ALL

Indicates that this is a link status display of all links.

LINK=*name**

Displays the status of the single nominated link. If the name entered ends with an asterisk (*), then all links whose linknames start with the nominated prefix are listed.

LINK=SINGLE

Indicates that this is a link status display of all single session links.

LINK=PARALLEL

Indicates that this is a link status display of all parallel session links.

TYPE=APPC

Indicates that APPC links are to be displayed. The default is INMC links, so TYPE=APPC must be entered to display APPC links.

MODENAME | MODENAME=*modename**

The MODENAME operand is used to display status information about each LU6.2 mode for all links selected for display. MODENAME=*modename** is used to restrict the display to a specific modename. If the name entered ends with an asterisk (*), then all modes with names starting with the nominated prefix are displayed.

Examples:

```
SHOW LINK
SHOW LINK=ABC
SHOW LINK=AB*
SHOW LINK=DYNAMIC
SHOW LINK TYPE=APPC
SHOW LINK TYPE=APPC MODE
SHOW LINK=NMP* TYPE=APPC
```

Notes: Links with a status of PEND-ACT can have Remote Operator Facility (ROF) signons queued against them that will complete when the link becomes ACTIVE.

At the end of the display is a TOTALS line. This line always displays the total number of static links defined, the total number of dynamic links defined, and the system limit for dynamic links (set by SYSPARMS DYNLMAX).

The status of an APPC link can be ACTIVE, DRAIN, QUIESCE, FORCE, or INACTIVE. ACTIVE indicates that there is at least one active LU6.2 session to the remote system. A status of DRAIN, QUIESCE, or FORCE indicates that the link has been stopped with that option. The INACTIVE state is a transitory state prior to link activation or after link deactivation.

To obtain more detailed information about a specific mode for an APPC link, use the MODENAME operand. The display shows mode status, session limits, and the current number of contention winner and loser sessions that are active.

See Also: The LINK (INMC), LINK (APPC), and LINK (NVC) command description.

SHOW LOCKS

Function: Displays the status of all locks held within the system.

```
SHOW LOCKS      [ CLASS={ ALL | SYS | NCL | EDIT } |
                  LIST={ ALL | WAIT }  |
                  PNAME=primaryname  |
                  MNAME=minorname |
                  TEXT=text ]
(0)
```

Use: This command provides a display of the status of the various classes of logical locks that can be held within the system. Different classes of locks are held by internal processes (SYS) or by NCL procedures (NCL).

A lock is a logical resource identified by a primary name and, optionally, a minor name. Each resource can be held exclusively or on a shared basis. NCL procedures can alter the lock status during processing, from exclusive to shared or shared to exclusive. The SHOW LOCKS display lists any locks that are waiting to alter their status.

Each resource can also have a text value associated with it. This value, used by NCL lock processing, usually identifies the current holder of the lock and can be interrogated by other NCL procedures that are competing for the same resource.

The use of logical locks by NCL procedures is described in the chapter titled *Synchronizing Access to Resources* in the *Network Control Language User's Guide*.

Operands: When SHOW LOCKS is entered without any other operands, it results in a display of all locks held within the system. Alternatively, any combination of the following operands can be included to provide a selective display of specific resources:

CLASS={ ALL | SYS | NCL | EDIT }

Allows a specific class of lock to be displayed.

NCL locks are taken out by NCL procedures using the &LOCK verb. SYS (system) locks are taken out internally by Management Services. EDIT locks are taken out by the panel editor when a user is editing a panel definition.

LIST={ ALL | WAIT }

LIST=ALL displays all locks in the system.

LIST=WAIT lists only those locks for which processes are waiting, including those that are waiting to alter their status from exclusive to shared or from shared to exclusive.

PNAME=*primaryname*

Displays the lock(s) held which have a primary resource name as specified. The *primaryname* can include wildcard (asterisk) characters in any position to provide a generic display.

MNAME=*minorname*

Displays the lock(s) held which have a minor resource name as specified. If this operand is used with the PNAME operand, then only the lock with the specific, fully qualified (PNAME/MNAME) resource name is displayed. If this operand is used without the PNAME operand, then all locks with the specified minor resource name are displayed. The *minorname* can contain wildcard (asterisk) characters in any position to provide a generic display.

TEXT=*text*

All locks with a TEXT value as specified are displayed. The text specified can contain wildcard characters (*) in any positions to provide a generic display. If the text value contains imbedded blanks, then enclose the text in quotes.

Examples:

```
SHOW LOCKS
SHOW LOCKS PNAME=FILELOCK
SHOW LOCKS PNAME=F* MNAME=G* TEXT='USER01 PROC0008'
```

Notes:

The SHOW LOCKS command can be used to identify conflicts between NCL procedures that are waiting for resources.

Following is an example of the SHOW LOCKS display.

```
N18704 NCL CLASS LOCKS.
N18705 PRIMARY NAME RESOURCE AAAA
N18706    LOCK-ID REQ  LOCK-TEXT
N18707          9 SHR  USER01  RLTEMP2 12.44.30
N18709    LOCK WAITING REQUESTS
N18707          6 SHR  USER02  CNTEST1 12.43.07
N18710    LOCK ALTERATION REQUESTS
N18707         10 SHR  USER01  RLTEMP2 12.44.31
N13503 *END*
```

See Also:

The PURGE command description.

The &LOCK verb description in the *Network Control Language Reference*.

SHOW LOGS

Function: Displays the current status of the system activity logs.

```
SHOW        LOGS
(0)
```

Use: The system continuously maintains a log of all significant events. Using the LOGSWAP command means that the current log can be closed and released and logging commenced on a new log if one is available.

The SHOW LOGS command displays the start and stop time for each of the defined logs, the status of the log (for example, AVAIL, INUSE, DEALLOC, or SW/IOER), the number of pages written to the log, and the reason for swapping if the log has been swapped.

If the log was swapped by the LOGSWAP command, then the user ID of the operator entering the command is displayed.

Notes: LOGSWAP causes cross-reference information to be printed at the start and end of each log to allow easy correlation of log listings.

See Also: The LOGSWAP command description.

SHOW LSR

Function: Displays LSR status information.

```
SHOW LSR      [ ={ STATISTICS | DEFAULT | CURRENT | PENDING } ]
(0)
```

Use: Displays information about the active VSAM LSR pools, or about LSR pool definitions.

Operands:

STATISTICS

Displays VSAM statistics about the active LSR pools. If no datasets are currently open with the LSR attribute, then no statistics are displayed, as there is no active pool.

SHOW LSR=STATISTICS has no function in the VSE/SP and VM/GCS environment, as VSE/VSAM does not maintain LSR statistics.

In MVS/XA and MVS/ESA, *all* active pools are displayed.

DEFAULT

Displays the default LSR definition.

CURRENT

Displays the current LSR definitions, that is, the definitions that are in use if any datasets are opened with the LSR attribute.

PENDING

Displays the pending LSR definitions, that is, the definitions that have been set up by the LSR REPLACE/UPDATE command. These are the definitions that will become current the next time a new LSR pool is built.

Examples:

```
SHOW LSR
SHOW LSR=PENDING
```

See Also: The LSRPOOL command description.

SHOW LUTRACE

Function: Displays the status of all LUTRACE requests.

```
SHOW        LUTRACE
(0)
```

Use: The SHOW LUTRACE command can be used to list all current LUTRACE requests. The information returned by this command includes the user ID of the user that initiated the request and the date and time when the request was made. In addition, the current status (either ACTIVE or PENDING) is reflected, as well as a count of the number of sends/receives, both on a session level and as a total across all sessions.

Notes: The session send/receive counts are not reset until a new session has begun, thus allowing the operator to review these statistics after the session has ended.

See Also: The LUTRACE command description.

SHOW MAI

Function: Displays information about MAI sessions for this user or all users.

```
SHOW MAI      [ ={ OWN | ALL | DISC | LOCK | string } ]
                LIST={ SUMMARY | ALL }
(0)
```

Use: Displays the status of MAI sessions currently held for either the entering user or all users. Both MAI-OC and MAI-FS sessions are included.

The display gives the session ID which is used to identify each session, the status of the session, the LU name in use for the session, the New Line character, the ACB share status, the MS window from which the MAI session is held, and other SNA-related information.

Operands:

=OWN
Displays the entering user's MAI sessions. OWN is the default.

=ALL
Displays all MAI sessions. The display includes the user IDs of users owning those sessions.

=DISC
Displays MAI sessions owned by disconnected regions.

=LOCK
Displays MAI sessions owned by locked regions.

=string
Provides a reduced display of MAI sessions. The string entered is used in a generic comparison against the owning region name and the names displayed in the APPL, UID, and NODE columns of a SHOW MAI=ALL command. Where the generic input string matches the leading portion of any of these names for a particular session, the usual message is formatted and displayed.

LIST={ SUMMARY | ALL }
Produces a second detail line containing the name of the DEFLOGON and the number of bytes currently used for session buffers.

Examples:

```
SHOW MAI=OWN
SHOW MAI
SHOW MAI=TSO
SHOW MAI=ALL
```

Notes: SNA-related information in the display is explained in the *SOLVE:Access Implementation and Administration Guide*.

A line of the display that corresponds to an MAI-FS session that has output queued is displayed in high intensity. See the online message help for N50802 for details of the information supplied by this message.

SHOW MAI

See Also: The MAILOGON, MAIINT, MAIDISC, and MAISEND command descriptions.

SHOW MAISTAT

Function: Displays information about the status of MAI subtasks.

```
SHOW MAISTAT      [=ACTIVITY ]
```

Use: Displays information about the status of operating system subtasks used by MAI.

Operands: **=ACTIVITY**

Displays the current processing of MAI-FS subtasks.

Notes: MAI-FS uses two types of subtask.

- A *service subtask* is used to open and close VTAM ACBs and to call any installation-supplied MAI user exits.
- A *session subtask* is used to conduct the session between MAI and the target application.

There can be one service subtask, and from one to sixteen session subtasks. Each session subtask can handle any number of sessions, and MAI spreads the workload amongst the subtasks available. Subtasks are detached when not required.

The use of multiple subtasks can improve performance on multiprocessor machines, and can also provide greater overlap of operations on high-paging systems. The SYSPARMS MAIFSTMX command controls the number of subtasks utilized by MAI-FS.

MAI-OC utilizes an operating system subtask for each MAI-OC session.

The response to a SHOW MAISTAT command is a heading line followed by one or more lines of information for each type of subtask. Finally, a summary line is produced. Following is the meaning of each of the fields displayed:

TASK

The type of task. MAIN indicates the Management Services main task. The main task performs some MAI-related functions. SERV indicates the MAI-FS service subtask. A number from 1 to 16 indicates an MAI-FS session subtask.

STATUS

Contains the current status of the subtask. AVAILABLE indicates that the task is available for new work. STOPPED indicates that the task cannot be used for new work. This status comes about if the SYSPARMS MAIFSTMX command is used to *reduce* the number of tasks that MAI-FS can utilize. ATTACHED indicates that the task is currently attached, and therefore has work allocated to it. DETACHED indicates that the task has been detached. ENDING indicates that the task has no currently allocated work, and is in the process of ending.

#SESS

In the case of the MAIN task, this field contains the current number of MAI sessions (both MAI-FS and MAI-OC). For MAI-FS, this can include sessions that have actually terminated, but the user has not yet acknowledged the Session Ended panel.

In the case of the SERV task, this field contains the number of VTAM ACBs that the task currently has open.

In the case of a session task, this field contains the number of sessions being handled by that task.

MAX-QD

The MAIN and session tasks handle work via a queue. This field contains the maximum depth that the queue has ever reached. That is, when the task came to service its work queue, it had to process this many work requests before going into a wait condition, or, in the case of the MAIN task, before handling other work.

AVE-QD

The MAIN and session tasks handle work via a queue. This field contains the average depth of the task's queue. That is, when the task came to service its work queue, it had to process on average this many work requests before going into a wait condition, or, in the case of the MAIN task, before handling other work.

If the average queue depth is consistently high, then increasing the number of MAI subtasks can improve performance. Use the SYSPARMS MAIFSTMX, described in the *SOLVE:Access Implementation and Administration Guide*.

#GM

The MAIN and session tasks utilize storage pools to pass work requests between various components. Utilization of storage pools eliminates the need for GETMAIN or GETVIS. This field contains the number of times the task's storage pool has been exhausted, thereby forcing use of GETMAIN or GETVIS to obtain more storage. This should occur only in times of increased activity, for instance during a network outage where many sessions are being terminated concurrently.

The final summary line of the display gives the current number of MAI sessions (both MAI-FS and MAI-OC), the maximum number of MAI sessions and the number of jumps that have been performed by MAI-FS users.

Some of the figures presented in the display can help to tune MAI-FS. For instance, if the average queue depths are consistently high, it is possible that starting extra session tasks could provide increased overlap of operations.

The response to a SHOW MAISTAT=ACTIVITY command is a heading line followed by a detail line for each MAI-FS subtask. Following is the meaning of each of the fields displayed:

TASK
Identifies the MAI task. SERV is the service subtask responsible for ACB open/close and for communicating with the MAIEX02 exit. A number in the range 1 to 16 indicates a session subtask responsible for communication with the applications.

ACTIVITY
The current function being performed by the task. A value of IDLE indicates the task is waiting for work. Other values are associated with a particular session and fields LUNAME, USERID, SESSID, and INTERVAL are set.

LUNAME
The LU representing the virtual terminal.

USERID
The user ID owning the session.

SESSID
The MAI session ID.

INTERVAL
The time interval since the activity started.

See Also: The SYSPARMS MAIFSTMX operand in the *SOLVE:Access Implementation and Administration Guide*.

SHOW MAP

Function: Displays information about Mapping Services maps defined in the system.

```
SHOW MAP      [ =name ]
(0)
```

Use: This command provides a brief description of Mapping Services maps currently defined in the system. These maps can be referenced from NCL for use in MDOs. Each line of display contains the following information about a map:

Map name
Name of the map as defined. This is the name used to reference the map from NCL.

Map key
The key associated with the map. This key is embedded in the header of any MDO to which this map is attached.

Use count
The number of Mapped Data Objects to which this map is currently attached.

Status Loaded/Error
Indicates whether the map has been loaded or whether it is in error status.

Operands: If SHOW MAP is entered without any other operands a display of all maps defined in the system is displayed.

=name
Only information about the particular map named is required.

Examples:

```
SHOW MAP
SHOW MAP=$CNM
```

Notes: The maps shown with the SHOW MAP command are defined in the Mapping Services Data Dictionary. New maps can be defined using the Mapping Services Data Dictionary maintenance facility. The maps can be used in NCL by attaching them to a Mapped Data Object.

See Also: SYSPARMS MAPLOAD and SYSPARMS MAPDEL operands in the *Management Services Administrator Guide*.
Mapping Services chapters in the *Network Control Language User's Guide*.

SHOW MSGQ

Function: Displays message queue depths.

```
SHOW MSGQ      [ ALL ] |
               [ [ OCS ] [ NCL ] [ BG ] [ SYS ] [ EMPTY ] ]
(0)
```

Use: The SHOW MSGQ command displays the depth of various system and message queues. If no operands are entered, all non-empty queues are displayed—the equivalent of entering SHOW MSGQ OCS NCL BG SYS.

Operands:

ALL
Indicates that all queues, including all empty queues, are to be displayed. This operand is mutually exclusive with all other operands.

OCS
Indicates that OCS environment message queues are to be displayed. This includes real LU-2 OCS sessions, LU-1 OCS, signed-on system consoles, and ROF sessions.

NCL
Indicates that NCL &INTCMD processing environment queues are to be displayed.

BG
Indicates that the virtual OCS queues for background NCL regions are to be displayed (for example BSYS or BMON). This does not mean the input queues for AOMPROC, for example.

SYS
Indicates that various system input queues are to be displayed, including LOG, CNMPROC, PPOPROC, and AOMPROCs.

EMPTY
Indicates that empty queues are also to be displayed. (These are normally suppressed.)

Notes: Large queue depths could indicate OCS sessions that are in HOLD mode.

The system queue depths of internal processes CNM, PPO and AOM, which issue the specialized &*xxxx*READ verbs are not displayed by this command.

See Also: The SHOW OCS and SHOW AOMSTAT command descriptions.

SHOW NCL

Function: Displays the status of NCL processes executing in the NCL processing region for a specific user's OCS windows.

```
SHOW NCL     [ =ENV | =REGION | =name | =ALL | ID=nclid ]
             [ PROC=procname ]
             [ PRTY=priority ]
             [ PG=perf-gp ]
(0/2)
```

Use: The SHOW NCL command displays the status of active NCL processes.

The display lists all procedures, both serial (that is, invoked via EXEC or FSPROC) or concurrent (that is, invoked by START). The display shows the following information for each listed procedure:

- The current (nested) procedure name and the nesting level.
- How the procedure was invoked (EXEC, FSPROC or START).
- The user ID for which the procedure is executing.
- The procedure's NCL performance group.
- The NCL statement currently being executed by the procedure.

Operands:

=ENV

The default, ENV indicates that the command is to list only those processes executing in the NCL processing environment associated with the OCS window from which the command is entered. If the command is issued from an NCL process it is restricted to the process's own NCL processing environment.

=REGION

This operand lists all processes executing in the issuing user's NCL processing region, that is, processes executing under either window and not limited to OCS windows.

=name

This operand identifies a specific NCL region to be displayed. Requires authority level 2 or as set by your installation. Enter either the user ID or the terminal name associated with the region.

=ALL

This operand is restricted to users with the appropriate command authority level, default is 2. ALL displays all NCL processes executing in the system.

ID=*nclid*

Specifies a particular process. If the process is executing outside the issuer's NCL region this requires authority level 2 or as set by your installation. ID= is mutually exclusive with NCL= operands and must be coded SHOW NCL ID=*nnnnnn*.

PROC=*procname*
This operand limits the display to occurrences of the nominated procedure name.

PRTY=*priority*
This operand limits the display to processes executing at the nominated dispatching priority (range 0 to 3).

PG=*perf-gp*
This operand limits the display to processes executing within the nominated NCL performance group (range 1 to 4).

Examples:

```
SHOW NCL
SHOW NCL=REGION
SHOW NCL=TERM123
SHOW NCL=USER5
SHOW NCL=ALL PROC=$LOGBROW
SHOW NCL ID=337
SHOW NCL=ALL PG=2 PRTY=2
SHOW NCL=ALL PG=2 PRTY=2 PROC=$NSASYNC
```

Notes:

Any procedure listed by the SHOW NCL command can be terminated by the FLUSH command, except for MSGPROC procedures. If the FLUSH or END command is to be used to end a process that is outside the user's NCL region the user must have the appropriate command authority.

Procedures executing under an OCS window are flushed if the user terminates the window or exits OCS mode.

See Also:

The GO, END, INTQUE, and FLUSH command descriptions.
The PROFILE MSGPROC operand.

SHOW NCLGLBL

Function: Displays the names of any defined NCL global variables, and optionally their contents.

SHOW NCLGLBL [=*name*] (2)

Use: The SHOW NCLGLBL command can be used to determine the names of all defined NCL global variables. By using the SHOW NCLGLBL=*name* command the current length and value of each global variable beginning with the entered name can be determined.

The information returned is simply a list of all global variables which are currently defined (that is, have a non-null value). The full variable name without the ampersand (&) is displayed.

For the SHOW NCLGLBL=*name* command, *name* refers only to the variable name following the global variable prefix. The prefix itself (by default GLBL) is not entered. The name can be an entire variable name or simply the leading part of the name. For each global variable found with the same leading name portion, the following information is displayed:

NAME
The full name of the variable (including the global variable prefix but excluding the leading ampersand).

LEN
The length of the current variable contents.

TEXT
The actual contents of the variable.

Operands: If SHOW NCLGLBL is entered without any other operands, a full list of NCL global variables currently defined is displayed.

NCLGLBL=*name*
Lists all the NCL global variables presently defined whose names begin with the prefix specified. *name* refers only to the variable name following the global variable prefix. The prefix itself (by default GLBL) is not entered. The display also shows the variable length and a maximum of 230 characters of the content of the variable.

Examples:

```
SHOW NCLGLBL          -* Displays global variable names only
SHOW NCLGLBL=CICS     -* Displays contents of those starting
                      -* with the global variable prefix
                      -* followed by CICS
SHOW NCLGLBL=         -* Displays contents of all global
                      -* variables
```

Notes: Using the SHOW NCLGLBL form of this command requires a higher authority level than that required for the SHOW command itself.

See Also: The SYSPARMS NCLGLBL command description.

SHOW NCLSTAT

Function: Displays the status of NCL procedures currently in storage.

```
SHOW NCLSTAT      [ =prefix ]
(0)
```

Use: The SHOW NCLSTAT command can be used to display the names of NCL procedures that have been preloaded using the SYSPARMS PRELOAD command, or those available for sharing via the SYSPARMS NCLPRSHR command. Their current status and some useful statistics are also displayed.

The SHOW NCLSTAT command provides the following information:

STATUS :
The following STATUS codes are possible:

PRELOAD
The procedure has been successfully preloaded and is available for use. The USERS field shows the number of users currently executing this copy of the procedure.

LOADING
The procedure is still in the process of being loaded and is not yet available for use.

P-UNLOAD
An UNLOAD request has been entered and accepted but is still pending as there are active users of this copy of the procedure. No new users are able to use this copy and another copy can be preloaded if required. The unload will complete when there are no further users of the procedure. The USERS field shows the number of users that are still executing this copy of the procedure.

AUTOSHR
This indicates that the particular procedure will automatically be shared if more than one request for it is received. This means that the procedure copy in storage is used which eliminates the need to re-fetch the procedure from the procedure dataset. Preloaded procedures are always autoshared. If the SYSPARMS NCLPRSHR option is set to YES, or a numeric value, all other procedures can also be autoshared, otherwise a new copy is loaded each time a procedure is requested that is not preloaded.

RETAINED
When SYSPARMS NCLPRSHR is set to a numeric value then, in addition to providing procedure autosharing, this number of non-preloaded procedures will be retained in storage in expectation of future reuse. These procedures show the RETAINED status with a zero use count.

CMDREPL
The procedure will be invoked instead of the native MS command of the same name. This occurs when a SYSPARMS CMDREPL=*command name* command has been issued to replace a command with a procedure of the same name.

PRIVATE
The procedure is a system procedure such as LOGPROC or PPOPROC. It will not be shared of retained.

NCLTEST
The procedure has been loaded for testing execution as controlled by PROFILE NCLTEST. It will not be shared or reused and will be deleted when execution is completed.

USERS:
The current usage count for a preloaded or autoshareable procedure. This is the number of concurrent active executions of the procedure.

REFCNT:
The referenced count is a total of the number of requests for a preloaded or autoshareable procedure that were satisfied from an in storage copy of the procedure, thus avoiding the need for physical disk I/O.

STMTS:
This is the number of statement lines (excluding comments) found in the procedure.

NCLBUF:
This is the number of NCL buffers (NCBF) required to contain the procedure after loading. The size and total allocation of NCL buffers can be determined from a SHOW BUFF command.

LIBRARY:
The DD name (z/OS or OS/390), sub-library (DOS) or filetype (VM/SP) from which the procedure was loaded.

AGE
For retained procedures an aged reference count, used to manage deletion of the retained procedure from the in storage retention queue. A high age-use count indicates a frequently referenced procedure which is more likely to be retained in storage. A dash (-) indicates that the procedure is not currently retained.

Operands:

NCLSTAT
Displays the full list of in-storage procedures.

NCLSTAT=*prefix*
Displays a selective list of in-storage procedures based on the generic prefix supplied. Statistics are displayed only for procedures commencing with a matching prefix.

Examples:

```
SHOW NCLSTAT
SHOW NCLSTAT=$IM
```

Notes: The SYSPARMS command UNLOAD operand can be used to unload a preloaded or autoshared procedure.

The SYSPARMS command NCLPRSHR operand can be used to request procedure autosharing without preloading.

Use of the SYSPARMS command might be restricted in your installation.

See Also: The operand descriptions for the commands UNLOAD, PRELOAD, and NCLPRSHR in the *Management Services Administrator Guide*.

SHOW NCLVARS

Function: Displays information on variables in use by NCL procedures.

```
SHOW NCLVARS    [ ={ procname | * | ? } ]
                [ ID=nclid ]
                [ LIST={ ALL | SUMMARY } ]
(0/4)
```

Use: The SHOW NCLVARS command can be used to display information on variables in use by NCL procedures. Information can be supplied in summary form or as a complete list of variables.

Information can be requested for all procedures, the current procedure or a named procedure.

Operands:

NCLVARS={ *procname* | * | ? }

Displays NCL variable usage information. If an asterisk (*) is entered, the current nesting level is processed. If a question mark (?) is entered, all nesting levels are processed. Alternatively, the levels that are an invocation of *procname* are processed.

ID=*nclid*

This optional operand specifies a particular NCL process to be interrogated. When sufficiently authorized, processes in other processing regions can be interrogated. If this operand is omitted, the first NCL process executing in the issuing environment is displayed.

LIST={ ALL | SUMMARY }

This optional operand specifies the type of information to be displayed. When omitted, a complete list of variables, their names, attributes and up to 12 characters of content is displayed. ALL returns the same output with an additional message for each variable, that is, the variable's entire contents. SUMMARY returns one line for each nesting level processed and contains a count of variables, total characters of data and an average length.

Examples:

```
SHOW NCLVARS=* ID=&ZNCLID
                   -* List variables for the current procedure
SHOW NCLVARS=? LIST=SUMMARY ID=2234
                   -* Summarize variable usage for all nesting
                   -* levels of process 2234
```

Notes: The SHOW NCLVARS command can be used as a diagnostic tool to reduce the storage use of high volume NCL processes.

The output from the SHOW NCLVARS command is returned to the OCS screen but does not appear in the activity log.

See Also: The SHOW NCL command description.

SHOW NDB

Function: Displays information about currently active or halted $NDB databases.

SHOW NDB [={ ALL | *dbname* | *dbnameprefix** }] (0)

Use: This variant of the SHOW command can be used to display information about $NDB databases that are currently active, started, stopping, and/or locked. If no operands are entered, a summary message listing the number of databases in each status will be displayed.

Operands:

=ALL

This operand causes a display of the status of each NDB that is currently active, started, stopping, or locked. A separate message is displayed for each database, with information about number of users, number of subthreads, and so on.

=dbname

This operand displays the same information as SHOW NDB=ALL, but only for the database named *dbname*.

=dbnameprefix*

This operand represents a generic prefix. It causes all NDBs whose names begin with the prefix to be displayed.

Examples:

```
SHOW NDB=MYNDB
```

Displays information about database MYNDB.

```
SHOW NDB=TEST*
```

Displays information about every database whose name begins with TEST. For example, TEST1, TEST2, TESTABC and so on.

Notes: The SHOW NDB=ALL command also shows the number of database requests executed since starting. Thus it can be useful to track the progress of a large bulk-load or update.

See Also: The SHOW NDBUSER command description.

SHOW NDBUSER

Function: Displays a list of all users currently signed on to $NDB databases.

```
SHOW        NDBUSER
(0)
```

Use: This variant of the SHOW command can be used to display a list of all users of $NDB databases. The list shows the database name, the user ID (or EASINET, if the NCL procedure is running under the EASINET feature), the terminal name, and NCL ID.

Notes: This command is useful when you must STOP or LOCK an NDB, to determine the impact of the command.

See Also: The SHOW NDB command description.

SHOW NETMCNTL

Function: Displays the status of NETM connectivity.

```
SHOW NETMCNTL      [=xxxxxxxx]
(0)
```

Use: The SHOW NETMCNTL command is used to display the status of NETM/OP Connection Path entries defined by the NETMCNTL ADD= command. The display indicates the entries that are defined for the receipt of unsolicited messages and for sending commands to NETM/OP.

For entries used for unsolicited message receipt, the display shows the category of messages being received.

For entries used for sending commands to NETM/OP, the display shows which entries are currently processing commands.

Operands:

NETMCNTL
Display the status of all Connection Path Entries defined using the NETMCNTL ADD= command.

NETMCNTL=*xxxxxxxx*
Display the status of Connection Path Entry *xxxxxxxx*.

Examples:

```
SHOW NETMCNTL
```

See Also: NETMCNTL command description.

SHOW NETSPY

Function: Lists the currently defined NetSpy connections and their status.

```
SHOW NETSPY      [ CONNECTIONS | CON ]
(0)
```

Use: This command provides general information on the status of NetSpy connections.

You can enter this command as either SHOW NETSPY or SHOW NSP.

Operands: **CONNECTIONS | CON**

Displays the currently defined NetSpy connections and their status. You can omit this operand.

Examples:

```
SHOW NETSPY CONNECTIONS
SHOW NETSPY CON
SHOW NETSPY
SHOW NSP CONNECTIONS
SHOW NSP CON
SHOW NSP
```

See Also: The NSPCONN DEFINE, NSPCONN DELETE, NSPCONN START, NSPCONN STOP, SHOW NETSPY DEFINITIONS, and SHOW NETSPY USERS command descriptions.

SHOW NETSPY DEFINITIONS

Function: Displays the currently defined NetSpy connections and their owners.

```
SHOW NETSPY      DEFINITIONS | DEF
(0)
```

Use: This command provides general information on the status of NetSpy connections.

You can enter this command as either SHOW NETSPY or SHOW NSP.

Operands: **DEFINITIONS | DEF**
Displays the currently defined NetSpy connections and their status.

Examples:

```
SHOW NETSPY DEFINITIONS
SHOW NETSPY DEF
SHOW NSP DEFINITIONS
SHOW NSP DEF
```

See Also: The NSPCONN DEFINE, NSPCONN DELETE, NSPCONN START, NSPCONN STOP, SHOW NETSPY, and SHOW NETSPY USERS command descriptions.

SHOW NETSPY USERS

Function: Lists the current users of the NetSpy interface.

```
SHOW NETSPY      USERS
(0)
```

Use: This command shows details of the current users of the NetSpy interface.

You can enter this command as either SHOW NETSPY USERS or SHOW NSP USERS.

Operands: **USERS**
Displays details (such as NCLID and user ID) of the current users of your NetSpy interface.

Examples:

```
SHOW NETSPY USERS
SHOW NSP USERS
```

See Also: The NSPCONN DEFINE, NSPCONN DELETE, NSPCONN START, NSPCONN STOP, SHOW NETSPY, and SHOW NETSPY DEFINITIONS command descriptions.

SHOW NPF

Function: Displays user's Network Partitioning Facility (NPF) resource tables and their status.

SHOW NPF [=*resource-name*] (0)

Use: This command provides general information on the status of the user's RESOURCE table definitions. The display lists the tables that can be searched by the system when determining the user's access to system and user-defined resources. For example, the $NMCMD resource tables determine which resources an operator can issue a VTAM command against.

The current and initial status of each table are displayed, and an indication is given if a table has definition or syntax errors.

You can list the contents of an individual table by using the SHOW NPTAB command.

Operands: ***=resource-name***

Displays user's own NPF tables with the specified names and their status.

Examples:

```
SHOW NPF=SOLVSYS*
```

Notes: Use this command to find out which tables are being used for your user ID and whether any of those tables contain errors.

You can specify a generic resource group name to scope the command display; for example, SHOW NPF=$NM*.

The NPTAB ACT or INACT commands can be used to change the status of RESOURCE tables.

See Also: The SHOW NPTAB and NPTAB command descriptions.

SHOW NPTAB

Function: Displays status of all or individual Network Partitioning Facility (NPF) resource tables.

SHOW NPTAB [=*tablename*] (0)

Use: Used without the *tablename* operand, this command provides a list of all NPF resource tables defined to the system, with an error status if applicable. This version of the command can be used to check whether any tables contain errors.

Used with the *tablename* operand, this command provides a display of the network resources defined within the selected table and, where necessary, error messages identifying records within the table that contain syntax errors. This version of the command is used to pinpoint definition errors and to check the resources defined within individual NPF tables.

Operands:

Note

The absence of operands indicates that a summary list of defined ATF table names is requested.

=tablename

Displays contents of the selected NPF table and identifies records within the table that contain syntax errors.

Examples:

```
SHOW NPTAB
SHOW NPTAB=TABLE22
```

Notes: When the contents of an individual NPF table are listed, the resource names can contain wildcard characters. An asterisk (*) in any position means that zero or more characters are matched at that position. A question mark (?) in any position means that one character is matched at that position.

See Also: The SHOW NPF command description.

SHOW NRD

Function: Displays the current queue of Non-Roll Delete Messages.

```
SHOW        NRD
(0)
```

Use: Displays each NRD message currently on the NRD queue. The display shows the following:

- Size of the queue
- DOMID of each message
- Time and date added
- Partial message text
- User ID if queued to a specific user
- NCLID if sourced by &WRITE

Notes: This command can be used to identify all NRDs in the system, not just those that an individual is entitled to view. It is useful for analysis of the current NRD queue by an NCL procedure (via the &INTCMD) and if it becomes necessary, to delete NRDs through the PURGE NRD command.

See Also: The PURGE command description.

Function: Display NTS resource or session information.

```
SHOW NTS     [ SA=sa [ EL=el ] ]
             [ NAME=[ netid1. ] name1 [ / [ netid2. ] name2 ]
                  [ SESS [={ PRI | SEC } ] ]
                  [ PCID=nnnn ]
                  [ CLASS | ACCT | TRACE | PARMS ] ]
             [ NETID=netid ]
             [ ER [=n] ]
             [ VR [=n] ]
(0)
```

Use: The SHOW command is an MS command but NTS recognizes the NTS operand and displays NTS resource or session information.

Operands:

Note

Entering SHOW NTS without any other operands provides a summary of current session awareness status. Further operands can be specified to limit the display.

SA=*sa*

Used to provide a list of those resources known to be active within a particular subarea. The *sa* value must be a hexadecimal subarea number.

EL=*el*

Used to specify a particular resource within a subarea by hexadecimal element address.

NAME=[*netid*1.] *name*1 [/ [*netid*2.] *name*2]

Used to specify a particular or generic resource name, or a particular session for which information is requested. A generic name is specified by a trailing asterisk and cannot be network qualified.

Specification of a subarea number *sa* can further restrict the scope for a generic name. If no other operands are supplied a resource list display is generated. If two names are specified then a session display is assumed (see SESS below). For either two names, or a single non-generic name, each name can be network qualified. If not then the native network in which NTS is operating is assumed. If a generic name is specified then at least one character must be provided, that is, NAME=* is invalid.

SESS [={ PRI | SEC }]

Used to request session information for the selected resource(s). Information concerning all sessions (SESS), only those sessions where the resource is primary (SESS=PRI), or those sessions where the resource is secondary (SESS=SEC) can be obtained.

PCID=*nnnn*

nnnn is the procedure correlation identifier assigned to the session by VTAM. This operand is used in conjunction with the SESS operand to specify a unique session.

CLASS

Used in conjunction with the SESS operand to display class information for the selected sessions. This information comes from the session records and reflects the class attributes taken when the session awareness arrived. It does not necessarily reflect the current state of the DEFCLASS definitions.

ACCT

Used in conjunction with the SESS operand to display any accounting information for the selected sessions.

TRACE

Used in conjunction with the SESS operand to display the latest trace information for the selected session (if available). Trace information is interpreted and formatted into messages.

PARMS

Used in conjunction with the SESS operand to display the session start parameter information (for example, BIND, ACTLU and so on) for the selected session (if available). This information is interpreted and formatted into messages.

NETID=*netid*

In an NTS Single Image environment, this operand specifies the name of the network that is the source network of the resource or session data. If omitted, the network in which NTS is operating is assumed.

ER [=*n*]

Used to provide a summary of active explicit routes known to NTS. If ER is specified, all explicit routes are displayed. If ER=*n* is specified, all explicit routes with *n* as the route number are displayed. The displays show the OSA (origin subarea) name and address, DSA (destination subarea) name and address, explicit route number and the number of active sessions on the explicit route.

VR [=*n*]

Used to provide a summary of active virtual routes known to NTS. If VR is specified, all explicit routes are displayed. If VR=*n* is specified, all virtual routes with *n* as the route number are displayed. The displays show the OSA (origin subarea) name and address, DSA (destination subarea) name and address, virtual route number, the route numbers of the explicit routes onto which the virtual route is mapped, and the number of active sessions on the virtual route.

Examples:

```
SHOW NTS
SHOW NTS SA=1B EL=17A
SHOW NTS NAME=LU23
SHOW NTS NAME=NET001.LU23/CICS ACCT
SHOW NTS ER=0
```

Notes:

The SHOW NTS command with no other operands can be used to provide a summary status display of NTS information. However, once other operands are entered the information displayed becomes more specific, being either a resource list display, or session detail display.

For a resource list display the selected resources are displayed under the appropriate subarea summary line. The resource name, type and element address within the subarea are formatted.

For a session detail display there are various formats. If no session options are used then the session partner names, network addresses, session start time and the number of currently queued trace records are displayed. If the CLASS operand is entered then the SAW class details held with the session and extracted at session start time are shown. This includes the SAW class name, COS name, ER, Reverse ER, VR and TP numbers, the session logging and accounting options, and the initial and final trace queue depths. If the ACCT operand is entered then the accounting start time along with the input and output byte counts, PIU counts and PIU average sizes are displayed. If the TRACE operand is entered then all trace data for the session is formatted and displayed at the user's terminal. Similarly, if the PARMS operand is used only that trace data which constitutes the session start parameters (for example, BIND, ACTLU, ACTPU, or ACTCDRM) is displayed.

See Also:

The DEFCLASS and STRACE command descriptions.

SHOW NTSDBMOD

Function: Provides a summary of the currently executing NTSDBMOD and SHOW SKEEP processes.

```
SHOW        NTSDBMOD
(0)
```

Use: The SHOW command is an MS command, but the NTSDBMOD operand is recognized by NTS as a request for information pertaining to the currently executing invocations of the NTSDBMOD and SHOW SKEEP commands.

Operands: NTSDBMOD

Notes: The SHOW NTSDBMOD command produces a heading line followed by a line for each currently executing NTSDBMOD or SHOW SKEEP process. This line contains information relating the user ID of the issuer of the process, the type of command issued, the identifier associated with this process and the progress of the process up to the time the SHOW NTSDBMOD command was issued.

The information is displayed under the following headings:

USERID
Displays the user ID of the issuer of this NTSDBMOD or SHOW SKEEP process.

COMMAND
Displays the type of process, that is, NTSDBMOD or SHOW SKEEP.

ID
Displays the identifier associated with this process. This number is used as the value *nn* for the NTSDBMOD CANCEL=*nn* command to terminate this process.

READ-CNT
Displays the number of records in the NTS Database read by this process up to the time that the SHOW NTSDBMOD command was issued.

See Also: The NTSDBMOD and SHOW SKEEP command descriptions.

SHOW NTSSTATS

Function: Display NTS statistical information.

```
SHOW        NTSSTATS
(0)
```

Use: The SHOW command is an MS command, but the NTSSTATS operand is recognized by NTS as a request for NTS statistical information.

NTSSTATS displays statistics on the number of session awareness and session trace buffers received from VTAM since session awareness was started, the number of entries processed from these buffers, and the statistics concerning the logging of session data to the NTS database on session end.

Notes: The SHOW NTSSTATS command produces a heading line followed by two lines of detail messages. The first detail line contains the word 'COUNTS' on the left hand side and contains the interval counts of the NTS statistics since the last SHOW NTSSTATS command was issued. The second line contains the word TOTALS on the left hand side and contains the total counts since session awareness was last started (for saw and trace buffer information) or since logging was last resumed (for logging information). After the command has been executed the interval counts are added to the total counts, and then the interval counts reset. This means that the command can be used periodically to produce statistical information covering specific intervals.

The information is displayed under the following headings:

SAWBUF
The number of session awareness buffers processed.

SAWENT
The number of session awareness entries that were processed from the buffers received from VTAM.

TRCBUF
The number of session trace buffers processed.

TRCENT
The number of trace PIU entries were processed from the buffers received from VTAM.

LOGGED
The number of session incidences which were logged to the NTS database.

READS
The number of records read from the NTS database during log processing.

ADDS
The number of records added to the NTS database during log processing.

UPDS
The number of records updated on the NTS database during log processing.

DELS

The number of records deleted from the NTS database during log processing.

SHOW NTSUSER

Function: Provides a summary of the current users of the NTS Review facilities, including NTS Database connections.

```
SHOW        NTSUSER
(0)
```

Use: The SHOW command is an MS command, but the NTSUSER operand is recognized by NTS as a request for current NTS usage information. A separate entry appears on each line of the display and represents the identity and processing location of each current user of NTS.

Notes: The SHOW NTSUSER command produces a heading line followed by a line for each currently active user of NTS. This line contains information relating a users current view of the internal NTS environment, whether the user is currently connected to the NTS Database and the name and identifier of the currently executing NTS NCL procedure.

Note

If a user is currently positioned at the NTS : Primary Menu panel, or positioned anywhere within NTS option 7 - Route Test Menu or NTS option C - Control Functions, no entry will appear for the user in the output of the SHOW NTSUSER command.

The information is displayed under the following headings:

USERID
The user ID string of this current user of NTS, entered to gain access to the MS region.

NTSVIEW
The view of the data within the NTS environment invoked by this current user.

D/B
(YES or NO) Whether or not this user is currently connected to the NTS database.

NCLBASE
The base NCL procedure for this instance of NTS Review Facilities. Multiple instances of NTS Review Facilities can be executed concurrently via the use of the function skipping keys. See the *Unicenter NetMaster Network Management for SNA Implementation and Administration Guide* for a detailed description.

NCLPROC
The name of the NTS NCL procedure currently being executed by this user.

NCLID
The NCLID of the current invocation of the procedure specified in NCLPROC.

SHOW OCS

Function: Displays information about OCS users.

```
SHOW        OCS
(0)
```

Use: The SHOW OCS command can be used to display information about the current OCS operators. The information displayed includes the user ID, terminal, user's name and location and the operational status of their OCS window.

Notes: The user's name that is displayed is extracted from the user ID definition as defined by the installation.

The operational status of the OCS window can be:

ACTIVE
: Means the window is active and can receive messages.

CLOSED
: Means the window is active, but is closed (another operational window occupies the entire screen dimensions) and any messages sent will be queued until the window is again opened. If the OCS user is currently executing in fullscreen mode a status of CLOSED will be displayed.

UNAVAIL
: Means the window is not available due to the terminal being in an in operational state. For example, it can be powered off, or the SYSREQ or TEST key pressed. Messages cannot be sent to the terminal.

It is useful for analysis of the current NRD queue by an NCL procedure and if it becomes necessary to delete NRDs through the PURGE NRD command.

SHOW PANELS

Function: Displays panel queue information.

```
SHOW        PANELS
(0)
```

Use: This command can be used to display the panel activity for the system.

The MS region maintains an *active panel queue* which holds in storage copies of panels currently in use by the system.

This means that if a request for a panel can be satisfied from a copy in the active panel queue then the usual I/O to read it from the panels dataset is avoided.

SHOW PANELS provides the following information.

PANEL
The name of a panel located in the active panel queue.

STATUS
Possible values are LOADED, indicating that the panel is actually resident in the active panel queue, or LOADING, indicating that a load is in progress for the particular panel.

HITS
The process of locating a requested panel in the active panel queue (thus avoiding I/O) is termed a *hit*. The number of hits for each panel found is displayed here.

PATH
The name of the library concatenation (path) from which the panel was loaded. Different users of the system can use different paths.

LIBRARY
The name of the library from which the panel was loaded. A single library can be used in multiple paths. Only one copy of a particular panel is kept in storage, even if it appears multiple times in the SHOW PANELS list for different paths.

AGEUSE
This is a number which reflects how often a panel has been *hit* (see above) in relation to the other panels in the active panel queue. It is used to ensure that the panels most likely to be used are retained in the queue.

Following these line displays comes an end message which also contains system values and totals which are useful in determining the value an installation should set for the depth of the active panel queue (SYSPARMS MAXPANEL).

ACTIVE
The number of panels currently in the active panel queue.

MAXPANEL

The value set by a SYSPARMS MAXPANEL command. If MAXPANEL is greater than the ACTIVE count, either not enough panel requests have been processed by the MS region since start-up to require any panels be purged from the active panel queue, or MAXPANEL exceeds all panel requests (that is, the depth of the active panel queue is greater than the number of unique panels requested).

REQUESTS

The total number of requests for all panels processed by the system.

HITS

The total number of times a requested panel was found on the active panel queue.

DROPS

The total number of times panels were dropped from the active panel queue because the number of panels set by SYSPARMS MAXPANEL was exceeded.

Notes:

If a panel, currently resident in the active panel queue, is updated and saved the entry in the active panel queue is invalidated and a subsequent request for the panel will result in it being fetched from the dataset.

This invalidation also occurs when library or path definitions are altered, or the library files are opened or closed using the UDBCTL command.

The SHOW PANELS command is most useful for monitoring panel activity to optimize the depth of the active panel queue (set using the SYSPARMS MAXPANEL command).

See Also:

The SYSPARMS MAXPANEL operand in the *Management Services Administrator Guide*.

SHOW PARM

Function: Displays start-up parameters.

```
SHOW        PARM
(0)
```

Use: The SHOW PARM command can be used to display the parameter data specified when the system was initialized. The PARM is specified by SYSPROGS and is described in the *Management Services Administrator Guide*.

Notes: Parameter data is specified on the JCL EXEC statement (OS/VS), on the ATTACH command in the GCS start-up procedure (VM/SP) or in the SYSIPT stream (VSE/SP).

SHOW PATH

Function: Displays panel library concatenation PATHs which have been defined to the system.

```
SHOW    PATH      [ =name ]
(0)
```

Use: Panel library PATHs are defined using the LIBPATH DEFINE command. The paths so defined can be replaced using the LIBPATH REPLACE command or deleted using the LIBPATH DELETE command. The path definitions can be displayed using the SHOW PATH command.

The information displayed by SHOW PATH is as follows:

PATH/LIB
The name of the path or library being displayed. The SHOW PATH command response is made up of a line to represent the path followed by a series of lines representing each library in the path. The library names are indented to allow you to distinguish between path names and library names.

EDIT
Indicates which libraries on the path can be edited:

ALL
The path definition allows all libraries in the path to be edited

NONE
The path definition does not allow any libraries in the path to be edited

LIST
The path definition allows selected libraries in the path to be edited

YES
The library named on this line can be edited

NO-LIB
The library definition does not allow this library to be edited

NO-PATH
The PATH definition does not allow this library to be edited

LIBDEF
Indicates if the library has been defined by the LIBRARY DEFINE command

DESCRIPTION
Is the description assigned to the path by the LIBPATH DEFINE command, or to the library by the LIBRARY DEFINE command.

SHOW PATH

Operands:

PATH

Displays current path definitions.

=name

The display is restricted by path name. If this value ends in an asterisk, then all path definitions whose name starts with this prefix are listed.

Examples:

```
SHOW PATH            -* List all path definitions
SHOW PATH=PANELS     -* List only the path named PANELS
SHOW PATH=P*         -* List all paths whose name starts with P
```

See Also: The LIBPATH DEFINE, LIBPATH REPLACE, LIBPATH DELETE, and SHOW LIB command descriptions.

SHOW PAUSE

Function: Displays the current pause status.

```
SHOW PAUSE      [ =ENV | =REGION | ALL | =name | ID=nclid ]
                [ PROC=procname ]
                [ PRTY=priority ]
                [ PG=perf-gp ]
(0/2)
```

Use: When executing NCL procedures, it is possible for execution to be suspended by the use of &PAUSE statements embedded in the procedure. The SHOW PAUSE command displays which currently active procedures are in suspend state.

If SHOW PAUSE is entered without any other operands, then the current pause status is displayed.

Operands:

=ENV
When this operand is specified or allowed to default, NCL running in the current and dependent environments is scanned.

=REGION
Scans all NCL procedures running for the current user, including both logical windows when split screen mode is in use.

=ALL
Scans all NCL in all regions, including NCL running in EASINET regions. This operand requires a command authority level of 2.

=*name* (user ID or terminal name)
Scans all NCL for the specified user or terminal. This operand requires a command authority level of 2 if the User ID or terminal name is not the issuing user or terminal.

ID=*nclid*
Displays a particular NCL process, and any dependent NCL processes, identified by the supplied ID. If the supplied ID is not associated with any NCL process running in the issuing region a command authority level of 2 is required. This operand is mutually exclusive with those above.

One or more of the following optional operands can be used to restrict the display to particular NCL processes that satisfy all of the specified criteria:

PROC=*procname*
Selects NCL processes that are running the specified procedure. PROC selects on the current procedure and on the base procedure.

PRTY=*priority*
Selects NCL procedures that are running at the specified priority (0 to 3).

SHOW PAUSE

PG=*perf-gp*
Selects NCL procedures that are running in the specified performance group (1 to 4).

Examples:

```
SHOW PAUSE
SHOW PAUSE=ALL PROC=$PAUSPROC
SHOW PAUSE=USER6
```

Notes:

Processing of a procedure can be terminated using the END and FLUSH commands.

If an OCS operator exits while an NCL procedure running in the current window is in a paused state, the procedure will be flushed.

When a procedure is paused due to the processing of an &PAUSE statement, an appropriate mode indicator (W or P) will remain displayed to the left of the command line until the pause state is ended.

See Also:

The GO, FLUSH, and END command descriptions.

SHOW PPIUSERS

Function: Displays statistical information of Program-to-Program Interface (PPI) users.

```
SHOW PPIUSERS
(0)
```

Use: The SHOW PPIUSERS command displays a list of PPI users, and various statistics about those users.

Notes: Only the system connected to the SOLVE Subsystem Interface (SSI) that is providing PPI services can process this command.

SHOW PPOSTAT

Function: Displays PPO statistical information.

```
SHOW   PPOSTAT
(0)
```

Use: Displays PPO message counts.

Notes: The resulting messages show message counts delivered across the PPO and SPO interface, the number delivered to PPOPROC, and the number deleted or delivered locally and to remote systems.

This provides an indication of how well the DEFMSG filters are reducing traffic to PPOPROC.

See Also: The DEFMSG command description.

SHOW REPLY

Function: Displays VTAM messages that require or required a reply.

```
SHOW        { REPLY | REPLIES }
(0)
```

Use: The system can receive VTAM messages that require a reply from the operator, using the REPLY command. The system keeps a time-stamped list of the last four such messages and will display them if this command is issued. These messages have an identification number associated with them that is required in the REPLY command used to respond to the message.

This reply identification is in the format *lnn*, where *l* is either the letter P or S and *nn* is a two digit number in the range 00 to 99. When replying to a message the exact format of the reply identification must be used.

Notes: The messages displayed by this command could have had replies entered for them. The outstanding replies will be NRD messages. If profiled to receive these messages, the operator can determine which replies, if any, have not been responded to, by using the NRDRET command.

See Also: The REPLY command description.

SHOW SCNT

Function: Displays the current session count.

```
SHOW    SCNT
(0)
```

Use: This command returns the total session count, which includes EASINET terminals plus users who are currently logged on to the MS region.

See Also: The SHOW SESS and SHOW USERS command descriptions.

SHOW SERVER

Function: Displays the status of registered server processes.

```
SHOW SERVER      [ =servername* | nameattr=servername* ]
                 [ CLASS=class ]
                 [ SCOPE={ ANY | CONTEXT | REGION | USER | SYSTEM } ]
(0)
```

Use: An NCL process can be registered as a *server process*. Registration involves association of the server with an object class. NCL server procedures have a class of ZZZPROCESS and a naming attribute of ZZZSERVER. Servers created during the processing of object services requests will have the class and name ID of the object identified in the request service data unit.

Registered *servernames* can be displayed using the SHOW SERVER command.

Operands: ***=servername* | nameattr=servername****

Specifies a full or partial name to restrict the display. Server names depend on the object class for which they are defined. NCL procedure defined servers are in the ZZZSERVER object class.

Specification of SERVER=*servername** is equivalent to SERVER ZZZSERVER=*servername**.

The *nameattr* specification identifies the naming attribute for the server. When specified, selection will be restricted to object servers for the naming attribute with values matching the generic *servername**.

CLASS=*class*

Specifies the class name for object servers to be displayed.

SCOPE={ ANY | CONTEXT | REGION | USER | SYSTEM }

The SCOPE operand is optional and can be used to specify the scope of the registered *servername(s)* to be displayed.

If SCOPE=ANY is specified or defaulted, all registered *servernames* that are not restricted by the SERVER= operand, are displayed. If CONTEXT, REGION, USER, or SYSTEM is specified, only *servernames* registered within that scope are displayed.

CONTEXT indicates that *servername* is to be unique within the current NCL context.

REGION indicates that *servername* is to be unique within the current region, as defined by a particular connection to Management Services.

USER indicates that *servername* is to be unique across all regions associated with the particular user ID.

SYSTEM indicates that *servername* is to be unique within this system.

SHOW SERVER

Examples:

```
SHOW SERVER
SHOW SERVER SCOPE=REGION
SHOW SERVER=MY*
SHOW SERVER CLASS=ZOSSEQ
SHOW SERVER ZOSSEQID=* SCOPE=CONTEXT
```

See Also: The START, FSPROC, FLUSH, END, and GO command descriptions.

SHOW SESS

Function: Displays the terminals that are in session.

```
SHOW        SESS [ =pattern ]
            [ AM=(am1,am2,...) ]
            [ EASINET={ NO | YES } ]
(0)
```

Use: SHOW SESS can be used to display the terminals that are currently in session with MS domains, including those that are under the control of the EASINET facility.

Issuing SHOW SESS with no operands displays only the LU names of those terminals currently in session. This is the same display as that produced by the SHOW TERM command.

Issuing SHOW SESS with any operand displays additional information for each LU name displayed.

Operands:

SESS [=*pattern*]

Displays terminals currently in session.

Omitting =*pattern* means that *all* terminals currently in session are displayed.

Specifying a *pattern* (for example, NM*) allows you to restrict the list to LU names that match the entered pattern.

AM=(*am1,am2,...*)

Restricts the list to sessions using specific access methods; for example, AM=(VTAM,TCPIP).

Valid access methods are:

- VTAM
- TCPIP
- XNF
- EPS
- SYSTEM

The SYSTEM access method represents the internal MS sessions (background regions).

EASINET={ NO | YES }

Allows you to omit (NO) or include (YES) all EASINET session terminals.

SHOW SESS

Notes: Users signed on to MS domains can be displayed by using the SHOW USERS command.

See Also: The SHOW TERM and SHOW USERS command description.

SHOW SKEEP

Function: Displays the NTS session keep counts for historical sessions.

```
SHOW SKEEP      { PRINAME=priname | SECNAME=secname }
                [ PRINET=prinet ]
                [ REFNET=refnet ]
                [ SECNET=secnet ]
(0)
```

Use: Provides a display of the session keep counts for historical sessions stored on the NTS database.

Operands:

SKEEP

Provides a display of session keep counts. Further operands can be specified to extract various other information as follows.

PRINAME=*priname*

Provides the name of the primary session partner for which sessions are to be selected. This name be a generic name specified with a trailing asterisk (for example, APPL*). If omitted then all sessions with the secondary session partner will be selected.

SECNAME=*secname*

Provides the name of the secondary session partner for which sessions are to be selected. This name be a generic name specified with a trailing asterisk (for example, LUXX*). If omitted then all sessions with the primary session partner will be selected.

Note

Either or both PRINAME and SECNAME must be coded.

PRINET=*prinet*

Provides the name of the primary session partner's network for a cross-network session. If omitted, the network in which NTS is operating is assumed.

REFNET=*refnet*

Provides the name of the reference network for a session incidence. In an NTS Single Image environment where session awareness data is being received from NTS systems in other networks, it is possible that multiple session incidence records will be logged. Each represents the session from the point of view of a different network. If omitted, the network in which NTS is operating will be assumed.

SECNET=*secnet*

Provides the name of the secondary session partner's network for a cross network session. If omitted the network in which NTS is operating is assumed.

SHOW SKEEP

Examples:

```
SHOW SKEEP PRINAME=TSO*
SHOW SKEEP SECNAME=LU123 PRINAME=ISM PRINET=NETA
```

Notes: On command completion the number of sessions selected for processing is advised.

See Also: NTSDBMOD command description.

SHOW SNAMS

Function: Displays SNAMS registered applications, focal points or entry points known to the local system.

```
SHOW SNAMS      APPL [ NAME=applicationName ] |
                EP   [ CATEGORY=msCategory ] |
                FP   [ CATEGORY=msCategory ]
(0)
```

Use: **SHOW SNAMS APPL**

Displays a list of SNA Management Services applications registered with the local MDS Router. The information shown for each listed application is as follows:

NAME
This is the registered application name, known only to the local system.

ID
This is the routing name recognized by the SNAMS. If it is an SNA-architected application, it is displayed as a hexadecimal quoted string. Otherwise it is displayed as EBCDIC text.

NCLID
This is the NCL ID of the process which registered the application.

FP
Indicates whether or not this application has registered as a focal point for an MS category. Values are Y for yes or N for no. The registered categories are available from the detailed display for this application.

FPN
Indicates whether or not this application has registered for focal point notification for an MS category. Values are Y for yes or N for no. The registered categories are available from the detailed display for this application.

Operands: **NAME=*applicationName***
This is a request to display information on a specific MS application which is registered as *applicationName*. The details displayed include focal point and focal point notification MS categories registered by the specified application.

Examples:

```
SHOW SNAMS APPL
SHOW SNAMS APPL NAME=USERAPPL
```

SHOW SNAMS

Use: **SHOW SNAMS FP**

Displays a list of APPN focal points known to the local node (that is, the local node assumes the role of an entry point). The information shown for each listed MS category is as follows:

CATEGORY
Focal points are referenced by their corresponding MS category code. If it is an SNA-architected application, it is displayed as a hexadecimal quoted string. Otherwise it is displayed as EBCDIC text.

DESCRIPTION
This is the descriptive name of the MS category. The MS region provides predefined descriptive names for the following SNA-architected MS categories:

X'23F0F1F5'	COMMON_OPS
X'23F0F1F7'	OPERATIONS_MGMT
X'23F0F3F1'	PROBLEM_MGMT

NODE
Contains the network ID and NAU name of the active focal point for this category. If the focal point is currently active, it is displayed in the format NETID.NAUNAME. If it is inactive or pending active, this value is not displayed.

APPL
This is the application routing name of the active focal point. If it is an SNA-architected application, it is displayed as a hexadecimal quoted string. Otherwise it is displayed as EBCDIC text.

STATUS
Indicates the current status of the focal point. Possible values are:

INACTIVE	Focal point is inactive and undefined
PENDING	Primary/Backup focal point is pending active
ACT-PRI	Primary focal point is active
ACT-BKUP	Backup focal point is active
PRIMARY	Primary focal point is inactive
BACKUP	Backup focal point is inactive
LOCAL	The focal point application is registered on the local node.

Operands: **CATEGORY=*msCategory***
Optionally restricts the display to those focal points with the generic category *msCategory* provided.

Examples:

```
SHOW SNAMS FP
SHOW SNAMS FP CATEGORY=USERCAT
SHOW SNAMS FP CATEGORY='23F0F3F1'X
```

Use: **SHOW SNAMS EP**

Displays a list of APPN entry points in the sphere of control of the local node (that is, the local node assumes the role of a focal point). The information shown for each listed entry point category is as follows:

CATEGORY
Focal points are referenced by their corresponding MS category code. If an SNA-architected application, it is displayed as a hexadecimal quoted string. Otherwise it is displayed as EBCDIC text.

DESCRIPTION
This is the descriptive name of the MS category. The MS region provides predefined descriptive names for the following SNA-architected MS categories:

X'23F0F1F5'	COMMON_OPS
X'23F0F1F7'	OPERATIONS_MGMT
X'23F0F3F1'	PROBLEM_MGMT

NODE
This comprises the network ID and NAU name of the entry point displayed in the format NETID.NAUNAME.

STATUS
Indicates the status of the focal point with respect to the entry point. Possible values are as follows:

PENDING	Primary/Backup focal point is pending active
ACT-PRI	Primary focal point is active
ACT-BKUP	Backup focal point is active

Operands: **CATEGORY=*msCategory***
Optionally restricts the display to those focal points with the generic category *msCategory* provided.

Examples:

```
SHOW SNAMS EP
SHOW SNAMS EP CATEGORY=USERCAT
SHOW SNAMS EP CATEGORY='23F0F3F1'X
```

SHOW SOCKETS

Function: Displays information about the use of TCP/IP services.

```
SHOW  SOCKETS
```

Use: This command is used to display information about the use of TCP/IP sockets by this region. It displays information about all open sockets that are used for communication with the TCP/IP subsystem.

Examples: `SHOW SOCKETS`

SHOW SSIEPS

Function: Displays SSI EPS information.

```
SHOW      SSIEPS
        [ DETAIL ]
        [ LINKS ]
        [ TOPOLOGY ]
        [ NOTIFY ]
        [ STATS ]
(0)
```

Use: The SHOW SSIEPS command displays information and statistics about the SOLVE Subsystem Interface (SSI) Endpoint Services (EPS) environment.

Operands:

SSIEPS
Mandatory—indicates that this is a SHOW SSIEPS command. If no other operands are provided, then a summary list of endpoints known to the SOLVE SSI region is displayed.

DETAIL
Causes the display to be augmented with additional information about each endpoint.

LINKS
Provides a list of all direct links to other connected endpoints (other SOLVE SSI regions or other connected MS regions) to be displayed. Some access methods (for example, XCF) provide additional information.

TOPOLOGY
Adds a complete topology display to the output, showing all links between endpoints.

NOTIFY
Augments the topology display with a list of the notifier endpoints for each link.

STATS
Provides additional statistical information.

Examples: `SHOW SSIEPS`

Notes: Only an MS region connected to the SOLVE SSI can process this command.

SHOW SSISTATS

Function: Displays SSI statistics.

```
SHOW  SSISTATS
(0)
```

Use: This command displays a list of user environments signed on the SOLVE SSI.

Notes: Only the system connected to the SOLVE SSI can process this command.

SHOW SSITERMS

Function: Displays a list of terminals attached to the SOLVE SSI.

```
SHOW  SSITERMS
(0)
```

Use: The SHOW SSITERMS command is used to display information about terminals which are connected to the SOLVE SSI. The display shows information such as the channel address of the terminal, its symbolic name, the DD name assigned during dynamic allocation of the device, and if the terminal is currently connected to an MS region.

Operands: **SSITERMS**
Displays a list of terminals attached to the SOLVE SSI.

Examples: `SHOW SSITERMS`

See Also: The ATTACH and DETACH commands.

SHOW SSIUSERS

Function: Displays SSI user details.

```
SHOW   SSIUSERS
(0)
```

Use: The SHOW SSIUSERS command displays a list of user environments signed on to the SOLVE SSI.

SHOW STRACE

Function: Displays NTS session trace activity.

```
SHOW STRACE      [ =ALL ]
(0)
```

Use: This command displays information regarding both the global and specific trace options.

Operands: **=ALL**

Provides a display of the current session trace options in effect by NTS. If STRACE=ALL is coded, all session traces started or stopped by NTS are displayed, including trace activity which has been exclusively started by NTS for accounting only. If =ALL is omitted, then only traces started by operator command, and hence accruing trace data for the sessions concerned, are displayed.

Examples:

```
SHOW STRACE
SHOW STRACE=ALL
```

Notes: A heading message followed by one or more detail messages are issued in response to this command. The first detail message provides the status of the global trace option for each resource type that can be globally traced (namely LUs, PUs and CDRMs). Subsequent messages provide the status of any specific trace requests which are currently active.

The status of NTS tracing can also be examined from option C - Control Functions on the NTS : Primary Menu.

See Also: The STRACE command description.

SHOW SUBSYS

Function: Displays user subsystem status.

```
SHOW SUBSYS      [ =* | =name | =prefix* ]
(0)
```

Use: Displays information about defined user subsystems. By default all defined subsystems are displayed. A specific subsystem, or all starting with the same prefix can be displayed.

The output from SHOW SUBSYS includes:

- The name of the subsystem.
- The name of the associated program.
- The status (ACTIVE or STOPPED). If stopped the reason that it was stopped is also shown (such as not started, stopped by *command*, ABEND and so on).
- The number of requests queued from NCL at this instant. This can be used to determine if the subsystem is hung.
- Whether the subsystem can be FORCEd (YES or NO). That is, whether the subsystem is currently executing user code.
- The PARMLIST and AUTOCALL options.

Operands: If no operand is provided, all defined subsystems are displayed.

=*
: Displays all defined subsystems.

=name
: Displays information about a specific subsystem.

=prefix*
: Displays information about all defined subsystems with names starting with *prefix*.

Examples:

```
SHOW SUBSYS
SHOW SUBSYS=USER*
```

See Also: The SUBSYS command description.
The &CALL verb description in the *Network Control Language Reference*.

SHOW SYSCONS

Function: Displays a list of currently signed on operating system consoles and their user IDs.

```
SHOW    SYSCONS
(0)
```

Use: Used to display a list of signed on system consoles.

The columns displayed are as follows:

CON-NAME
The console logical or terminal name. Equivalent to the LU name (shown as this in SHOW USERS display).

KEY-UID
The user ID used as the logical key when incoming commands are processed. Used to detect when an operating system console has a new RACF signon.

ACT-UID
The actual user ID under which the console has been signed on to the MS region.

CON-ID
The console ID. Either a 2-digit decimal number (00 to 99) or 8 hexadecimal digits (being the extended 4-byte console ID).

UID-1,UID-2,UID-3
The user IDs used during signon processing to establish the actual signed on user ID.

Notes: The output of this command can aid in problem resolution with console signon names. In a non-MVS or similar environment (VM/GCS or VSE) there will only ever be one entry in the displayed list. In an environment that does support multiple consoles, but which does not support named consoles (that is, not MVS/ESA 4.1 or later), the console names will always be derived from the console IDs. Similarly, the user IDs will be derived from the console number or the basic console user ID.

In MVS/ESA 4.1 or later, the actual console names and signed on user IDs can be used, depending on the SYSPARMS settings.

See Also: The SHOW CONSOLES and SHOW USERS command descriptions.

SHOW SYSPARMS

Function: Displays the current SYSPARMS settings.

SHOW SYSPARMS [=*prefix*] (0)

Use: Lists the status of SYSPARMS parameters current at the time of the display.

Operands: ***=prefix***

Specifies that the display is to list only the SYSPARMS operands that start with the specified prefix.

Examples:

```
SHOW SYSPARMS
SHOW SYSPARMS=NCL
```

Notes: The SYSPARMS command is described in the *Management Services Administrator Guide*.

There is a fullscreen presentation of all SYSPARMS settings (on the Display/Update SYSPARMS Operands panel) within the System Support Services function. The menu options to access this panel depend on whether your region uses Automation Services:

- Use =MS.S.T.S without Automation Services
- Use =A.SS.M.S.T.S with Automation Services

SHOW TCPIP

Function: Displays information about the use of TCP/IP services.

SHOW TCPIP [USERS | STATUS]

Use: This command is used to display information about the use of TCP/IP services by this region.

Operands: **USERS | STATUS | AM**

Instructs the system what information to display:

USERS

Instructs the system to display information about the users of TCP/IP services.

STATUS

Instructs the system to display information about the TCP/IP services status.

AM

Instructs the system to display a list of active TCPIP access method sessions.

Examples:

```
SHOW TCPIP USERS
SH TCPIP
```

SHOW TERM

Function: Displays the terminals that are in session.

```
SHOW TERM
(0)
```

Use: SHOW TERM can be used to display the terminals that are currently in session with MS domains.

This command displays all terminals in session, including those that are under control of the EASINET facility.

Notes: Users that are signed on to the MS region can be displayed by using the SHOW USERS command.

See Also: The SHOW SESS and SHOW USERS command descriptions.

SHOW TIMER

Function: Displays current timer initiated commands.

```
SHOW TIMER      [ =OWN | =ALL ]
                [ TID=timerid ]
(0)
```

Use: The SHOW TIMER command can be used to display any pending timer initiated commands that are waiting to be executed. Timer initiated commands are created by either the AT command or the EVERY command. They provide a means of executing commands at a predetermined time or at a given frequency.

The SHOW TIMER command, by default, displays only those timer commands that are scheduled for execution under the control of the requesting user ID. Optionally, you can specify the ALL operand to display all outstanding timer commands.

Operands:

=OWN
Displays requesting user ID's outstanding timer commands.

=ALL
Displays all outstanding timer commands.

TID
Specifies an identifier for the timer. This further refines ALL or OWN by matching on the user-assigned timer ID. This operand can be specified as an absolute value (for example, TID=CHECKFILES) or a generic match (for example, TID=CHECK*).

Examples:

```
SHOW TIMER
SHOW TIMER=ALL
```

See Also: The AT, EVERY, and PURGE command descriptions.

SHOW TIMEZONE

Function: Displays defined time zones and the system time zone offset.

```
SHOW   TIMEZONE
(0)
```

Use: The SHOW TIMEZONE command displays a list of the current time zone names, their Greenwich Mean Time offset, and description. The system time zone offset is also displayed.

Notes: The SHOW TIMEZONE display has this format:

```
N17K01 SYSTEM TIMEZONE OFFSET IS +1000
N17K02 TZ-NAME  OFFSET   DESCRIPTION
N17K03 AUSESTDT +1000    Australian Eastern Std Time
N11907 *END*
```

See Also: The TIMEZONE command description.

SHOW TNCTL

Function: Displays a list of all defined TNCTL (Telnet) ports and their option settings. The ports are listed in ascending order, with the SHARED port first, if it is defined.

SHOW TNCTL

Examples:

```
SHOW TNCTL
SH TNCTL
```

SHOW TSO

Function: Displays TSO users.

SHOW (0)	{ TSO | TSOS | TSO=*asid* | TSO=*search-arg* }

Use: This command is supported under z/OS and OS/390. It allows the displaying of pertinent information about TSO users. This information is extracted from the operating system. Information displayed includes the procedure which the user used when logging onto TSO, the address space identifier (ASID), the terminal which they are using, and the TSO appl ID.

The current TSO MAXUSERS setting and the current number of users is also displayed.

Operands:

TSO
Displays the current TSO users.

TSOS
Displays current TSO users including *startings*.

TSO=*asid*
Displays only the selected operating system address space ID, if currently in use for a TSO session.

TSO=*search-arg*
Specifies a partial or full argument that will be used as a generic operator on data contained in the User ID, Nodename, Applid and Procname fields. All entries meeting this selection criteria will be displayed. The *search-arg* operand need only specify leading characters.

Examples:

```
SHOW TSO
SHOW TSO=TERM01
SHOW TSO=AB001
SHOW TSO=006F
SHOW TSO=#LPROC1
```

Notes: The system must be APF authorized for this command to be executed.

See Also: The SHOW TSS command description.

SHOW TSS

Function: Displays TSS users.

SHOW (0)	{ TSS | TSSS | TSS=*asid* | TSS=*search-arg* }

Use: This command is only supported under MSP and VOS3. It allows the displaying of pertinent information about TSS users. This information is extracted from the operating system. Information displayed includes the procedure which the user used when logging onto TSS, the address space identifier (ASID), the terminal which they are using, and the TSS APPLID.

The current TSS MAXUSERS setting and the current number of users is also displayed.

Operands:

TSS
Displays the current TSS users.

TSSS
Displays current TSS users including *startings*.

TSS=*asid*
Displays only the selected operating system address space ID, if currently in use for a TSS session.

TSS=*search-arg*
Specifies a partial or full argument that will be used as a generic operator on data contained in the User ID, Nodename, Applid and Procname fields. All entries meeting this selection criteria will be displayed. The *search-arg* operand need only specify leading characters.

Examples:

```
SHOW TSS
SHOW TSS=TERM01
SHOW TSS=AB001
SHOW TSS=006F
SHOW TSS=#LPROC1
```

Notes: The system must be APF authorized for this command to be executed.

See Also: The SHOW TSO command description.

SHOW UDB

Function: Displays VSAM dataset information.

```
SHOW UDB      [ =ddname ]
(0)
```

Use: This command can be used to determine the status of both Management Services' own VSAM datasets and User DataBases (UDBs). Management Services VSAM services are driven by a component called VFS (Virtual File Services).

The MS region uses a number of VSAM system datasets to control its operation.

The information returned by this display is as follows:

DDNAME
The JCL DD or DLBL name associated with this file.

FILEID
For UDBs this is the logical file ID that was assigned with the UDBCTL command. This is the file ID that must be used when referring to this file with an NCL &FILEID statement. For UDBs a single dash (-) in this field indicates that no file ID is currently assigned. In this case the STUS field will be set to STOP.

TYPE
The type of VSAM dataset.
(BASE = *base cluster*, AIX = *alternate index*, PATH = *path*)

A/U
The number of active users that currently have a logical connection to the file through VFS. This might include NCL procedures such as LOGPROC. It will not be possible to logically stop access to a dataset (using the UDBCTL command) unless this number is 0.

SM
The maximum number of strings used by VSAM to service this file. This can be used to assist in tuning the dataset.

DATA
The percentage utilization of the DATA component (See notes below).

EXT
The number of extents used by the DATA component. (See notes below).

INDX
The percentage utilization of the INDEX component. (See notes below).

EXT
The number of extents used by the INDEX component. (See notes below).

RKP
The relative key position defined for the dataset.

KYL
The length of the key defined for the dataset.

UKY
For AIX specifies if the dataset requires unique keys.

STUS
The current dataset status.

SYS
This is a system dataset used by Management Services.

AVAL
A UDB which is available for use. This dataset can be accessed by referencing it with an NCL &FILEID statement specifying the file ID in the FILEID field.

STOP
A UDB which is logically stopped because no logical file ID has been assigned using the UDBCTL command or the dataset has been stopped using the UDBCTL command. A status of stopped does NOT imply that the dataset is closed.

CLSD
A UDB which is physically closed. This can occur either as a result of an error during open processing (in which case the ACB open code will also be displayed) or because the UDBCTL command has been used to close the dataset, perhaps for off-line processing. It will not be possible to assign a file ID to this UDB until it has been successfully opened using the UDBCTL command.

ERR
Error occurred during open for the dataset.

KERR
Dataset not supported because key length greater than maximum supported.

UERR
Error detected in dataset definition.

OPTNS

One or more of the following codes, indicating options used when opening the dataset:

Code	Meaning
R	Opened for reuse
I	Opened for input only
S	Using Sequential Insert Strategy (SIS)
L	Using the Local Shared Resource (LSR) pool
D	Using deferred writes
F	File contains Fujitsu (that is, not IBM) DBCS control characters

Operands:

UDB

Displays all VSAM dataset information.

UDB=*ddname*

Displays the selected dataset only. If *ddname* ends with an asterisk (*), then it is treated as a generic value, and all datasets whose DD names match the generic value are displayed.

Examples:

```
SHOW UDB
SHOW UDB=IMNDB
SHOW UDB=IM*
```

Notes:

If an error occurred that stopped the MS region from opening a UDB, then the associated ACB open failure code is displayed in a message that replaces the information for that file. An attempt to reopen the UDB can be made using the UDBCTL command.

Other information is extracted from the VSAM ACB and hence reflects VSAM's view of the dataset. The current percentage utilization can be inaccurate. The available space figures maintained by VSAM only include those control areas that have never been used and hence are completely empty. It is possible for records to occupy a control area and then be deleted, thus leaving the control area empty. VSAM does not then include this amount in its available space figures. However, when the percentage is used in conjunction with the extents information, the user can determine if the dataset has approached its maximum possible allocation and hence should be enlarged.

VSAM can have a maximum of 123 extents if the available space on the volume and/or in the dataspace permits.

The DATA and INDX columns in the display are not supported in VSE and VM systems and will contain a dash (-) to indicate that no data is available.

See Also:

The UDBCTL, SHOW VSAM, and SHOW UDBUSER command descriptions.

SHOW UDBUSER

Function: Displays NCL UDB user information.

```
SHOW    UDBUSER
(0)
```

Use: This command can be used to determine the active NCL UDBs and displays the user IDs and procedures which are currently using the file. The information returned by this display is as follows:

DDNAME
The JCL DDNAME (OS/VS) or DLBL name (VM/SP and VSE/SP) associated with this file.

FILEID
The logical file ID assigned to the UDB using the UDBCTL command. This is the name used by the NCL procedure to identify the file it is using.

USERID
Identifies a user or system task which is currently processing the UDB.

NCL-PROC
Identifies the NCL procedure which issued the &FILEID statement for the UDB on behalf of the user ID shown.

NCLID
Identifies the NCL ID of the NCL process under which the NCL procedure is executing.

LUNAME
Identifies the LU name at which the NCL procedure is executing.

One of the following captions can also be displayed:

(STRSH)
The user is currently experiencing a VSAM string shortage condition, and is suspended until a string becomes available.

(BFRSH)
The user is currently experiencing a VSAM buffer shortage condition, and is suspended until buffers become available.

Notes: File processing is active if the base procedure, or a nested procedure, has issued an &FILEID for the UDB, but either the base procedure has not ended, or an &FILEFREE has not been issued against this UDB.

See Also: The UDBCTL, SHOW UDB, SHOW VSAM, and SHOW USERS command descriptions.

SHOW USERACCT

Function: Displays the status of user NCL CPU time accounting.

```
SHOW  USERACCT
(0)
```

Use: This command displays the status of user NCL CPU time accounting. User NCL CPU time accounting is controlled by the USERACCT command. The SHOW USERACCT command is used to discover if user NCL CPU time accounting is active, and if so when it was started, how often reports are generated, and the SMF record ID of user NCL CPU time reports.

Operands: **USERACCT**
Requests the display of the current status of user NCL CPU time accounting.

Examples: `SHOW USERACCT`

See Also: The *Recording CPU Usage* chapter in the *Management Services Administrator Guide*.

SHOW USERS

Function: Displays the current signed-on users.

```
SHOW USERS      [ =name | =prefix ]
                [ SOURCE=type ]
                [ IPDETAILS= YES | NO ]
(0)
```

Use: The SHOW USERS command can be used to display the users that are signed on to the system. The display shows the node names of the terminals at which local users are logged on as well as connections to and from remote MS regions.

If SHOW USERS is entered without any other operands, then a list of all signed-on users is displayed.

Operands:

=name
Displays the nominated user.

=prefix
A prefix can be entered to provide a generic display of all user IDs that start with the nominated prefix. The prefix is terminated with an asterisk (*).

SOURCE=*terminal* | *local-ip-address* | *remote-ip-address*
Displays a list of users from a particular origin. You can nominate a specific terminal name or an IP address, either local or remote. You can also use wildcard characters (* or ?) to display a list of users from matching terminals or IP addresses.

IPDETAILS=YES | NO
Specifies whether IP addresses (local and remote) are displayed in the output.

Examples:

```
SHOW USERS                      -* List all users
SHOW USERS=NMBMON               -* List user NMBMON only
SHOW USERS=NM*                  -* List all user IDs starting
                                   with NM
SHOW USERS IPDETAILS=YES        -* List all users and their IP
                                   addresses
SHOW USERS SOURCE=NMMAF032      -* List users attached to
                                   terminal NMMAF032
SHOW USERS SOURCE=123.0.11.22   -* List users with local or
                                   remote IP address 123.0.11.22
```

Notes: Terminals that are in session but are not currently in use by a signed-on user ID can be displayed by using the SHOW SESS command.

Your *own* user ID is displayed in high-intensity (if on a VDU type terminal).

If a user is logged on from a remote location, then the TERMINAL name is shown as *REMOTE* and the TYPE as INMC for INMC. This is followed by the word FROM and the ID of the remote MS region from which the user has logged on.

If a user has one or more connections to a remote MS region, then one line is displayed for each connection, following the standard information line for that user. In such cases, the terminal TYPE is shown as INMC followed by the word TO and the ID of the remote MS region.

If a terminal is powered off or otherwise unavailable (for example, in TEST mode or SYSREQ pressed), then the caption (U/A) is displayed.

If a user has locked their terminal by using the LOCK command, then the caption (LOCKED) is displayed.

A disconnect time is given for users with a disconnect environment. A numeric reconnection identifier is displayed in the TERMINAL field.

The display also shows internal system user IDs associated with the processing environments for system level procedures such as LOGPROC or PPOPROC.

See Also: The SHOW SESS command description.

SHOW VARTABLES

Function: Displays information about GLOBAL or REGION or AOM vartables.

```
SHOW  VARTABLES   [ ={ SYSTEM | AOM | nclid | REGION } ]
                  [ USER=userid | LU=luname ] ]
                  [ NAME=name ]
(0)
```

Use: The SHOW VARTABLES command displays a list of all currently allocated vartables, allocated by the NCL &VARTABLE ALLOC statement. By default, a list of the SYSTEM vartables is displayed.

Operands:

=SYSTEM
This optional operand indicates that a list of all SYSTEM vartables is to be produced. This is the default.

=AOM
This operand indicates that vartables with a scope of AOM are to be displayed.

=nclid
This optional operand indicates that a list of all PROCESS vartables for the nominated NCLID is to be produced. The NCL process must be executing in the issuer's region.

=REGION
This optional operand indicates that a list of all REGION vartables is to be produced. The list will indicate the LU name and user ID that each table is associated with. The USER= or LU= parameters can be used to restrict the display to a specific region.

USER=*userid*
For SHOW VARTABLES=REGION, restrict the display to the indicated *userid*.

LU=*luname*
For SHOW VARTABLES=REGION, restrict the display to the indicated terminal (LU).

NAME=*name*
Restrict the output to vartables commencing with the indicated name or prefix.

Examples:

```
SHOW VARTABLES
SHOW VARTABLES=REGION
```

See Also: The &VARTABLE verb description in the *Network Control Language Reference*.

SHOW VMOP

Function: Displays the AOM/VM sessions active to VMOPERATOR.

```
SHOW   VMOP
(0)
```

Use: This command can be used to display a list of the following active commands:

- SYSCMD DEST=VMOP
- SECUSER
- SYSLOG

Notes: Sessions for your own user ID will have your user ID displayed in high intensity. If the session is from the same window the SHOW VMOP command was issued from, the destination user ID will also be displayed in high intensity.

If a SECUSER session has been established as an exclusive session, the caption (EXCL) will be displayed after the destination userid.

SYSCMD DEST=VMOP sessions only remain active for the duration of the command.

SECUSER and SYSLOG sessions remain active until a command specifying OPT=REL is issued, or until AOM/VM or VMOPERATOR terminates.

Function: Displays extended VSAM dataset information.

SHOW (0)	{ VSAM | VSAMX } [=*ddname*]

Use: This command can be used to determine attributes of both Management Services' own VSAM datasets and User DataBases (UDBs).

Operands:

VSAM
Show attributes of all datasets.

VSAMX
For each VSAM dataset show an additional line of data and index counts (see notes below).

=ddname
Show attributes of the specific *ddname* entered. If *ddname* ends with an asterisk (*) then it is treated as a generic value, and all datasets whose DD name matches the generic value will be listed.

Examples:

```
SHOW VSAM
SHOW VSAM=IMNDB
SHOW VSAM=IM*
```

Notes: The information is returned by this display under the following headings:

```
N15101 DDNAME RECSZ D-CISZ I-CISZ CI-SP CA-SP D-BF I-BF STRSH BFRSH LSR CTL
```

which have the following meanings:

DDNAME
The JCL DD name (OS/VS) or DLBL name (VSE/SP) associated with this file.

RECSZ
The defined maximum record size for the dataset.

D-CISZ
The VSAM Control Interval size for the Data component.

I-CISZ
The VSAM Control Interval size for the Index component of a KSDS.

CI-SP
: The number of Control Interval splits that VSAM has performed for this dataset.

CA-SP
: The number of Control Area splits that VSAM has performed for this dataset.

D-BF
: The maximum number of data buffers allocated for processing a string of the VSAM dataset. This value represents a high-water mark and can be used in tuning the BUFND VSAM operand specified in the JCL or on the UBDCTL command.

Note

On a z/OS or OS/390 system the value shown might exceed the specified BUFND value, due to dynamic string expansion.

I-BF
: The maximum number of index buffers allocated for processing a string of the VSAM dataset. This value represents a high-water mark and can be used in tuning the BUFNI VSAM operand specified in the JCL or on the UBDCTL command.

Note

For z/OS or OS/390 the value shown might exceed the specified BUFNI value, due to dynamic string expansion.

STRSH
: The number of times a string shortage condition caused a temporary delay in processing a file. Management Services writes a monitor message the first time this occurs only.

BFRSH
: The number of times a buffer shortage condition caused a temporary delay in processing a file. This should only occur for datasets running with LSR=YES (see below).

LSR
: YES or NO indicates whether or not this VSAM dataset is using the Local Shared Resource pool.

CTL
: DSN or DD indicates whether or not VSAM is sharing control block structures at a Dataset name level (DSN) or at a DD name level (DD). This applies to z/OS and OS/390 only. When processing with Alternate Indices or Paths sharing should be at the DSN level.

If there are any datasets shown with LSR=YES further messages provide LSR pool information as follows.

BUFFSZ
The buffer size for this LSR pool.

BUFFND
The number of times requested data was found in a buffer in storage, thus avoiding I/O.

BUFRDS
The number of reads required to satisfy requests for data.

FWRITE
The number of times VSAM was forced to write a buffer to disk to free the buffer for other use.

WRITES
The number of times a buffer was written to disk following PUT processing.

For the SHOW VSAMX command, the second line of information returned by this display is as follows:

D-EXT
The number of extents allocated to the data component of the dataset.

I-EXT
The number of extents allocated to the index component of the dataset.

D-EXCP
The number of EXCP macros that VSAM has issued for access to the data component.

I-EXCP
The number of EXCP macros that VSAM has issued for access to the index component.

See Also: The UDBCTL and SHOW UDB command definitions.

SHOW XMIT

Function: Displays the status of all or selected FTS Transmission Requests.

```
SHOW  XMIT   [ =name ]
             [ DIRECTION={ IN | OUT } ]
             [ CLASS=class ]
             [ STATUS={ ACTIVE | READY | COMPLETED |
                        HELD | FAILED } ]
             [ LINK=linkname ]
(0)
```

Use: This command provides a display of the status of all or selected FTS transmission requests.

If a request name is specified, a detailed display of information relating to the request is produced, including the datasets being transmitted from and to, the status of the transmission and checkpoint information.

If no specific request name is coded, a summary list of transmission requests is provided. This list can be shortened by coding any combination of other operands (DIRECTION, CLASS, STATUS and LINK)

Operands: ***=name***

display a specific transmission request. The resulting display contains detailed information about the request.

If the named request is incoming, operand 'DIRECTION=IN' is also required.

DIRECTION={ IN | OUT }

Restricts a summary display to only INbound, or only OUTbound, transmission requests.

CLASS=*class*

Restricts a summary display to outgoing transmissions of a particular class. Class is in the range A to Z.

STATUS={ ACTIVE | READY | FAILED | COMPLETED | HELD }

Restricts a summary display to only those requests of the given status.

LINK=*linkname*

Restricts a summary display to only those requests defined for the given link.

Examples:

```
SHOW XMIT                    -* List all transmission requests
SHOW XMIT STAT=F LINK=PERTH -* List all failed transmissions
                             -* executed on link PERTH
SHOW XMIT=DAILYREPORT        -* List full details of outgoing
                             -* request DAILYREPORT
SHOW XMIT=ABC DIR=IN         -* List full details of incoming
                             -* request ABC
```

Notes:

When attempting to display full details of a specific incoming request, remember that incoming request names might not be unique. It might be necessary to specify LINK= to uniquely specify the request.

Operands on this command can be abbreviated to any length which maintains their unique recognition (for example, S for STATUS).

See Also:

The TRANSMIT command.

SHOW XNFTRACE

Function: Displays the status of all XNFTRACE requests.

```
SHOW  XNFTRACE

(0)
```

Use: The SHOW XNFTRACE command can be used to list all current XNFTRACE requests. The information returned by this command includes the user ID that initiated the request and the date and time the request was made. In addition the current status, either ACTIVE or PENDING, are displayed.

Notes: This command applies only to MS regions using Hitachi's XNF product.

See Also: XNFTRACE command description.

SHUTDOWN

Function: Commences an orderly shutdown of the system. Also allows the orderly shutdown of the system to be cancelled.

```
SHUTDOWN    [ CANCEL | $ABEND$ | DISC=CANCEL ]
(3)
```

Use: The SHUTDOWN command provides the means for an orderly shutdown of the system. When the SHUTDOWN command is entered the system is placed in *shutdown status*. This prevents further logons and disconnections. When the last user logs off from the MS region, the system automatically initiates the final shutdown process.

The DISC=CANCEL operand can be used to cancel users who are currently logged on but disconnected.

The shutdown process can be reversed by entering the SHUTDOWN command with the CANCEL operand. This cancels the shutdown status and allows logons to be performed.

Cancelling a SHUTDOWN command must be performed before the last user logs off.

As an aid to MS support, the SHUTDOWN $ABEND$ command causes an immediate ABEND of the MS region. Unlike cancelling the MS region, this allows the standard formatted dump and system dumps to be provided. This facility should be used only when requested by your support center.

Operands:

CANCEL
Cancels the previous SHUTDOWN command and allows logons.

$ABEND$
Causes the MS region to immediately ABEND with a 116-00 internal ABEND.

DISC=CANCEL
Commences shutdown and cancels all disconnected users.

Examples:

```
SHUTDOWN
SHUTDOWN CANCEL
```

Notes: Issuing a SHUTDOWN command returns a message indicating the number of active and disconnected users. All user regions must be terminated before the system can terminate. Disconnected users might need to be cancelled by using the DISC=CANCEL operand. Alternatively, issue a SHUTDOWN CANCEL command, to allow the disconnected users time to reconnect and complete their interrupted tasks.

The SHUTDOWN command cannot be abbreviated and must be entered in full.

SHUTDOWN

The STATUS command can be used to display whether shutdown status is in effect.

Terminals operating only under the Network Solicitor facility (EASINET) are not impacted by the SHUTDOWN command and continue to function correctly until the system finally closes down.

The SHUTDOWN $ABEND$ command is useful to force an immediate abend of the MS region when requested by support personnel.

See Also: The FSTOP and STATUS command descriptions.

SIGNOFF

Function: Signs off from a remote system.

```
SIGNOFF        { linkname | DOMAIN=domain-id | SSCP=sscp-name }
(0)
```

Use: The SIGNOFF command can be used to terminate a signon to a remote system.

Operands:

linkname
The name of the remote system. This is the name assigned to the link definition that defines the remote system. Use the SHOW USERS command to determine the names of the remote systems to which you are connected.

DOMAIN=*domain-id*
The domain-ID of the remote system from which you want to signoff.

SSCP=*sscp-name*
The symbolic name of the SSCP under which the target remote system is running.

Examples:

```
SIGNOFF PROD
SIGNOFF MIAMI
```

Notes: If multiple OCS windows are operational (that is, operating in split screen mode) it is possible to be signed on to a remote system from each of the operational windows. The SIGNOFF command terminates a signon from the window where it was issued only. When no operational windows remain at the remote system, the user ID is logged off.

Therefore, SIGNOFF commands must be entered from each of the operational windows from which a SIGNON (or implied signon by the routing of a command) was entered.

Exiting from OCS mode automatically terminates any signon from that window to a remote system.

SIGNOFF also terminates ROF sessions owned by background environments such as the background monitor, background logger, and LOGPROC processor, which are entitled to maintain their own ROF links to other systems provided that their logical user IDs are defined to the remote systems.

See Also: The SIGNON and ROUTE command descriptions.

SIGNON

Function: Signs on to a remote system, or refreshes a user's security profile.

```
SIGNON      [ REGION={ OWN | ALL | APPC } ] |
            [ linkname | DOMAIN=domain-id | SSCP=sscp-name ]
            [ ID=msgid ]
            [ { COLOR | COLOUR }=color ]
            [ { HLIGHT | HLITE }=hlight ]
            [ PAREN={ YES | NO } ]
            [ PASSWORD=password ]
            [ PREFIX={ BEFORE | AFTER | NO } ]
(0/2)
```

Use: More than one system can be active in a single or multi-CPU environment, and can be connected by a facility known as INMC. Once connected, the Remote Operator Facility (ROF) allows an operator in one system to sign on to another system, execute commands in that system, and have the results returned and displayed at the entering terminal.

The SIGNON command has two functions:

- To sign on to another system
- To refresh a user's security profile without having to log off and then sign on to the system again

When using the signon function, the link to the other system must be active for the signon to complete. If the link is not active, the SIGNON command is queued and completes automatically when the link becomes active.

Operands: If no operands are entered, the security profile is refreshed for all logged-on users with the same user ID as that of the user issuing the command. Any changes that have been made to the user's user ID definition are automatically implemented as a result of the refresh. A SIGNON command issued to a background user ID (via a SUBMIT command) refreshes the security profile of the background user ID in the same manner.

REGION={ OWN | ALL | APPC }

The REGION operand is mutually exclusive with all other SIGNON operands. It is used to refresh the user definitions for the indicated set of active users, according to the UAMS definition of the user who issued the command.

REGION=OWN refreshes the user definition for the user issuing the command.

REGION=ALL (the default) refreshes all currently logged on users who have the same user ID as the user issuing the command, including APPC regions.

REGION=APPC refreshes all active APPC regions with the same user ID as the user who issued the command.

linkname

Specifies the name of the remote system. This is the name assigned to the link definition that defines the remote system. Use the SHOW LINKS command to determine the names allocated to the links to remote systems.

DOMAIN=*domain-id*

Specifies the domain ID of the remote system to which the signon is to occur. Use the SHOW DOMAINS command to list the domain names of the attached systems.

SSCP=*sscp-name*

Specifies the symbolic name of the SSCP under which the target remote system is running. As signons based on SSCP names are used to direct commands to VTAM, SIGNON is preferentially directed at a system which has its Primary Program Operator (PPO) interface active.

ID=*msgid*

Messages returned from a ROF session are prefixed by default with the MSGID specified at the time the link was established (see the LINK START command description). The user can override the default prefix with a personal choice of prefix defined by the ID= operand. The valid range is 1 to 8 alphanumeric characters.

{ COLOR | COLOUR }=*color*
{ HLIGHT | HLITE }=*hlight*

Specifies the color and highlighting to be assigned to messages received by the user at the OCS window. COLOR can be specified as RED, YELLOW, BLUE, WHITE, GREEN, TURQUOISE or PINK. HLIGHT can be specified as REVERSE, BLINK, or USCORE.

Color and highlighting assignments override any default color or highlight options specified on the LINK command which was used to establish the INMC link to the remote system. This allows individual users to tailor their own message presentation as required.

PAREN={ YES | NO }

Specifies that if the ROF message prefix is to be added to the message text, it is to be enclosed in parentheses. The default is as set by the SYSPARMS ROFPAREN= command. PAREN=NO specifies that if the ROF message prefix is to be added to the message text it is not to be enclosed in parentheses.

PASSWORD=*password*

Use this operand to specify the appropriate password when your user ID at the remote system carries a password that is different from the password in the local system. The entered password flows in encrypted form to the remote system and is presented to the security exit for verification. The password is deleted from the command entry line and is not recorded on the log.

PREFIX={ BEFORE | AFTER | NO }

Specifies the message prefix to be added to ROF messages received from remote systems. The default is as set by the SYSPARMS ROFPREFX command.

BEFORE specifies that the message prefix is to be added to the message text *before* the message is seen by the user's MSGPROC procedure. This means that the prefix is seen as the first word of the message when it reaches the user's MSGPROC procedure. The prefixed message is seen by *dependent processing environments* created by &INTCMD SIGNON statements issued within a MSGPROC procedure to receive ROF messages from remote systems.

AFTER specifies that the message prefix is to be added to the message text *after* the message is seen by the user's MSGPROC procedure. This means that the prefix is not seen as the first word of the message when it reaches the user's MSGPROC procedure.

NO specifies that the message prefix is to be suppressed and is not added to ROF messages received at this OCS window.

Examples:

```
SIGNON REGION=OWN
SIGNON PROD PASSWORD=PX862
SIGNON TESTSYS ID=TEST COLOR=BLUE PAREN=NO
```

Notes:

If no SIGNON command is entered, then the MS region automatically signs on a user when the first command is routed to the remote destination. However, if the link is not active when the ROUTE command is entered, then the command is rejected.

The SIGNON command queues when the link is not active, and completes automatically when the link becomes active. Use of the SIGNON command is important for operators who are to receive unsolicited messages from a remote system, as a queued SIGNON command completes as soon as the link becomes active. Any unsolicited messages are immediately received without further action by the operator.

The REGION operand is mutually exclusive with all other SIGNON operands.

The authority and privileges assigned to users when they sign on to a remote system are those assigned to their user ID definition in that system. For instance, while users can be authorized for authority level 3 commands in their *local* system, they might be restricted to level 1 commands in the remote system. Any commands routed to the remote system that exceed the assigned authority level are rejected. An attempt to sign on a user ID that is not defined in the remote system is rejected.

If a number of signons to remote systems are always required, then operators should consider establishing an NCL procedure containing the necessary SIGNON commands. This could be defined as the initial command (PROFILE INITCMD) for the user ID, which is automatically executed on entry to OCS.

An operator can have more than one operational OCS window when utilizing split screen mode. The equivalent number of operational windows can also be active at the remote system. Multiple operational windows at a remote system can also be profiled differently, by routing a PROFILE command which specifies the required operands.

On exiting from OCS, any remaining signons to remote systems are automatically terminated.

The term *user* can mean a real OCS user, or a background environment such as the background monitor, background logger, or LOGPROC processor. All of these are entitled to maintain ROF sessions with other systems, provided that their logical user IDs are defined to the remote systems.

When an ROF session exits to a remote system, each command routed to the other system for execution is treated as a separate *transaction*. The results of the command are correlated and returned to the process that issued the command. This means that when a real user enters a ROUTE command from an OCS window, the results of the command are returned to the window. Also, when an NCL process executing in the user's NCL processing environment issues a command (for example, &INTCMD ROUTE), the results of that command flow back to the process's dependent response queue for analysis by &INTREAD. This enables several processes to use the ROF connection to a remote system. Each one can control and identify the answers to its own commands.

See Also: The SIGNOFF, LINK (INMC), LINK (APPC), LINK (NVC), and ROUTE command descriptions.

SPLIT

Function: Enters split screen mode and opens another logical window. If already in split screen mode, adjusts the size of the current logical window boundaries to the new specification.

```
SPLIT          [ ROW=nn ]
               [ COL=nn ]
               [ PSKIP=option ]
(0)
```

Use: The system supports a split screen mode of operation in which more than one logical screen window can be operative. In OCS mode, the SPLIT command is provided to allow the user to assign the *split* function to a particular function key. Outside OCS, this assignment is allocated to F2 and F14. If only one logical window is operative (occupying the whole screen), then a new window is *opened* when the split command is entered. The current window and the new window share the available space on the real display screen. The division of space depends on the location of the cursor when the SPLIT command is executed, or on the ROW and COL operands if specified.

Horizontal and vertical splitting of the screen are supported. A vertical split is requested by having the cursor on the bottom line of the physical screen when the SPLIT command is executed.

When multiple windows are operative, they operate independently of each other. Input is accepted from the window in which the cursor is located.

Operands:

ROW=*nn*
Specifies the screen row at which the split is to take place. If COL is also specified, ROW takes precedence.

COL=*nn*
Specifies the screen column at which the split is to occur. If ROW is also specified, the ROW operand takes precedence.

PSKIP=*option*
Specifies the primary menu selection that will occur for the new window.

Examples:

```
SPLIT
SPLIT ROW=9 PSKIP=U.3
SPLIT COL=25
```

Notes: The SPLIT command is not normally entered from the command line, although this is possible. By default, function keys F2 and F14 are both allocated to the SPLIT command. Outside OCS, the SPLIT function is always associated with F2 and F14. See the *Management Services User's Guide* for more information about splitting your screen.

SPO

Function: Starts or stops the VTAM Secondary Program Operator interface.

```
SPO          { START [ DOMAIN={ domid | * } ] | STOP }
(3)
```

Use: The MS region supports the VTAM SPO interface to initiate VTAM operation commands and receive the response messages.

Entering the SPO START command causes the MS region to attempt the use of VTAM SPO capabilities. Successful operation requires that the SPO attribute has been defined with the MS primary ACB VTAM APPL definition.

Operands:

START [DOMAIN={ *domid* | * }]

Commence operation of SPO interface. If the DOMAIN operand is specified, then all VTAM commands are redirected to the specified domain by the local system.

If a link to the domain *domid* is terminated, SPO on the local system is also terminated. Subsequently, if the link becomes active again, the SPO is automatically activated by the local system and redirected to domain *domid*.

STOP

Terminate operation of SPO interface.

Examples:

```
SPO START
SPO STOP
```

Notes: For successful operation of VTAM commands issued from the MS region, the SPO interface must be active.

See Also: The STATUS command description.

SSI

Function: Performs Subsystem Interface functions.

SSI (0/3)	{ SIGNOFF | STATUS | STOP }

Use: This command can be used to signoff the SOLVE SSI, to determine the status of the SOLVE SSI, or to stop the SSI.

Operands:

SIGNOFF
Indicates that you wish to signoff any virtual environment created in the SOLVE SSI for your current command environment.

STATUS
Indicates that you wish to obtain status information about the SSI.

STOP
Indicates that you wish to stop the SOLVE SSI. The SSI job terminates. This operand requires an authority level of 3.

Examples:

```
SSI STOP
SSI SIGNOFF
```

Notes: Stopping the SSI might cause other facilities, for example, PPI or AOM, to be inactivated.

See Also: The SHOW SSIUSERS command description.

Function: Starts executing an asynchronous NCL process.

```
START        { procname [ parm1 parm2 ... parmn ] | PROC=procname
               [ SERVER=servername
                     [ SCOPE={ REGION | USER | SYSTEM } ] ]
               [ NOTIFY={ NO | YES } ]
               [ PARMS=parm1 parm2 ... parmn ] }
(0)
```

Use: The START command is used to initiate the execution of a new NCL process by invoking the nominated NCL procedure, optionally passing execution parameters. The difference between the START and the EXEC commands is that a process that executes as a result of a START command processes *independently* of and concurrently with any other process executing in the same NCL processing environment, whether initiated by START or EXEC.

This means that an NCL procedure can be STARTed to perform lengthy and complex processing, but the user can still issue EXEC or START commands to run other procedures at the same time. Whereas successive EXEC commands stack, pending completion of the procedures in order, started procedures execute concurrently without having to wait for any other procedure to complete.

Started NCL procedures can issue EXEC statements internally. Such nested procedures operate according to the rules for the EXEC command, and the original started procedure is suspended while any of its nested procedures execute.

A process can START another process, which then executes independently of its originator. Alternatively a process can START another process as a dependent process, by using the &INTCMD statement. In this case the started process executes concurrently with its originator but is terminated if its originator ends.

A started procedure can be given a name by using the PROC= operand. This NCL process is then termed a *server process* and its name is called the *servername.*

Operands: ***procname* | PROC=*procname* (mandatory)**

The name of the NCL procedure to be loaded and executed when the new process is created. The usual procedure name restrictions and library requirements apply.

SERVER=*servername*

The SERVER operand is optional and can be used to specify the (up to 32 character) *servername* under which the new NCL process is to be registered within the scope defined by the SCOPE operand. If a server of that name is not currently registered within the specified scope, the NCL process is started and the *servername* registered. If a server of that name is already registered within the specified scope, the START command is rejected.

SCOPE={ REGION | USER | <u>SYSTEM</u> }

The SCOPE operand is optional and can be used to specify the scope for the registration of *servername*, and hence is valid only if the SERVER parameter is present.

servername must be unique within the scope indicated. This prevents two server processes from executing within the same sphere of control. When a request to start a server process is received, a check is made to determine whether the *servername* is registered, and the process is started. If the name is not unique, then the server process start is rejected.

If an attempt is made to start a server process where the *servername* is currently registered with the scope specified, the START command is rejected. Otherwise, the *servername* is registered within the scope specified, and the server process becomes *visible* to other processes within that scope for the purpose of sending messages.

REGION indicates that *servername* is to be unique within the current region, as defined by a particular connection to Management Services.

USER indicates that *servername* is to be unique across all regions associated with the particular user ID.

SYSTEM indicates that *servername* is to be unique within this MS region.

NOTIFY={ NO | YES }

The NOTIFY operand specifies whether specific notification of process start and end are required to be returned to the command originator.

The default is NO; in this case, no message is issued if the process starts successfully. On normal termination, message N03906 is issued, unless suppressed by a &CONTROL NOENDMSG setting in the terminating process. On abnormal termination, one or more messages are issued to indicate the reason for the failure, and message N03906 is issued regardless of any &CONTROL settings.

```
N03906 type proc PROCESSING COMPLETE.  NCLID nclid.
```

If NOTIFY=YES is specified, then process start and end messages are always issued. After the process has been created, the procedure has been loaded (if necessary), and the first statement in the procedure is to be executed, message N23Q01is issued, indicating successful process start . On normal termination, message N23Q02 is issued; on abnormal termination, message N23Q03 is issued. When operating in this mode, message N03906 is suppressed (except for logging CPU time statistics as requested by SYSPARMS NCLOGTRM=YES); however, all other messages are unaffected.

```
N23Q01 PROCESS STARTED.  NCLID: nclid DOMAIN: domain PROC: proc
```

```
N23Q02 PROCESS ENDED.  NCLID: nclid DOMAIN: domain PROC: proc
Rand so onODE: rand so onode
```

```
N23Q03 PROCESS FAILED.  NCLID: nclid DOMAIN: domain PROC: proc
REASON: reasonCode - reasonText
```

PARMS=*parm1 parm2 ... parmn* | PARMS=*parm1 parm2 ... parmn*

Parameters to be passed to the procedure at the start of execution. If specified, this operand must be the last present. All data is taken as a string of parameters, and used to create the initial parameters (such as &1, &2, &ALLPARMS) in the new procedure.

Examples:

```
START NCPCHECK NCP0020
START PROC=MYPROC SERVER=FRED SCOPE=USER NOTIFY=YES
```

Notes:

A single Management Services user is limited (by default) to 128 procedures executing concurrently, regardless of whether they are initiated by START or EXEC commands. This default can be changed by the SYSPARMS NCLUMAX command.

See Also:

The EXEC, FLUSH, SUBMIT, LIST, NCLCHECK, FSPROC, and SHOW SERVER command descriptions.

The *Network Control Language User's Guide* for details and concepts of serial and parallel processes, dependent processes, and NCL processing environments.

STATUS

Function: Displays current general system status.

```
STATUS
(0)
```

Use: The STATUS command lists the overall system status. Information displayed includes:

- System ID
- Number of users
- Network identifier and VTAM levels (if applicable)
- Status of various system interfaces
- Configuration and product release level information

Additional severe error information can also be included in the display (for example, if activity logging has ceased due to errors).

STRACE

Function: Starts and stops session trace activity.

```
STRACE        { { START=name [ MONITOR | MONITOR=name ] }
                    [ WRITE={ MSG | LOG | ALL } ]
                    [ SELECT={ ALL | NEGRESP } ]
                    [ PIU={ PARTIAL | FULL } ] |
                        STARTALL [={ LU | PU | CDRM } ] |
                        STOPALL [={ LU | PU | CDRM } ] |
                        STOP=name |
                        MONITOR=NO }
              [ QUEUE ]
(4)
```

Use: The STRACE command is the principal command for controlling the activity of NTS session trace functions, providing all the options available for selecting the session trace activity that is required. Session tracing can be started for all resources, or for specific resources, at any stage once session awareness is active.

Operands:

START=*name*

Requests that session tracing be commenced for the resource specified. Multiple concurrent resource tracing is permitted, up to the limit set by the SYSPARMS NTSMAXTR operand value. If the resource named is currently unknown to NTS, the request is accepted and remains pending resource activation. When the resource becomes known to NTS, trace data is collected for all sessions concerning the resource. If the resource subsequently becomes inactive, the trace request is in a pending state, awaiting a subsequent activation. The trace request is remembered until it is either explicitly stopped (with a STRACE STOP=*name* request) or overridden with a global trace request (for example, STRACE STARTALL, or STRACE STOPALL), or until session awareness terminates.

MONITOR [=*name*]

Requests that NTS monitor trace data for the resource *name* specified for this user. Unless the MONITOR operand is specified as part of an STRACE START=*name* command (in which case the resource to be monitored is assumed to be the one for which the trace is being started), the *name* must always be specified with the MONITOR operand. As trace data arrives for the resource nominated, messages formatting the trace information are sent to the user (or as directed by the WRITE operand below). The trace information can be determined by the SELECT operand (see below).

WRITE={ MSG | LOG | ALL }

Valid only with the MONITOR operand, and determines where monitored trace information is directed. If MSG is specified (or defaulted), then session trace monitor messages are written to the user but not delivered to the activity log. If the LOG option is used, then NTS delivers messages to the activity log only; the ALL option delivers messages to both the user and the activity log.

SELECT={ ALL | NEGRESP }

Valid only with the MONITOR operand, and determines whether all trace information is to be monitored, or only PIUs resulting in a negative response being issued. If ALL is specified (or defaulted), then no filtering of monitored trace data takes place. If NEGRESP is specified, then only those PIUs carrying an RU causing a negative response, and the negative response PIUs themselves, are selected for monitoring.

PIU={ PARTIAL | FULL }

Specifies the amount of trace data requested from VTAM for the resource specified. If PARTIAL is specified (or defaulted), then for all PIUs except FMD data, the complete PIU is collected. For FMD data RUs, the TH, RH, and 11 bytes of the RU are collected. If FULL is specified, then the complete PIU is collected.

This operand is valid only with a specific trace request (START=name).

Releases of ACF/VTAM prior to 3.4.1 might not support the collection of the complete PIU.

This operand can be used to stop or start complete PIU tracing while tracing is already active for this resource, including GLOBAL tracing.

STARTALL [={ LU | PU | CDRM }]

Allows session tracing to be started for all resources, namely LUs, PUs, and CDRMs, or all resources of the type specified.

STOPALL [={ LU | PU | CDRM }]

Allows session tracing to be stopped for all resources, namely LUs, PUs, or CDRMs, or all resources of the type specified.

STOP=*name*

Requests that session tracing be stopped for the specified resource. If global tracing is in effect then this resource is excluded from any further global tracing activity.

MONITOR=NO

Terminates any currently active STRACE MONITOR request that this user has outstanding. If the user exits OCS (or issues an &INTCLEAR when running in an internal command environment), then an implicit STRACE MONITOR=NO command is executed.

QUEUE

Specified with one of the above options, results in the request being queued if the CNM ACB is not open.

Examples:

```
STRACE START=LU001 QUEUE
STRACE STARTALL=LU
STRACE START=LU445 MONITOR SELECT=NEGRESP
STRACE STOPALL
STRACE MONITOR=CICS
```

Notes: VTAM rejects any attempt to start a specific trace for an unknown resource. For the request to be accepted by VTAM, the major node containing the resource definition must be active, although the resource itself might be inactive.

Only the most recent STRACE requests within a particular scope are remembered by NTS. For example, consider the case where an STRACE START command is entered for a PU named PU01 and LU named LU99. If a subsequent STRACE STARTALL=LU command is processed, then the request to trace LU99 is overridden by the more powerful request to trace all LUs, and of course if the tracing of all LUs is then terminated, no specific LU traces are active. However, the trace request for PU01 is still current. Similarly, if an STRACE STARTALL command was issued instead of the STARTALL=LU request both specific trace requests would be overridden.

See Also: The SHOW STRACE, DEFCLASS, REPCLASS, DELCLASS, NTSMOD, and SAWARE command descriptions.

SUBMIT

Function: Passes a command to a background environment for processing.

```
SUBMIT        { region-id | USER=userid }
              [ AUTH=auth ]
              [ CMD= ] command text
(0/2)
```

Use: The SUBMIT command allows the user, or an NCL procedure executed by the user, to pass a command to one of the system's *background environments* for execution. The command can be any valid MS or VTAM command, including requests to EXEC or START NCL procedures.

The SUBMIT command propagates the issuing user's command authority.

Operands: ***region-id***

This operand must be supplied as the first operand. It identifies the background region under which the command is to be executed. It must have one of the following keyword values:

BLOG

The *Background Logger*. Commands executed in this region have messages sent to the log.

BMON

Indicates that the following command is to be executed by the *Background Monitor* environment.

BSYS

Indicates that the command is to execute under the *Background System* environment.

BSVR

Indicates that the command is to execute under the *Background Server* environment.

AOMP

The AOMPROC NCL processing region that is established when AOMPROC is active. Messages are sent to all AOM message receivers.

CNMP

The CNMPROC NCL processing region that is established when CNMPROC is active. Messages are sent to all CNM message receivers.

LOGP

Indicates that the command is to execute under the LOGPROC environment.

PPOP

Indicates that the command is to execute under the PPOPROC environment.

USER=*userid*

This operand must be supplied as the first operand. It identifies a signed-on user with an OCS environment under which the command will be executed.

Use of this operand requires a higher level of authority; by default, 2.

AUTH=*auth*

This operand allows you to reduce the effective user authority to a nominated value. The value must be from 0 to 255. If it is greater than your own authority, then it is ignored. If it is less than your authority, then it overrides your authority when the target command is checked.

[CMD=] *command text*

This operand is used to clearly mark the start of the command. It must be the last operand and can be used with any MS or VTAM command.

Examples:

```
SUBMIT BMON START NCPCHECK NCP022
SUBMIT USER=U01245 AUTH=0 CMD=SHOW USERS
```

Notes:

The various background processing environments provide logical OCS terminals and are capable of executing commands and NCL procedures in exactly the same manner as a real OCS terminal can. The SUBMIT command is therefore equivalent to entering the command on the logical keyboard for these environments.

The USER operand allows you to route commands fro immediate execution in a specific user environment. This is similar to (and preferable to) the AT *+0 ROUTE=*userid* CMD=*cmd* command.

Other optional features can provide additional background environments that can accept commands sent to them by SUBMIT; for example, the NEWS feature (CNMP) or the AOM feature (AOMP).

SUBMIT (AOMP)

Function: Passes a command to a background environment for processing.

SUBMIT (0)	AOMP	*command*

Use: The SUBMIT command allows the user, or an NCL procedure executed by the user, to pass a command to one of the system's background environments for execution. The command can be any valid MS, VTAM, or Operating System command, including requests to EXEC or START NCL procedures.

The SUBMIT command propagates the issuing user's command authority.

Operands: **AOMP**
This operand indicates that the command is to execute under the AOMPROC environment.

command
This is any MS, VTAM, or Operating System command.

Examples:

```
SUBMIT AOMP START TAPEMON
```

Notes: The background processing environments provide logical OCS terminals and are capable of executing commands and NCL procedures in exactly the same manner as a real OCS terminal. The SUBMIT command is therefore equivalent to entering the command on the logical keyboard for the environment.

Function: Defines, starts, stops and deletes subsystems.

```
SUBSYS    DEFINE=name
          [ PROGRAM=pgm ]
          [ NOSTART ]
          [ PARMLIST={ OLD | NEW } ]
          [ AUTOCALL={ YES | NO } ]

SUBSYS    START=name

SUBSYS    STOP=name

SUBSYS    DELETE=name

SUBSYS    RELOAD=name

SUBSYS    FORCE=name
(3)
```

Use: This command allows the one-time attachment of a program. NCL procedures can then &CALL this program either:

- Implicitly, by using &CALL *pgmname*, or
- Explicitly, by using &CALL SUBSYS=*ssname*

These requests are queued to the long-running subtask that executes this program.

This allows the called program to open files and obtain storage without the problems that occur when using the standard &CALL/ATTACH facility. For example, files can be left open.

Operands:

DEFINE=*name*

The nominated subsystem is defined by using the DEFINE operand. The program indicated by the name option is loaded into storage. By default, START processing is also performed (see below).

name

This name is the name of the subsystem, and is also the name of the program to load, if the PROGRAM parameter is not specified. The name option must follow standard NAME rules: 1 to 8 characters; the first character must be alphabetic or a national character; the remainder must be alphanumeric or national characters.

The named program is loaded, and, unless the NOSTART option is specified, the subsystem task is started.

PROGRAM=*pgm*

Allows specification of a program name that is different from the subsystem name on the SUBSYS DEFINE. The same edit rules apply as for the subsystem name. This operand can also be specified as PGM=*name*, where *name* is the subsystem name.

Note

If several subsystems share the same program name, ensure that the program is truly re-entrant, as Management Services loads and internally shares a single copy of the program.

NOSTART

This optional operand indicates that an implied SUBSYS START should not be performed as part of DEFINE processing.

PARMLIST={ OLD | NEW }

Indicates the type of parameter list to be built for subsystem communication. OLD (default) means a simple parameter list, the same as the old &CALL parameter list. NEW means the new format parameter list, that contains system, subsystem, user, and other information.

It is strongly recommended that the NEW option be used, as this provides the ability to detect that this is a subsystem, and to have private correlators and Management Services information.

With this form, subsystem start, stop, system stop, and NCL process termination requests are also provided to the subsystem.

AUTOCALL= { YES | NO }

Indicates whether any existing &CALL statements, which have only the program name (&CALL *progname* and not PGM=*pgmname*) are to be automatically rerouted to the subsystem. The default is YES, which means that existing &CALL programs (for example, CCPOSCON) can be DEFINED as a subsystem (defaulting to PARMLIST=OLD) and existing programs can immediately use them. This reduces ATTACH overheads.

Typically this would only be done for existing programs. New systems should explicitly use SUBSYS= on the &CALL statement.

START=*name*

Indicates that you wish to start or restart the named subsystem. The subsystem task is restarted. The subsystem reinitializes.

STOP=*name*

Indicates that you wish to stop the named subsystem. The subsystem task is terminated. This allows it to stay defined, but calls are no longer permitted. It must be started; otherwise an error message is returned.

DELETE=*name*

Indicates that you wish to delete the named subsystem. The subsystem must not be active (that is, not started).

The indicated subsystem is cleaned up. The program is deleted and the definition is removed. Any following &CALL requests to this subsystem (explicit, &CALL SUBSYS=*name*) are rejected.

RELOAD=*name*

Indicates that you wish to load a fresh copy of the subsystem program. The subsystem must not be active (that is, not started). This can be used as an alternative to STOP/DELETE/DEFINE.

The program is deleted (provided that it has no other users), and a new copy is requested. The subsystem can then be restarted using SUBSYS START.

Note

If the program has other users, the DELETE does not work, and the reload still obtains the same version as before.

FORCE=*name*

Sometimes a subsystem can hang in the user code. This could be a loop or an unsatisfied WAIT. Regardless of this, all NCL processes waiting on a call can wait indefinitely. This is especially likely while developing the SUBSYS code.

To prevent this problem, the SUBSYS FORCE command is provided. This command can be used whenever the subsystem is forcible. That is, whenever the subsystem subtask is executing user-code. This can be determined by the SHOW SUBSYS command.

The result of the SUBSYS FORCE is a force-detach of the subsystem. It is cleaned up and then assumes the STOP status.

Examples:

```
SUBSYS DEFINE=CCPOSCON PARMLIST=OLD AUTOCALL=YES
SUBSYS DEFINE=USERSYS1 PROGRAM=UPGM001 PARMLIST=NEW
SUBSYS STOP=USERSYS1
```

Notes:

Subsystems allow a program to run continuously, so that it can remember information across calls. Each subsystem has its own task. All &CALL requests to a particular subsystem are queued to the subsystem task. The program is given the requests in order.

When writing a subsystem, the following should be considered:

- Use the new format parameter list. Remember that you can validate the format by checking that the value pointed to by R1 on entry is equal to R1. This parameter list format contains much useful information.
- Several facilities, such as an NCL process cleanup call, initialize, and system shutdown calls, as well as private correlators, are available only if the new format parameter list is used.

- Validate that you have been called as a subsystem by checking the subsystem name field in the (new) parameter list for an asterisk. If an asterisk is found, then this is an &CALL attach, not a SUBSYS call.
- A subsystem gets a private correlator word for each unique NCL process that calls it. This correlator word can be altered (it starts as 0), and is remembered and returned on the next call for that NCL process. It is an ideal place to hang process-related control blocks.
- The shared correlator is shared with all subsystems or other programs called by &CALL for this NCL process.
- Any NCL process that calls a subsystem (with PARMLIST=NEW specified) causes a cleanup call to be provided to that subsystem. This call is made regardless of the type of NCL process termination (normal, abnormal, or condition code). The private correlator is provided, and this allows the subsystem to clean up any control blocks as required.
- If an NCL process is flushed while on a SUBSYS call, the subsystem is not notified of the fact until later. It need not worry about the process disappearing while actually processing the current call.
- A SUBSYS DEFINE (without the NOSTART operand specified) or SUBSYS START command causes the subsystem to be given an initialization call. This allows the subsystem to initialize its environment. It should anchor any control blocks in the SUBSYS correlator (in the new format parameter list).
- A SUBSYS STOP provides the subsystem program with a terminate call. This allows the subsystem to clean up any control blocks or work areas prior to being detached.
- If the MS region is shut down (FSTOP or SHUTDOWN after the last user logs off), the subsystem is provided with a system shutdown call. It must quickly clean up its environment. It is force-detached several seconds later regardless of whether it is finished or not.
- Only as many parameters as were passed can be returned to a caller. Each parameter is limited to 256 characters of data. Binary values are preserved both on input and on return. Be sure to set the lengths correctly.

Note

All calls to a specific subsystem are queued and processed in turn. This means that, if the subsystem code takes a long time to process a call (for example, issuing a WAIT on VTAM input), all pending calls are also delayed. This is a restriction of the current implementation, but can be avoided by queuing requests to the subsystem, to a manager process (using the INTQ command and a globally known NCLID), and the manager process talking to the subsystem on behalf of all users of the subsystem. Then the subsystem code can wait on many events and return the first event to the manager, which can then pass it back to the requestor.

- If the subsystem ABENDs, the subsystem stops. It can be restarted.
- Remember that SUBSYS FORCE is useful when a subsystem hangs.

- The subsystem load module must reside in a library accessible to the MS region. Typically this is the STEPLIB. Remember that the MS region libraries must be authorized. This means that any libraries in this STEPLIB concatenation must also be authorized. This is a security consideration.
- 31-bit mode programs are supported. All supplied control blocks (for example, PARMLIST) are always below the 16 mb line.
- If performing multiple functions, use the first parameter passed as a function indicator:

```
&CALL SUBSYS=SUB1 OPEN ....
&CALL SUBSYS=SUB1 READ ....
&CALL SUBSYS=SUB1 READ ....
&CALL SUBSYS=SUB1 CLOSE ...
```

See Also: The SHOW SUBSYS command description.

The &CALL verb description in the *Network Control Language Reference.*

The chapter about the SUBSYS facility in the *Management Services Administrator Guide*.

SUSLOGON

Function: Suspends an entry or entries in the DEFLOGON table.

```
SUSLOGON    { INPUT=entryname | APPL=applname }
            [ TEXT=msgtext ]
(3)
```

Use: This command is used to temporarily suspend logons from EASINET or MAI to a particular application. Optionally, a message can be supplied that will be displayed on users' terminals if they attempt to use the suspended entry or entries. The command can be used to stop users logging on to an application.

Operands:

INPUT=*entryname*

The entry name of a defined DEFLOGON entry which is to be suspended. Defined entries can be displayed using the SHOW DEFLOGON command. This is used if just a single entry is to be suspended. Users who attempt to logon to an application by entering the text which matches *entryname* are denied access.

INPUT can be abbreviated to IN.

APPL=*applname*

Logons to the specified application are suspended. Multiple DEFLOGON entries can exist that allow users to log on to an application in different ways. This operand causes suspension of all such entries.

TEXT=*msgtext*

The text of a message to be displayed at a user's terminal if they attempt to log on to the suspended application. The text can be up to 50 characters in length, and must be enclosed in single quotes (') if it contains any blanks. If the TEXT operand is not entered, then the following message is displayed:

```
N20213 LOGON TO SELECTED APPLICATION IS SUSPENDED.
```

Examples:

```
SUSLOGON APPL=IMS TEXT='IMS SCHEDULED DOWN UNTIL 15.00'
SUSLOGON INPUT=TSOA
```

Notes: Entries can be reactivated by using the ACTLOGON command.

See Also: The ACTLOGON, DEFLOGON, DELLOGON, REPLOGON, and SHOW DEFLOGON command descriptions.

SWAP

Function: Reverses the dimensions of the current logical windows from horizontal to vertical, or vice versa.

SWAP (0)	[PSKIP=*menu option*]

Use: The system supports a split screen mode of operation in which more than one logical screen window can be operative. In OCS mode, the SWAP command is provided to allow the user to assign the SWAP function to a particular function key. Outside of OCS, this assignment is fixed and allocated to F9 and F21.

The SWAP command swaps the amount of space allocated to one window (that is, its dimension in rows and columns) with the amount of space allocated to the other window.

The window positions do not change.

It is possible to have one window completely closed. In this case the SWAP command closes the current window completely and allows the other window to occupy the entire screen area.

If you are operating with a single window open, pressing the SWAP key automatically opens the second window, positioned to display the primary menu.

Operands: **PSKIP=*menu option***

You can specify the primary menu select that is to be invoked in the target window. On completion of the swap, the new window is positioned in the chosen primary menu option.

Examples:

```
SWAP
SWAP PSKIP=D.C.H
```

Notes: The SWAP command is not normally entered from the command line, although this can be done. By default, function keys F9 and F21 are both allocated to the SWAP command.

The SWAP function is always associated with F9 and F21 in modes other than OCS.

SYSCMD

Function: Issues a command to the local operating system and returns associated response messages to Management Services.

For z/OS, OS/390, MSP, or VOS3:

```
SYSCMD      [ CON={ n | MASTER } ]
            [ WAIT=nnn ]
            [ MIGID={ * | ASIS | DEFAULT | NO | YES | EXIT } ]
            [ DATA= ] command-text

SYSCMD      OPT=LOCK
            [ WAIT=nnn ]
            [ MIGID={ * | ASIS | DEFAULT | NO | YES | EXIT } ]
            [ DATA= ] command-text

SYSCMD      OPT=REL
(2)
```

For VM:

```
SYSCMD      [ DEST={ PROP | GCS | VMOP } ]
            [ DELAY=n ]
              DATA=command-text
(2)
```

Use: In OCS mode an operator can use the SYSCMD command to send an operating system command to the operating system and have the resulting message responses delivered back to the originating OCS window.

If an NCL procedure uses &INTCMD to issue SYSCMD from a *dependent processing environment*, then the command can be used to send an operating system command to the operating system and have the resulting message responses delivered back to the NCL procedure. The procedure retrieves the responses by using &INTREAD. This technique allows NCL processes to carry out unattended operating system commands.

For z/OS, OS/390, MSP, and VOS3

The three versions of the the SYSCMD syntax shown for the z/OS, OS/390, MSP, and VOS3 environments function as follows:

- The first version of SYSCMD is used to issue a command. You can optionally indicate a specific console, or the use of a master console. You can also specify the migration ID option, and an optional wait period to apply if no consoles are available.
- The second version of SYSCMD is used to obtain a console and lock it to the environment. You can optionally issue a command. You can also specify the migration ID option, and an optional wait period to apply if no consoles are available.

- The third version of SYSCMD is used to release any console associated with an environment.

Operands: **CON={ *n* | MASTER }**

This operand, valid on SYSCMD with no OPT or WAIT specified, allows you to do either of the following:

- Issue a command from an arbitrary OS/390 console ID (ranging from 1to the maximum ID on your operating system). This command is in the format SYSCMD CON=*n*.

  No messages are returned from OS/390 for commands issued in this way, and the environment is not associated with the console.

  This option is not supported if extended MCS consoles are in use.

- Issue a command as if it is being entered from the current master console. This command is in the format SYSCMD CON=MASTER.

  The environment is logically associated with the master console, and for a time period (set by the AOMCTO3 SYSPARM), all messages routed to the master console are copied to this environment.

  You must have MASTER console authority in the AOM UAMS definition to be able to use this option. If extended MCS consoles are in use, this operand is ignored. However, because the normal console environment is given MASTER authority, you can still use this option.

WAIT=*nnn*

When you are requesting a console (and/or issuing a command), this optional operand indicates the time (that is, up to *nnn* seconds) that you are prepared to wait for a console to become available.

If this operand is not specified, and no consoles are available, then an immediate error message is returned.

If a non-zero value is specified, and no consoles are available, then the request is queued, and, if a console becomes available, then processing completes. If the period expires, then the standard error message is returned.

If a SYSCMD WAIT=*nnn* is outstanding, no other SYSCMD commands (except SYSCMD OPT=REL, which cancels the wait) are permitted.

The maximum value for *nnn* is 300.

MIGID={ * | ASIS |DEFAULT | NO | YES | EXIT }

This operand indicates whether a migration ID is needed.

Note

If the current console type (AOMCTYPE) is not EXTMCS, then this operand is logically ignored, though its syntax is checked.

The presence or otherwise of a migration ID for a console can be set only when the console is acquired. For SYSCMD, this means that the migration ID setting for the console depends on the effective setting of this operand for the SYSCMD command that causes a console to be acquired. This is a SYSCMD that is issued (optionally with OPT=LOCK) when no console is currently associated with the environment.

If you explicitly attempt to switch from having a migration ID to not having one (or vice versa), then an error is raised.

The values for the MIGID operand are:

*** | ASIS** Indicates that the migration ID setting is to be used as is; that is, if a console is already acquired, then the current setting is to be used. If a console is acquired as a result of this command, then the system default setting (as set by SYSPARMS AOMCMIGI=NO/YES) is to be used. This is the default if no value is specified; it is the same as MIGID=DEFAULT.

DEFAULT Indicates that the migration ID setting for this console, if not currently acquired, is to be the same as the SYSPARMS AOMCMIGI setting.

If a console is currently acquired, then the value derived from the SYSPARMS AOMCMIGI setting is used. If this is different from the acquired console setting, then an error is raised.

NO Indicates that no migration ID is needed. When a console is acquired, no migration ID is assigned. If a console has already been acquired, and it has a migration ID, then an error is raised.

YES Indicates that a migration ID is needed. When a console is acquired, a migration ID is assigned. If a console has already been acquired, and it does not have a migration ID, then an error is raised.

EXIT Indicates that the need for a migration ID is to be determined by calling an exit. You can specify MIGID=EXIT only if a command is also specified.

The command text is passed to the MS migration ID determination exit. This exit makes a conservative decision and logically sets NO or YES. If a user exit is active (SYSPARMS AOMCMIGX=*name*), then it is called, and it changes the MS setting.

The resultant setting from the exit(s) is used as the logical value for the MIGID operand for this command.

If no console is being acquired, then the setting is used to determine whether a migration ID is assigned. If a console has already been acquired, and the migration ID setting is different, then an error is raised.

DATA=*command-text*

This is the text of the command to be sent to the operating system for execution.

In z/OS or OS/390, if no other operands are specified and a previous SYSCMD OPT=LOCK was not executed, then a temporary console is acquired (if one is available) and the command is submitted to the operating system.

The command specified by *command-text* fails if no consoles are available.

In VM, the command is queued to PROP or GCS for execution.

Note

DATA is optional, but is recommended to prevent a possible syntax error if the command text contains a SYSCMD keyword.

OPT=LOCK

This operand requests that a console be associated with this environment (if one is not already), and that it be locked to the environment. That is, no timeouts are applied.

If a console is not locked, then the AOMCTO1 and AOMCTO2 SYSPARM settings, together with the last command issue time and last WTO receipt time, are used to time out ownership of the console. When the timeout expires, the console is released from the environment and becomes eligible for assignment to other environments.

The console remains associated until either a SYSCMD OPT=REL is issued, or the environment is terminated (for example, by pressing PF3 in OCS).

OPT=REL

This operand requests that any console, including a locked console, associated with this environment be released. Using this option does not produce an error message if no console is associated with the environment.

DEST={ PROP | GCS | VMOP }

For VM, indicates where the command is to be executed.

PROP (the default) routes the command to the PROP virtual machine for execution there.

GCS issues the command under the GCS environment of the virtual machine where the MS region is executing.

VMOP routes the supplied command to the VMOPERATOR machine for possible execution.

Users must be authorized in their user ID definition for the specified destination.

DELAY=*n*

Specifies a delay, from 0 to 60 seconds, after issuing the command, to allow all responses to be returned. This delay time is added to the value of AOMCTO1 (for DEST=PROP), AOMCTO2 (for DEST=GCS), or AOMCTO3 (for DEST=VMOP) for this command.

If you know that this command will take a significant time to execute, you should specify a non-zero value here. Failure to do so results in a warning message being issued by AOM to monitor users about SYSCMD command timeouts. This is because, in the VM environment, commands sent to PROP or GCS are single threaded, and any delay in processing one command delays the processing of all following commands.

Notes: ***General***

AOM must be started, and not paused, for the SYSCMD command to be accepted.

No validation of the command text is performed by AOM.

Notes: ***For z/OS, OS/390, MSP, and VOS3***

In the z/OS, OS/390, MSP, and VOS3 environments, processing of the SYSCMD command is now *asynchronous*. This means that, if you issue the command from an NCL process, either natively or in an &INTCMD environment, then you must check the results before destroying the environment (for example, by an INTCLEAR). Failure to do this causes unpredictable results. If the environment is destroyed before the command is issued to OS/390, then the command is not issued.

In addition, if migration IDs are in use (that is, EXTMCS consoles are being used), then all SYSCMD uses for a particular environment must be consistent. The SYSCMD command that creates a new console environment sets the use or otherwise of a migration ID. This setting is done in either of the following ways:

- Explicitly, by using the MIGID=NO/YES/EXIT operand
- Implicitly, by using the MIGID=DEFAULT/* operand, or by omitting it (so that the AOMCMIGI SYSPARM setting is used)

Any following SYSCMD commands that use the same console environment must be consistent with this setting. If they explicitly specify the use or otherwise of a migration ID (that is, the command has MIGID=NO/YES/EXIT coded), then the explicit setting must agree with the environment setting (for MIGID=EXIT, the explicit setting for this command is determined by the exit).

A console is temporarily assigned to a window when the SYSCMD command is entered without the OPT=LOCK operand. The console remains assigned to the window until the console timeout period expires, at which time it is released by AOM and returned to the console pool. See SYSPARMS operands AOMCTO1 and AOMCTO2 for details on console timeout.

The SYSPARMS CONSOLES=*nn* command specifies how many consoles AOM attempts to obtain from the available console pool for use by the SYSCMD command.

In VOS3, all commands are translated to uppercase before being passed to VOS3.

Notes: ***For VM***

In VM, if the number of command requests specified by the SYSPARMS AOMPCLIM parameter is already queued to either PROP or GCS, then the command is rejected with an error message.

In the VM environment, the SYSCMD command functions *synchronously.*

See Also: SYSPARMS AOMCMIG1, AOMCMIGX, AOMCTO1, AOMCTO2, AOMCTO3, and AOMPCLIM operands in the *Management Services Administrator Guide*.

SYSLOG

Function: Issues commands to browse the VMOPERATOR system log. (VM only)

```
SYSLOG        [ OPT=CONN [ DATA=command-text ] | OPT=REL ]
(2)
```

Use: In OCS mode an operator can use the SYSLOG command to issue commands to VMOPERATOR to retrieve information from the VMOPERATOR system log file.

Operands:

OPT=CONN
Establishes a session with VMOPERATOR and access to the current system log file. If the DATA= operand is specified, this operand issues the command provided.

DATA=*command-text*
The text of the VMOPERATOR log browse command. The DATA= operand is required if command-text is specified.

OPT=REL
Terminates the session with VMOPERATOR. The DATA= operand is not allowed when this operand is specified.

Notes: See the VMOPERATOR documentation for the list of valid log browse commands available.

Once a session has been established with VMOPERATOR, it remains in place until SYSLOG OPT=REL is specified or until AOM or VMOPERATOR terminates.

SYSMON

Function: Sends data to System Monitor in 3600/4700.

```
SYSMON        puname
              [ TYPE={ CMD | LOGON | MSG | ATTN } ]
              [ text ]
(1)
```

Use: This command is used to log on to the System Monitor residing in a 3600 or 4700 controller, and then to send data to the Monitor.

Operands:

puname

The node name of the 3600/4700 physical unit with which communication is to take place.

TYPE={ CMD | LOGON | MSG | ATTN }

Sends a command to the System Monitor. The text of the command to be sent should follow this operand.

TYPE=CMD is the default if the TYPE operand is omitted but text is supplied.

TYPE=LOGON logs on to the System Monitor. The logon password should follow this operand.

TYPE=MSG sends a message to the 3600/4700 control operator, if one is logged on. The text of the message to be sent should follow this operand.

TYPE=ATTN sends an attention interruption. Used to request the next page of information if a command generates multiple pages of output. TYPE=ATTN is the default if the TYPE operand is omitted and no text is supplied.

text

The command, message, or password to be sent.

Examples:

```
SYSMON PU3600 TYPE=LOGON 12345
SYSMON PU3600 TYPE=CMD 031 0201
SYSMON PU3600 010 9001
SYSMON PU4700 (page request)
SYSMON PU3600 TYPE=ATTN (page request)
SYSMON PU4700 T=MSG System back up
```

Notes: The TYPE operand can be abbreviated to T.

Responses to commands are displayed on the issuer's terminal. The first line is prefixed with the name of the controller which sent the response.

SYSPARMS

Function: Defines/changes system default values.

```
SYSPARMS    operand=value operand=value
            operand=value...
(4/9)
```

Use: The SYSPARMS command is normally used only by Systems Programmers. It is, therefore, documented in the *Management Services Administrator Guide*, and in the relevant product implementation and administration guides.

TCPIP MODIFY

Function: Alters the current trace options setting of the LOG.

```
TCPIP MODIFY      [TRACE={NO|YES|SUMMARY}]
                  [DNR={VENDOR | SOLVE }]
(4)
```

Use: This command is used to alter the current TCP/IP trace options setting without having to stop and restart the interface.

Operands:

TRACE={NO |YES|SUMMARY }

Specifies whether the tracing of calls to TCP/IP is required. If YES is specified, then the trace records are written to the LOG. If SUMMARY is specified, then only the first 32 bytes of sent/received data is written to the LOG.

DNR={VENDOR | SOLVE }

You can use TCPIP MODIFY to change the value only when TCP/IP is stopped. You cannot change the DNR setting while the TCP/IP interface is active.

If the DNR operand is omitted from a TCPIP START command, then the previously set value is used.

TCPIP QUIESCE

Function: Closes the TCP/IP interface.

```
TCPIP QUIESCE
(4)
```

Use: This command is used to close the TCP/IP interface in an orderly manner. No new sockets are opened and existing open socket connections are allowed to finish before the interface is closed.

TCPIP START

Function: Initializes TCP/IP services.

```
TCPIP START      [ TYPE=interface-name ]
                 [ TRACE={ NO | YES | SUMMARY }]
                 [ DNR={ VENDOR | SOLVE }]
                 [ OPTIONS type-specific operands ]
(4)
```

Use: This command is used to initialize TCP/IP services.

Operands: **TYPE=*interface-name***

Defines the specific TCP/IP vendor interface that is to be used. In most cases this operand can be omitted, because the MS region attempts to automatically pick the correct interface based on the current operating system and environment.

Valid types are:

IBM	Interfaces to IBM MVS TCP/IP V3.2 or Communications Server using HPNS (High Performance Native Sockets)
IUCV	Interfaces to IBM MVS TCP/IP V3.1 or V3.2 using MVS-simulated IUCV (Inter User Communication Vehicle)
TCPAXS	Interfaces to TCPaccess Version 4.1 or later
VM	Interfaces to IBM VM TCP/IP
XNFTCP	Interfaces to Hitachi XNF/TCP
TISP	Interfaces to Fujitsu TISP

If your site has more than one of the TCP/IP facilities, the MS region chooses a default in the order IBM, IUCV, TCPAXS. If this results in the wrong default, you can specify the correct one by using the TYPE operand.

Omitting the TYPE operand results in the last specified (or the default) type being used.

When using the TCPIP DEFINE command, this operand supplies the name of a new interface.

TRACE={ NO | YES | SUMMARY }]

Specifies whether the tracing of calls to TCP/IP is required. If YES is specified, then the trace records are written to the LOG. If SUMMARY is specified, then only the first 32 bytes of sent/received data is written to the LOG.

DNR={VENDOR | SOLVE }

Specifies whether the TCP/IP interface is to use a vendor Domain Name Resolver (DNR) or the SOLVE DNR. When the MS region initializes, the default for the DNR operand is VENDOR (for compatibility).

If the DNR operand is omitted from a TCPIP START command, then the previously set value is used.

You can use TCPIP MODIFY to change the value only when TCP/IP is stopped. You cannot change the DNR setting while the TCP/IP interface is active.

If you specify DNR=SOLVE, then you can use the DNR command to control the operation of the SOLVE DNR.

OPTIONS *type-specific operands*

If a TCPIP START command is issued with no options, it uses the previous options for the specified interface.

After the OPTIONS operand, all following data on the TCPIP START command is treated as options for the specified interface. Each TYPE has a separate options string. You can reset the options for a type to null by supplying the OPTIONS operand with no following data.

The type-specific operands are as follows:

For IUCV or IBM:
[PREFIX={TCPIP | *high level qualifier of TCPIP.DATA dataset*]

For TCPAXS:
[SSID={ACSS | *subsystemid of TCPaccess* }]

For TISP:
[APPL=*applname*]
[TCP=(*aaa*,*nnnnn*,*mmmm*)]
[UDP=(*aaa*,*nnnnn*,*mmmm*)]

For XNFTCP:
[UCE=*nnnnnnnn*]

[APPL=*applname*] (TISP only)

The name of the main TISP APPL to be used by the MS region. This APPL is opened when a TCPIP START command is entered. The APPL name is 1 to 8 characters, and must be defined in the KGTISP*xx* member of the TISP parameter library. The default, if this operand is omitted, is SOLVAPPL.

PREFIX=*tcpip_high_level_qualifier* (IBM TCP/IP only)

Defines the high-level qualifier of the TCPIP.DATA dataset. For example, if PREFIX=TCPIP.V3R2, then the TCPIP START command looks for a client configuration dataset called TCPIP.V3R2.TCPIP.DATA.

Note

Rather than using PREFIX= , you should allocate the client configuration dataset with a DD name of SYSTCPD.

SSID=*subsystemid of TCPaccess* (TCPaccess only)

Identifies the TCPaccess subsystem to be used.

TCP=(*aaa,nnnnn,mmmm*) (TISP only)

Defines the TISP APPL names to be used by the MS region for TCP communication. The default, if this operand is omitted, is TCP=(SLT,5000,100).

The meanings of the variables in this operand are as follows:

aaa Represents a 3-character prefix which is to be used for TCP APPL names.

nnnnn Represents the first APPL number which is to be used for TCP communications where no port number is specified. It must be in the range 1000 to 32760.

mmmm Represents the number of APPLs which are available for TCP communications where no port number is specified. It must be in the range 1 to 1000.

The APPL name used by the MS region for TCP communication is derived from the 3-character prefix followed by a 5-character suffix which consists of the port number, right-justified and padded on the left by zeros. For example, specifying PORT=1234 results in APPL SLT01234 being used. Specifying PORT=0 results in APPL SLT05000 being used.

All TCP APPLs to be used by the MS region must be defined in the KGTISP*xx* member of the TISP parameter library.

UCE=*nnnnnnnn* (XNF/TCP only)

The UCE name of the XNF/TCP subsystem. This operand is not normally needed, as the interface determines the correct UCE name automatically. Specification of this operand overrides the automatic determination and can be used if the automatic determination fails.

The UCE name is 1 to 8 characters, PDSNAME format. It must be the UCE name of XNF/TCP as defined in the TCPDF member of the XNF/TCP parameter library.

UDP=(*aaa,nnnnn,mmmm*) (TISP only)

This operand defines the TISP APPL names to be used by the MS region for UDP communication. The default, if this operand is omitted, is UDP=(SLU,5100,100).

The meanings of the variables in this operand are as follows:

aaa Represents a 3-character prefix which is to be used for UDP APPL names.

nnnnn Represents the first APPL number which is to be used for UDP communications where no port number is specified. It must be in the range 1000 to 32760.

mmmm Represents the number of APPLs which are available for UDP communications where no port number is specified. It must be in the range 1 to 1000.

The APPL name used by the MS region for UDP communication is derived from the 3-character prefix followed by a 5-character suffix which consists of the port number, right-justified and padded on the left by zeros. For example, specifying PORT=1234 results in APPL SLU01234 being used. Specifying PORT=0 results in APPL SLU05100 being used.

All UDP APPLs to be used by the MS region must be defined in the KGTISP*xx* member of the TISP parameter library.

Examples:

```
TCPIP START TYPE=IBM TRACE=SUMMARY OPTIONS PREFIX=SYS1.TCPIP
```

See Also: The TCPIP STOP command description.

The DNR command description.

TCPIP STOP

Function: Terminates TCP/IP services.

```
TCPIP STOP
(4)
```

Use: This command is used to terminate TCP/IP services.

TELNET

Function: Starts a Telnet connection.

```
TELNET      { ? | host_name }
            [ PORT={ 23 | n } ]
            [ ID={ host-name | name } ]
            [ CTRL={ ¢ | c } ]
            [ ENTER={ CRLF | CR } ]
            [ MODE={ LINE | OCS | FS } ]
            [ TRACE={ NO | YES } ]
            [ TRTAB=name ]
            [ COLOR | COLOUR=color ]
            [ HLIGHT=highlight ]
```

Use: The TELNET command is used to connect to remote hosts and routers.

Operands:

?

Displays help in the form of the TELNET command syntax.

host_name

The host name or IP address of the host to connect to.
(*Mandatory*)

PORT={ 23 | *n* }

The port number on the remote host. Port 23 is the default port number for TELNET.

ID={ *host_name* | *name* }

An identifier that is associated with the Telnet connection. By default, this is the name of the host to which you are connected. Use this ID on any subsequent TNSEND, TNCMD, or TNDISC commands.

CTRL={ ¢ | *c* }

The CTRL character is used for the following distinct purposes:

- To nominate the character to use for simulation of the Ctrl key; for example, ¢C becomes Ctrl-C.
- To send a line without the usual ENTER character (CRLF or CR). For example, you might want to send a single space in response to a --More-- prompt. To do this, press the space bar, type ¢, and press ENTER.

- To send characters that are otherwise not supported on 3270 keyboards:

To send...	Enter...
[	¢{
]	¢}
Del (X'7F')	¢#
NUL (X'00')	¢0
Esc (X'1B')	¢2
X'1C'-X'1F'	¢3-¢6

ENTER={ CRLF | CR }

When a line is sent to the remote host, the specified sequence is added to the end to indicate the end of the line. By default CRLF is sent, but you can choose to send a CR alone if necessary.

Note

To send data without the end-of-line character, end the line with the CTRL character (usually ¢).

MODE={ OCS | LINE | FS }

Determines whether the OCS window is *taken over* (that is, put into a mode) by the TELNET command. When used in an OCS window the default value is OCS, otherwise the default value is LINE. The FS value is Full-screen mode and gives the same result as entering the command $TNCALL ACTION=TELNET.

In modal OCS processing, the connection ID is the OCS window ID (displayed at the bottom right of the screen), the Enter key is defined to send the entered data on the connection, F12 is defined as the standard Enter key. F3 ends the connection and returns OCS function key assignments to normal.

TRACE={ NO | YES }

Causes the data transmitted and received on the connection to be traced to the activity log.

Note

The data is displayed in dump format where the hex data is in ASCII and the display part is in EBCDIC.

TRTAB=*name*

Allows a specialized ASCII-EBCDIC-ASCII translation table to be used. If TRTAB is not specified a default translation table is used. If specified, the name must be the PDS member name of an entry in the dataset specified by the $IPINIT TNTRDSN operand (the default is *prefix*.SEZATCPX). This dataset contains the *source* form of the translation tables used by IBM TCP/IP or TCPaccess. If the specified table is not found, a warning message is issued and the connection continues, using the standard translation table.

COLOR | COLOUR=*color*

Specifies a color attribute to be associated with messages arriving from a remote host. The color can be any one of red, white, blue, green, yellow, pink, or turquoise.

HLIGHT=*highlight*

Specifies a highlighting attribute to be associated with messages arriving from a remote host. This can be any one of REVERSE, USCORE, or BLINK, providing reverse video, under-scored, or blinking highlighting respectively. (The HLIGHT operand can also be specified as HLITE).

Examples:

```
TELNET router1
TELNET router1 ID=FRED COLOR=PINK
```

Notes:

You can send data on the connection by using the TNSEND command, send a command on the connection by using TNCMD, and end the connection by using TNDISC.

The TELNET command is implemented by a command equate. You can use the SHOW EQUATES command to display it.

See Also:

The TNSEND, TNCMD, and TNCTL commands.

TERMINAL

Function: Provides extended color and/or highlighting datastreams to terminals which support these features but which were bound to MS without such attributes.

```
TERMINAL    [ { COLOR | COLOUR }={ YES | NO } ]
            [ { HLIGHT | HLITE }={ YES | NO } ]
            [ TYPE={ STD | FJ | FJ7 } ]
            [ DBCS={ YES | NO } ]
            [ INPUTCTL={ STD|NOBSP } ]
(0)
```

Use: The MS region can send extended color and/or extended highlighting datastreams, and does so automatically to terminals that support these functions provided that the system can determine the device capabilities from the BIND parameters. If a color terminal logs on to the MS region but the BIND parameters do not indicate the device's attributes correctly the terminal will not be used in full color mode. In this case the TERMINAL command can be used to override this fact. Extended datastreams will subsequently be sent to he device. Conversely, the TERMINAL command can be used to prevent the MS region from sending such extended datastreams.

A terminal operating in a LU1 region can use the TERMINAL command to control x'16' being interpreted as a backspace.

Operands:

{ COLOR | COLOUR }={ YES | NO }
Indicates whether or not you desire extended color datastreams to be sent to your terminal.

{ HLIGHT | HLITE }={ YES | NO }
Indicates whether or not you desire extended highlighting datastreams to be sent to your terminal.

TYPE=STD | FJ | FJ7
Indicates whether the terminal is to be treated as an IBM 3270 (STD), Fujitsu 3 color (FJ) or Fujitsu 7 color (FJ7) terminal for DBCS support. If you use this operand, make sure that your choice is correct. It is relevant only if you also use the DBCS= operand. If your choice is wrong results are unpredictable and session termination is likely.

DBCS=YES | NO
Indicates whether the terminal is to be considered capable of receiving or sending *double byte character set* datastreams. If DBCS=YES is entered, the DBCS implementation will be determined by the TYPE operand. If the wrong terminal type is specified DBCS support is likely to fail.

INPUTCTL={STD|NOBSP}
In an LU1 region this indicates whether backspace interpretation is in effect. If INPUTCTL=NOBSP is entered x'16' will not be interpreted as a backspace. In non LU1 regions INPUTCTL=NOBSP is not supported. The setting will produce an error message. INPUTCTL=STD should be used in these regions.

TERMINAL

Examples: `TERMINAL COLOR=YES HLIGHT=NO`

Notes: If you specify that you desire either form of extended datastreams and your terminal does not actually support the feature screen results are unpredictable and your session could even be terminated through errors.

TIME

Function: Sets and displays the current date and time of day and time zone information.

TIME (0)

Use: The TIME command is used to set the region time from the system clock. It can be used after the system clock has been changed.

If the command detects that the system clock has been set forward, it executes all AT commands that would have executed during the period lost by the change in time.

The TIME command also displays the local time and date, Greenwich Mean Time and date, and the user's time, date, and time zone offset.

Examples: This is an example of the TIME command display:

```
N11001 LOCAL TIME: 17.12.25   THU 03-MAY-2001   94.125   +1000
N11002 GMT TIME:   07.12.25   THU 03-MAY-2001   94.125
N11003 USER TIME:  17.12.25   THU 03-MAY-2001   94.125   +1000 AUSESTDT
```

TIMEZONE

Function: Maintains time zone names and offsets.

TIMEZONE	ADD=*name* OFFSET=*shhmm* [DESC=*description*]
TIMEZONE (3)	DEL=*name*

Use: The TIMEZONE command registers a time zone name and the associated Greenwich Meantime Time (GMT) offset and, optionally, a description. It also deletes existing time zone names.

Operands:

ADD=*name*

This operand specifies a time zone name to be added. The name must be from 1 to 8 alphanumeric or national characters. The names USER and SYSTEM are not allowed.

OFFSET=*shhmm*

This operand specifies the GMT offset, expressed as a signed hours and minutes offset. The format is *shhmm*, where *s* is the sign, *hh* is the hours, and *mm* is the minutes offset (for example, +1030). Zones east of GMT use a positive offset. Zones west of GMT use a negative offset.

DESC=*description*

This optional operand allows a description of up to 40 characters to be entered. This description can be displayed using the SHOW TIMEZONE command.

DELETE=*name*

This operand specifies a time zone name to be deleted. No other operands are allowed with the DELETE operand.

Examples:

```
TIMEZONE ADD=AUSESTDT OFFSET=+1000 DESC=Australian Eastern Std Time

TIMEZONE DEL=AUSESTDT
```

Notes: A user's time zone is maintained through UAMS, and can be specified by users through the Password/Details Change function on the UAMS : Primary Menu. These UAMS fields are validated by the $UA table TZNAME and all valid time zone names must be added to this table. See the chapter *Maintaining Tables* in the *MODS Programming and Administration Guide* for details of how to add time zones names to the TZNAME table.

See Also: The SHOW TIMEZONE command description.

TITLE

Function: Sets the title to be displayed on the top line of the OCS window.

TITLE (4)	[*text of title* | DBCS=*dbcs-character-string*]

Use: The title that is displayed on the top line of the OCS screen can be set using the TITLE command.

The default title is Operator Console Services.

Operands: ***text of title***

Text of title exactly as it is required to appear. The title is centered and a leading and trailing blank added. The text should not be entered in quotes. The maximum length of the title is 70 characters.

If no title is required, then enter the TITLE command without any text.

DBCS=*dbcs-character-string*

The double-byte character set text required as the OCS title. This is the title that is displayed on OCS windows on display terminals enabled for DBCS operation (for example, KANJI).

The text is centered. Maximum length is 35 (double-byte) characters.

To erase a previously set title, enter the TITLE command without text.

Examples:

```
TITLE -- CHICAGO --
TITLE DBCS=........
```

Notes: The TITLE command is normally included in the NMINIT procedure, which is executed automatically during system initialization.

One each of the different titles can be set. The DBCS title string is sent automatically to OCS windows on terminals that are operating with DBCS enabled.

If no DBCS title is set, then DBCS-enabled terminals use the standard (non-DBCS) title.

TNCMD

Function: Sends a Telnet command on a connection.

```
TNCMD        { ? | id }
             CMD={ IP | AO | AYT | BRK }
```

Use: The TNCMD command is used to send commands defined by the Telnet protocol to the remote host.

Operands:

?

Displays help in the form of the TNCMD command syntax.

id

An identifier that is associated with the Telnet connection (see the TELNET command for more information).
(*Mandatory*)

CMD={ IP | AO | AYT | BRK }

Sends one of the following commands to a remote host:

- IP - Telnet Interrupt Process
- AO - Abort Output
- AYT - Are You There
- BRK - Break

Examples:

```
TNCMD router1 CMD=AYT
```

See Also: The TELNET and TNSEND commands.

Function: Manages TELNET ports.

```
TNCTL     ADDPORT=n | DEFPORT=n | REPPORT=n
          NVT={ NO | YES }
          TN3270E={ NO | YES }
          ANYNAME={ NO | YES }
          EASINET={ NO | YES }
          TEOPTS={ opt,opt,... }
          TEDSS={ rows,cols,... }
          COMMENT={ comment }

TNCTL     DELPORT=n
(2)
```

Use: Use TNCTL ADDPORT or TNCTL DEFPORT to define a Telnet port. Use TNCTL DELPORT to delete a Telnet port. Use TNCTL REPPORT to replace a Telnet port.

See the following sections for detailed descriptions of TNCTL ADDPORT, TNCTL DEFPORT, TNCTL DELPORT, and TNCTL REPPORT.

TNCTL ADDPORT
TNCTL DEFPORT
TNCTL REPPORT

Function: Manages TELNET ports.

```
TNCTL    ADDPORT=n | DEFPORT=n | REPPORT=n
         NVT={ NO | YES }
         TN3270E={ NO | YES }
         ANYNAME={ NO | YES }
         EASINET={ NO | YES }
         TEOPTS={ opt,opt,... }
         TEDSS={ rows,cols,... }
         COMMENT={ comment }
(2)
```

Use: Use TNCTL ADDPORT or TNCTL DEFPORT to define a Telnet port. Use TNCTL REPPORT to replace a Telnet port.

Note

Using TNCTL REPPORT replaces the entire port definition. Unspecified parameters are set to their defaults, not to their previous values.

Operands:

ADDPORT=*n* | DEFPORT=*n* | REPPORT=*n*

Identifies the port to be added. It can be either of the following:

- A number from 1 to 65535
- SHARED, meaning the current shared port

Note

For REPPORT=*n*, you cannot specify the number of the shared port (which could change anyway).

If the port is not currently defined, then TNCTL REPPORT acts as TNCTL ADDPORT.

NVT={ NO | YES }

Controls whether NVT (line-mode LU1) sessions will be accepted from this port. The default is NO.

TN3270E={ NO | YES }

Controls whether TN3270E sessions will be accepted from this port. The default is YES.

This allows you to define ports for clients that do not work properly with TN3270E, whilst defining other ports that allow TN3270E for those clients that do.

ANYNAME={ NO | YES }

Controls whether any arbitrary terminal name (LU name) is accepted if the client asks for a specific terminal name.

If this operand is set to NO (the default), then the name must be in the format set by the TNSPREFX SYSPARM (4-character prefix and 4-digit numeric suffix), and not in use. If this operand is set to YES, then any valid name (except one that would be in the generic or specific pool) that is not currently in use is accepted.

EASINET={ NO | YES }

Controls whether the session will be connected immediately to the MS domain, with the MS logon screen presented (NO, the default), or whether EASINET will be driven.

Note

EASINET can log on to the current MS domain only; any attempt to pass the terminal to another application fails. The advantage of using EASINET here is that it allows you to customize your logon screen.

TEOPTS={ *opt,opt,…* }

Enables or disables specific parts of the TN3270E protocol. If a specific client does not work properly, then disabling part of the protocol may help. Each option has an enabled setting and a disabled setting.

BIND | NOBIND

Whether a bind image is sent to the client.

RESP | NORESP

Whether responses are used.

SYSREQ | NOSYSREQ

Whether the SYSREQ key is supported.

TERM | NOTERM

Whether terminal sessions are supported.

PRINT | NOPRINT

Whether printer sessions are supported.

EOJ | NOEOJ

Whether the EOJ option is used (printer sessions only).

TNCTL ADDPORT
TNCTL DEFPORT
TNCTL REPPORT

TEDSS=(*rows,cols*)

Sets an alternate screen size, for TN3270E terminals only, if the client connects with the terminal-type name of IBM-DYNAMIC. The screen size is returned to the terminal in the BIND-IMAGE (unless NOBIND is set). This allows screen sizes other than the standard 24x80 (model 2) to be used. The primary screen size is set to 24x80.

Note

If a user connected using this option selects MAI, and attempts to connect to an application, then the LOGMODE used on the MAI definition is important. You may need to set up a LOGMODE entry that allows dynamic alternate screen size, where the application queries the terminal.

COMMENT=*comment*

Specifies a comment to be displayed on the SHOW TNCTL command output. This allows documentation of the port definition.

This operand must be the last on the command.

Examples:

```
TNCTL ADDPORT=9256 NVT=YES
TNCTL DEFPORT=SHARED
```

See Also: The TNCTL DELPORT and SHOW TNCTL commands.

TNCTL DELPORT

Function: Deletes a TELNET port definition.

You can delete a TNCTL port definition at any time, including while TCP/IP is active. Existing TELNET sessions started from that port are not affected; however, no new ones can be started. TNCTL DELPORT=SHARED stops further TELNET sessions starting from the shared port.

```
TNCTL     DELPORT=n
(2)
```

Use: Use TNCTL DELPORT to delete a Telnet port.

Operands: **DELPORT=*n***

Identifies the port to be deleted. It can be either of the following:

- A number from 1 to 65535
- SHARED, meaning the current shared port `TNCTL`

Examples: `TNCTL DELPORT=9256`

See Also: The TNCTL and SHOW TNCTL commands.

TNDISC

Function: Disconnects a Telnet connection.

TNDISC	{ ? | *id* }

Use: The TNDISC command is used to end a connection immediately.

Operands:

?

Displays help in the form of the TNDISC command syntax.

id

An identifier that is associated with the Telnet connection (see the TELNET command for more information).
(*Mandatory*)

Examples:

```
TNDISC router1
```

See Also: The TELNET command.

TNSEND

Function: Sends data on a Telnet connection.

TNSEND	{ ? | *id* } [DATA=] *message*

Use: The TNSEND command is used to communicate with a remote host.

Operands:

?

Displays help in the form of the TNSEND command syntax.

id

An identifier that is associated with the Telnet connection (see the TELNET command for more information).
(*Mandatory*)

[DATA=] *message*

The data to send on the connection. The DATA keyword can be omitted.

Examples:

```
TNSEND router1 DATA=LOGON USER01
TNSEND router1 ¢C
```

Notes: You can use the CTRL character as specified on the TELNET command to simulate the Ctrl key and send special characters on a connection.

See Also: The TELNET command.

TRACE

Function: Initiates a VTAM network trace.

TRACE (1)	{ *name* | *operand,operand,..operand* },TYPE=*type*

Use: This command builds and issues a VTAM MODIFY command in the format:

`F NET,ID=`*`name`*`,TRACE,TYPE=`*`type`*

or

`F NET,TRACE,`*`operand,operand`*`,TYPE=VTAM`

This command can be used to initiate a VTAM network trace.

Operands:

name
The VTAM name of the resource to have the trace initiated.

operand
The VTAM internal trace operand(s) required.

TYPE=*type*
The type of VTAM trace to be initiated.

Examples:

```
TRACE LU01,TYPE=BUF
TRACE LINE5,TYPE=LINE
TRACE OPTION=(PIU,API),TYPE=VTAM
```

Notes: Alternatives to this command would be:

```
F NET,ID=LU01,TRACE,TYPE=BUF
F NET,ID=LINE5,TRACE,TYPE=LINE
F NET,TRACE,TYPE=VTAM,OPTION=(API,PIU)
```

See Also: The F and NOTRACE command descriptions.

TRACEROUTE

Function: Traces a route taken by TCP/IP packets to a remote host.

```
TRACEROUTE  { ? | host_name | ip_address}
            [ COUNT={ 3 | count } ]
            [ SIZE={ 64 | packetsize } ]
            [ WAIT={ 5 | seconds } ]
            [ HOPS={ 5 | number } ]
```

Use: The TRACEROUTE command is used to obtain a hop-by-hop record of the route taken by a packet through a network.

Operands:

?

Displays help in the form of the TRACEROUTE command syntax.

host_name* | *ip_address

The name or IP address of the TCP/IP host that is the target of the trace route request. (*Mandatory*)

COUNT={ 3 | *count* }

The number of requests to send to each hop along the route to the TCP/IP host. If no count value is specified, then the default value of 3 is assumed. If a value is supplied, then it must be in the range 1 to 10.

SIZE={ 64 | *packetsize* }

The size in bytes of each probe message. If no packet size value is specified, then the default value of 64 is assumed. If a value is supplied, then it must be in the range 40 to 2048.

WAIT={ 5 | *seconds* }

The number of seconds to wait for a response to each probe request. If no wait value is specified, then the default value of 5 seconds is assumed. If a value is supplied, then it must be in the range 1 to 999.

HOPS={ 5 | *number* }

The number of hops to traverse on the search for a route to the remote TCP/IP host. If no hop limit is specified, then the default value of 5 hops is assumed. If a value is supplied, then it must be in the range 1 to 255.

TRACEROUTE

Examples:

```
TRACEROUTE mercury
TRACEROUTE 192.168.2.66 COUNT=1 HOPS=10
```

Notes:

The TRACEROUTE command can be used in the diagnosis of connectivity and performance related problems on a TCP/IP network. It can be used to find breaks in the route from an IBM TCP/IP host or a TCPaccess host to a remote host.

The TRACEROUTE command gathers all of its results prior to displaying them. Therefore, if routers or remote hosts do not respond to the requests, it might take a long time for the command to complete. This is because the timeouts are applied to each message at each hop; that is, *hopcount* * *seconds* * *count* = worst case time taken to complete the trace route.

TRANSMIT

Function: To request a dataset transmission.

TRANSMIT (0)	*defname* [*defname defname* ... *defname*]

Use: Issues request for execution of one or more Transmission Definitions.

Operands: ***defname* [*defname defname* ... *defname*]**

The names of one or more Transmission Definitions, containing details of the transmission activity that is required.

Examples:

```
TRANSMIT SPOOL17
TRANSMIT FILE1 FILE2 FILE3
```

Notes: XMIT can be used instead of TRANSMIT.

Authority checking is performed to ensure that the issuer of the command is authorized to execute the selected transmission request(s).

Multiple requests of the same class issued on one TRANSMIT command are processed in the order specified.

UDBCTL

Function: Controls the status of User DataBases (UDBs).

```
UDBCTL        { OPEN=ddname
                  [ BUFND=bufnd ]
                  [ BUFNI=bufni ]
                  [ CTLDD ]
                  [ DBCS={ IBM | FUJITSU } ] |
                  [ EXIT={ YES | NO } ]
                  [ ID={ fileid | * } ]
                  [ INPUT | [ RESET ] [ SIS ] [ DEFER ] ]
                  [ LSR | LSR=n ]
                  [ RLS ]
                  [ STRNO=strno ]
                ddname=fileid |
                RESET=ddname |
                STOP=fileid |
                CLOSE=ddname }
(3/4)
```

Use: The UDBCTL command lets you perform a range of online status changes to User DataBases (UDBs).

UDBCTL can be used to assign a logical file ID to a UDB that will subsequently be used in an NCL &FILEID statement. It can also be used to logically stop, physically close and open, empty, or reset a UDB.

Operands: **OPEN=*ddname***

Opens the UDB with the specified DD name. UDBs can be opened only by using the UDBCTL OPEN=*ddname* command. The OPEN operand can also be used either to open a UDB that failed to open at system initialization time or to open a UDB that was previously closed by using the CLOSE operand.

Processing performed during open processing is the same as that performed during system initialization. If the dataset was not previously closed successfully, then it is verified (by using the VSAM IDCAMS utility) and, if necessary, the file is loaded prior to making it available for processing. If the open fails, then an appropriate message is displayed, giving the VSAM ACB error. This message is also displayed in subsequent displays using the SHOW UDB command. After the open is complete, a logical file ID must still be assigned to make the UDB available for processing. Thus, a successful open moves a UDB from CLSD status to STOP.

Those DD names which begin with UDB are opened automatically by the MS region during initialization (unless the UDBDEFER initialization parameter was specified) (OS/VS only). The file ID and various VSAM processing options can be specified on the UDBCTL OPEN command as detailed below.

BUFND=*bufnd*

(*bufnd* = 1 to 99)
Allows the default or JCL-defined VSAM BUFND operand to be overridden during dataset open processing. This provides the number of data buffers VSAM allocates for handling data control intervals.

If the LSR operand is also specified, this operand is ignored.

BUFNI=*bufni*

(*bufni* = 1 to 99)
Allows the default or JCL-defined VSAM BUFNI operand to be overridden during dataset open processing. This provides the number of index buffers VSAM that allocates for handling index control intervals.

If the LSR operand is also specified, this operand is ignored.

CTLDD

Specifies that VSAM management occurs at the DD name level rather than at the default dataset level.

DBCS={ IBM | FUJITSU }

If the UDB is to contain data that includes Double Byte Character Set (DBCS) control characters, this operand can be used to override the system default assumptions. By default, the MS domain assumes that UDBs containing DBCS strings contain control characters that match the system's DBCS mode (see the SYSPARMS DBCS operand). The DBCS operand on UDBCTL lets you override the system default and create UDBs with specifically IBM or Fujitsu DBCS control character strings. There is no default for this operand: the default is determined by the setting of the SYSPARMS DBCS operand or (if no SYSPARMS command has been issued) by the operating system type.

EXIT={ <u>YES</u> | NO }

This operand can be used to specify whether NCLEX01 file security is to be called for this file. The default, EXIT=YES, means that the exit is called. Specifying EXIT=NO prevents calls to the exit for this file, and all accesses are allowed. This applies to the NCL &FILE verbs and also to the &NDB verbs.

ID=*fileid* | *

The logical name by which this UDB is to be referenced in NCL procedures. This name must be from 1 to 8 characters in length, start with an alpha or national character ,and consist of either numeric, alpha, or national characters. If an incorrect file ID is assigned, then the STOP operand must be used before a new name can be assigned. Specifying ID=* sets the file ID to the DD name specified by the OPEN operand. If this operand is omitted, then the file ID can be assigned by a subsequent UDBCTL *ddname=fileid* command.

INPUT

Specifies that the dataset is being opened for read-only access. This can lead to some VSAM efficiencies.

Note

The INPUT operand is mutually exclusive with the RESET, SIS, and DEFER operands.

RESET

(Default command authority=4)
Provides a means to empty a dataset which has been defined with the REUSE option (more details under the RESET=*ddname* operand description).

SIS

Means that this dataset is to be processed using VSAM's Sequential Insert Strategy, which provides improved VSAM management when dataset additions are always at the logical end of file or in groups of ascending keys.

DEFER

Indicates that VSAM update processing is using deferred I/O, which results in far fewer I/O requests. The advantages of such processing should be weighed against the possibility of losing data if there is a task abend or system failure.

LSR | LSR=*n*

Specifies that this dataset is to use VSAM Local Shared Resources for its buffer management. This provides certain efficiencies in operation.

Specifying LSR is equivalent to specifying LSR=0. A specific LSR pool can be indicated using LSR=*n*. If the number specified is higher than the highest LSR pool supported, then LSR=0 is assumed.

This operand is ignored if the RLS operand is also specified and RLS is supported by the operating system environment.

RLS

Specifies that this dataset is to use VSAM Record Level Sharing for its buffer management. This operand is supported only if your operating system is OS/390 V1 or later, with DFSMS V1.3 or later, and with a parallel sysplex environment. If RLS is not supported, then this operand is ignored.

STRNO=*strno*

(*strno* = 1 to 99)
Allows the default or JCL defined VSAM STRNO operand to be overridden during dataset open processing.

ddname=fileid

This allows a UDB to be assigned to a file ID for use with NCL procedures. Use this form of the UDBCTL command to assign the file ID if the UDBCTL OPEN command did not specify the ID operand, or if the UDBCTL STOP=*fileid* has been issued for this dataset.

RESET=*ddname*

The RESET operand requires a default command authority of 4 (all other operands require a default authority of 3). This is required to avoid inadvertent use of this facility. This operand applies only to UDB files automatically initialized by system startup. The RESET operand is accepted only if the UDB is already in CLSD (closed) status. The RESET operand resets or empties the entire UDB. In VSAM terms, it sets the high-order RBA to 0 and in doing so erases any data currently contained in the UDB.

Note

The RESET operand can be used only on certain UDB types. The UDB must have been defined with the REUSE attribute when it was originally created. In addition, VSAM does not permit a UDB that is defined with an alternate index to be reset.

If a UDB has been emptied by using the RESET operand, then it must be opened by using the OPEN operand if further processing is intended. Both operations can be achieved by specifying the RESET option on the UDBCTL OPEN command.

STOP=*fileid*

This operand can be used to logically STOP a UDB. This is rejected if the UDB has active users that are currently referencing it. The STOP function deletes any assigned file ID and makes the UDB unavailable for further processing until another file ID is assigned.

CLOSE=*ddname*

The CLOSE operand physically closes the VSAM ACB. It also implies a logical STOP of the UDB and hence is not performed if there are active users of the UDB. The close processing deletes any assigned file ID. If the UDB is subsequently reopened, then another file ID has to be assigned. The CLOSE function is normally used to make a UDB available for updating by another system.

Examples:

```
UDBCTL OPEN=USERFILE ID=* LSR DEFER RESET
UDBCTL CLOSE=BATCH1
UDBCTL STOP=BATCH1
```

Notes:

When opening a UDB, an MS domain performs an initial load of the UDB if it is empty. This is necessary to meet VSAM requirements for update processing. If this is done, then a single record, with a key of all X’00’, is loaded, and then deleted. If this is not required, then an external utility can be used to load the UDB prior to making it available for online processing. The options that apply to dataset open relate to the way the dataset is to be processed after it is opened. Hence, to alter any such options, the dataset must first be closed, then reopened specifying the desired options. For full details on all the supported VSAM processing options described above, see your *VSAM Programming Reference* manual.

See Also:

The ALLOCATE (z/OS, OS/390, MSP, and VOS3), SHOW UDB, SHOW VSAM, and SHOW UDBUSER command descriptions.

UNALLOC

Function: Unallocates a closed dataset currently allocated to the MS region.

UNALLOC (0/3)	DD=*ddname*

Use: The MS region supports the dynamic unallocation of datasets that were allocated through either the MS JCL or the ALLOCATE command, and that are now closed.

Operands: **DD=*ddname***

The DD operand is used to identify the dataset to be unallocated. (Use the SHOW ALLOC command to determine a dataset's DD name if it is unknown.)

Examples:

```
UNALLOC DD=SYS00002
UNALLOC DD=MYDD1
```

Notes: Before a dataset can be unallocated, it must be closed. For UDBs, this means issuing a UDBCTL CLOSE=*ddname* command.

The SHOW ALLOC command can be used to display all datasets allocated through the ALLOC command, and the DD names which are associated with each.

Datasets that have been concatenated using the CONCAT operand (z/OS and OS/390 only) can only be referenced using the prime DD name by which the group of datasets is known. If the prime DD name is subsequently freed using the UNALLOC command the entire concatenated group is no longer associated or recognized as a group and each DD name can be independently referenced. A separate UNALLOC command is required for each DD name to be unallocated.

To change the order of a concatenated group of datasets, you must unallocate them and then reallocate them in the desired sequence.

See Also: The SHOW ALLOC, ALLOCATE (z/OS, OS/390, MSP, and VOS3), and UDBCTL command descriptions.

Function: Issues a VTAM VARY command.

V (1)	{ *name,action* | *keyword* | NET,ID=*name,action* }

Use: This command has three basic formats:

- The first format allows entry of a VTAM network node name and VARY action required.

  An example of this is: `V NCP001,ACT`

  which is translated into a command with the format:
  `V NET,ID=NCP001,ACT`

- The second format supports the entry of reserved keywords that are known to VTAM.

  An example of this is: `V TERM,SLU=TERM1`

  which is translated into a command with the format:
  `V NET,TERM,SLU=TERM1`

  The supported keywords are listed below.

- The third format allows entry of normal fully expanded VTAM commands in an unmodified and untranslated format. An example of this is: `V NET,ID=NCP001,INACT,F`

Operands:

name,action
The VTAM name for the resource to be varied and the type of vary action required.

keyword
Under ACF/VTAM, *keyword* is one of the following:

ACQ	ANS	DIAL	DRDS
HANGUP	INOP	LOGON	NOLOGON
PATH	REL	TERM	

Under ECS/VTAM, *keyword* is one of the following:

ACQ	PATH	POWON	REL

Under VTAM-G, *keyword* is one of the following:

ACQ	ANS	DRDS	INOP
LOGON	PATH	REL	ROUTE
TERM			

A full explanation of these keywords can be found in the *VTAM Operation* manual.

NET,ID=*name,action*
Standard format VTAM command, including the name of the resource to be varied and the type of vary action required.

Examples:

```
V NCP001,ACT
V NODE1,INACT,I
V TERM,SLU=TERM1
```

Notes: Standard format VTAM command usage is supported, using the third format described above.

Using the ACT, INACT, and FORCE commands should be considered as an easier alternative to the V command.

See Also: The ABENDCMD, INACT, and FORCE command descriptions.

Function: Exits from OCS screen mode.

```
X
(0)
```

Use: The X command is used to exit from OCS. By default, the X command is associated with function keys F3 and F15.

The X command is always associated with F3 and F15 outside the OCS Mode.

Notes: When exiting from OCS, current PROFILE values are reset. Any function key values are reset to the global values and any paused NCL procedure are flushed.

XLATE

Function: Performs name translation testing through the Alias Name Translation Facility of NEWS.

```
XLATE          NAME=name
               NET=orignet
               TARGNET=targnet
             [ REAL ]
             [ TYPE={ LU | COS | MODE } ]
(1)
```

Use: This command can be used to determine the translated name that the Alias Name Translation Facility would return to VTAM when requested to perform translation for the names provided as input.

Operands:

NAME=*name*
The required name (1 to 8 characters) of the type specified by the TYPE operand for which translation is requested. When the REAL operand is also specified, this name is assumed to be a real name, otherwise it is assumed to be an *alias* name.

NET=*orignet*
The required network name (1 to 8 characters) of the originating network in which the *name* provided can be used.

TARGNET=*targnet*
The network name (1 to 8 characters) of the target network for which the supplied name is to be translated. Unless the REAL operand is omitted and the input name is TYPE=LU, this operand is required.

REAL
Indicates that the name supplied refers to a real name and that the translated name for the target network will be an alias name. If this operand is omitted, then the name supplied is assumed to be an alias name and the translated name for the target network is a real name.

TYPE= { LU | MODE | COS }
Indicates what type of input name is supplied. Omitting this operand results in the default type LU being assumed.

Examples:

```
XLATE NAME=TSO NET=XNET TARGNET=CDNET REAL
XLATE NAME=IMSA NET=NETWORKF TARGNET=NETWORKA
XLATE NAME=LUM4 NET=$$NET TARGNET=IMSNET TYPE=MODE
```

See Also: The DEFALIAS and SHOW DEFALIAS command descriptions.

The chapters about NEWS in the *Unicenter NetMaster Network Management for SNA Implementation and Administration Guide*.

Function: Removes all XNF connectivity.

XNF (3)	STOP

Use: The XNF command terminates all XNF connections between this MS region and other nodes in the XNF network. This includes terminal connections, connection to other MS regions, and connections to NETM. When all connections are terminated, the MS region's ULE is released.

This command is intended for use by the system operator. If XNF needs to be shut down for maintenance, then the XNF command allows the MS region to remain active while XNF is stopped. Any MS process that does not require XNF services continues to execute normally.

After XNF STOP is executed, a WTOR message is written to the console. The system operator must reply to this message to reactivate MS connections to XNF.

Operands: **STOP**

Terminates all XNF connections.

Notes: This command applies only to MS regions using Hitachi's XNF product.

XNFTRACE

Function: Initiates or terminates tracing of a specified XNF connection.

XNFTRACE { START=*xxxxxx* | STOP={ *xxxxxx* | ALL } } (4)

Use: This command can be used to request that tracing of the node specified on the START= or STOP= operand is to be commenced or halted. Multiple connections can be traced concurrently, each connection being identified by a separate START= operand.

Operands:

START=*xxxxxx*
Starts a trace of node *xxxxxx*. If there is currently no connection to the node, then the request is queued, to start as soon as a connection is established.

STOP={ *xxxxxx* | ALL }
Stops the trace of node *xxxxxx*, or all nodes when the ALL option is specified.

Examples:

```
XNFTRACE START=TERM1
XNFTRACE STOP=TERM2
```

Notes: This command applies only to MS regions using Hitachi's XNF product.

The trace output is written to the MS activity log.

See Also: SHOW XNFTRACE command description.

$SPLC15U ACTION=SNMPGET

Function: Retrieves and displays SNMP objects.

```
$SPLC15U    ACTION=SNMPGET
            MANAGER=manager_name
            DEVICE={ device_name | device_address }
          [ COMMNAME={ public | community_name } ]
            OBJECT=object_ID
```

Use: The SNMPGET command is used to browse a group of SNMP objects.

Operands:

ACTION=SNMPGET
The action to be performed.
(*Mandatory*)

MANAGER=*manager_name*
The name of the SNMP manager to be used, as defined in the NetMaster for TCP/IP configuration file.
(*Mandatory*)

DEVICE={*device_name*|*device_address*}
The name or address of the SNMP device that is to receive the command.
(*Mandatory*)

COMMNAME={public|*community_name*}
If no community name is specified, the default value of *public* is assumed.
(*Optional*)

OBJECT=*object_ID*
The identifier of the object whose details are to be retrieved.
(*Mandatory*)

$SPLC15U ACTION=SNMPSET

Function: Changes the values of SNMP objects.

```
$SPLC15U    ACTION=SNMPSET
            MANAGER=manager_name
            DEVICE={ device_name | device_address }
          [ COMMNAME={ public | community_name } ]
            OBJECT=object_ID
            VALUE=new_object_value
```

Use: The SNMPSET command is used to change the value of individual objects.

Operands:

ACTION=SNMPSET
The action to be performed.
(*Mandatory*)

MANAGER=*manager_name*
The name of the SNMP manager to be used, as defined in the NetMaster for TCP/IP configuration file.
(*Mandatory*)

DEVICE={*device_name*|*device_address*}
The name or address of the SNMP device that is to receive the command.
(*Mandatory*)

COMMNAME={public|*community_name*}
If no community name is specified, the default value of *public* is assumed.

OBJECT=*object_ID*
The identifier of the object whose details are to be retrieved.
(*Mandatory*)

VALUE=*new_object_value*
The new value to be used to update the current value of the object identified by the OBJECT= operand.
(*Mandatory*)

Note

If a value contains blanks, then it must be enclosed in quotes. For example, to change the value of the object: *system.syscontact.0* (the contact name for a device), type:

OBJECT=SYSTEM.SYSCONTACT.0 VALUE=‘Joe Bloggs’

If the value is to be null, it *must* be specified as follows:

VALUE=‘ ’