Airline Control System
Version 2.4.1

Operation and Maintenance

SH19-6955-23

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page xiv.

This edition applies to Release 4, Modification Level 1, of Airline Control System Version 2, Program Number 5695-068, and to all subsequent releases and modifications until otherwise indicated in new editions. Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below. A form for readers' comments appears at the back of this publication. If the form has been removed, address your comments to:

ALCS Development
2455 South Road
P923
Poughkeepsie NY 12601-5400
USA

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

Contents

Figures

Tables

Notices

References in this publication to IBM® products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

```
The Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
USA
```

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

```
IBM Corporation
Department 830A
522 South Road
Mail Drop P131
Poughkeepsie, NY 12601-5400
USA
```

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Programming interface information

This Operation and Maintenance manual is intended to help operators and system programmers to operate Airline Control System Version, Program Number 5695-068. This Operation and Maintenance manual documents General-Use Programming Interface and Associated Guidance Information provided by Airline Control System Version 2 Program Number 5695-068.

General-Use programming interfaces allow the customer to write programs that obtain the services of Airline Control System Version 2.

However, this Operation and Maintenance manual also documents Product-Sensitive Programming and Associated Guidance Information.

Product-Sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-Sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-Sensitive Programming Interfaces and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or by the following marking:

Product-sensitive Programming Interface

Product-Sensitive Programming Interface and Associated Guidance Information...

End of Product-sensitive Programming Interface

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

About this book

This book describes how to operate and maintain Release 4.1 of Airline Control System (ALCS) Version 2, an IBM licensed program.

ALCS is one of a family of IBM programs designed to satisfy the needs of airlines and other industries with similar requirements for high-volume and high-availability transaction processing.

The product, which is also known as TPF/MVS, provides the Transaction Processing Facility (TPF) application programming interface (API) for z/OS® environments. It supersedes ALCS/Multiple Virtual Storage/Extended Architecture (ALCS/MVS/XA), known as ALCS Version 1.

Throughout this book:

- Airline Control System Version 2 is abbreviated to ALCS unless the context makes it necessary to distinguish between ALCS Version 2 Release 4.1, and the predecessor products ALCS Version 2 Release 3, ALCS Version 2 Release 2, ALCS Version 2 Release 1, ALCS/MVS/XA and ALCS/VSE.
- Airlines Line Control Interconnection (ALCI) includes the function of network extension facility (NEF).
- Advanced Communications Function for the Virtual Telecommunication Method is abbreviated to VTAM®.
- TPF refers to all versions of Transaction Processing Facility and its predecessor, Airlines Control Program (ACP).
- MVS™ refers to z/OS.

Who should read this book

This book is intended mainly for ALCS operators and system programmers but will also be useful to application programmers who need to run their applications on a test system.

Chapter 7, “Using the ALCS trace facility,” on page 306 describes the commands and subcommands of the ALCS trace facility, which is used by application programmers.

You should read *ALCS Concepts and Facilities* before you read this book or attempt to run an ALCS system.

How this book is organized

The book is organized as follows:

Chapter 1, “Operating the ALCS system,” on page 1
Provides information required by the operator and the system programmer to run typical ALCS activities.

Chapter 5, “Recoup,” on page 52
Gives an overview of the ALCS long-term pool space recovery utility Recoup.

Chapter 6, “ALCS command reference,” on page 55
Describes the purpose and use of each ALCS operator command. The commands are presented in alphabetical order.

Chapter 7, “Using the ALCS trace facility,” on page 306
Describes the ALCS tracing facilities used by application programmers.

Chapter 8, “Maintaining ALCS,” on page 349
Describes how to run regular maintenance procedures on an ALCS system and how to tune the ALCS system to run most effectively. ALCS tools are described in detail and references are made to MVS tools which may be useful.

Appendix A, “Command synonyms,” on page 401
Lists TPF, ALCS/VSE, and ALCS/MVS/XA commands that have ALCS equivalents.

Appendix B, "Acronyms and abbreviations," on page 404
Lists acronyms and abbreviations used throughout the ALCS library. Not every term necessarily occurs in this book.

The book also contains a glossary of terms, a bibliography, and an index.

Using ISPF panels

ALCS provides Interactive System Productivity Facility (ISPF) panels to simplify installation and maintenance tasks. Figure 1 on page xvii shows the starting screen for these tasks.

```
  File  Options  Help
--------------------------------------------------------------------------------
                            ALCS primary menu
Command ===> ____________________________________________________________

Choose one of the following actions, then press ENTER


Actions:
__  1.  Installation
    2.  Generation and database creation
    3.  Maintenance
    4.  Operations
    5.  Application development

ALCS system . . SMPE      IBM-supplied SMP/E system definitions
```

Figure 1. ISPF panel: ALCS Primary Menu

Generating sample job streams

You can use the ALCS ISPF panels to produce a copy of a job stream (to integrate into your site procedures for example).

Each panel which runs a job stream contains an action bar with a **File** pulldown menu. This pulldown menu contains an option to **Edit the job stream**. This option builds the job (with the parameters which you specify before invoking the panel) and places you in ISPF edit mode.

Use the ISPF-edit **`Create`** command to save a copy of the built JCL in your own private library.

Online help for the ALCS ISPF panels

Press F1 to request help for a task. The type of help you will get depends on the cursor position:

Field-level help
Use the tab key to place the cursor on the required field and press the F1 function key. This help is also known as **contextual help**.

Panel-level help
Place the cursor away from any field and press the F1 function key. This help is also known as **general help** and **extended help**.

Further help on ISPF panels in the ALCS library

Table 1 on page xvii shows where you can find more information about tasks listed in the main ALCS ISPF panel.

Table 1. Where to find more information about the primary ISPF panel

Task	
Installation	*ALCS Installation and Customization*

Table 1. Where to find more information about the primary ISPF panel (continued)	
Task	
Generation and Database creation	*ALCS Installation and Customization*
Maintenance	Chapter 8, “Maintaining ALCS,” on page 349
Operations	Chapter 1, “Operating the ALCS system,” on page 1
Application Development	*ALCS Application Programming Guide*

Chapter 1. Operating the ALCS system

This chapter describes the different system states of ALCS and explains the procedures for initiating and terminating ALCS.

It also describes how to control some typical ALCS activities, including:

- Controlling the VTAM, TCP/IP, and SLC communication networks
- Recovering from system failure and restarting
- Backing up and restoring the database
- Logging and restoring database updates
- Reconfiguring ALCS

ALCS Concepts and Facilities describes how ALCS relates to the hardware and software in a typical installation. See Figure 2 on page 2 for a summary of an ALCS installation.

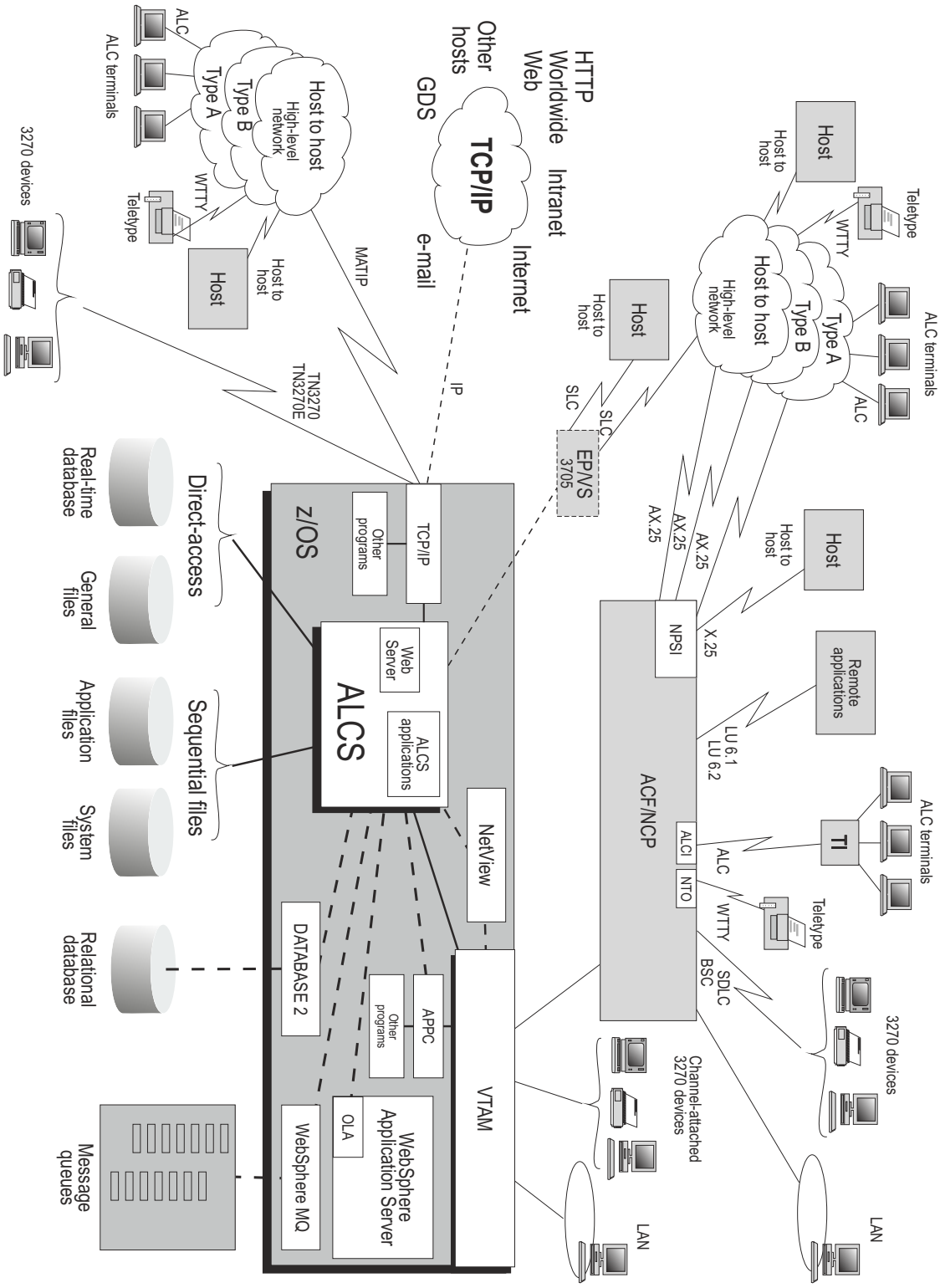


Figure 2. ALCS overview

Operational states

While ALCS is operational, it is in one of the following system states:

NORM
(Normal state) This state provides the highest (least restricted) level of ALCS service to users. Typically, all user applications are available but some application maintenance or control functions are not. Most (but not all) ALCS operator commands are available in NORM state.

MESW
(Message switching state) This state provides the next level of service below NORM state. Typically, a subset of user applications and some maintenance or control functions are available in this state. If the IPARS Message Switching application is installed, it will probably be available. Most ALCS operator commands are available in MESW state.

CRAS
(Computer room agent set) This state provides the next level of service below MESW state. Normally, user application functions are not available in CRAS state but application maintenance and control functions are.

IDLE
This normally provides the lowest (most restricted) level of user service. In IDLE state, ALCS typically rejects all input messages except operator commands. ALCS IDLE state is similar to the TPF 1052 and utility states.

RESTORE
This is the IDLE state with a special system flag set to force long term pool record dispense to take place from the pool dispensing array for restore structure (PDAR).

Note: The actual application functions that are available in each of the four system states are defined by the application design.

Non-operational states

ALCS is in one of two other states when it is non-operational:

STANDBY
ALCS is in this state when it is has been initialized but before it becomes operational. There are two forms of standby state:

- ALCS is not enabled for automatic system takeover and it is waiting for a reply to message DXC110R (see “ALCS actions on start up” on page 9). ALCS uses very few resources in this state.
- ALCS is enabled for automatic system takeover and it is waiting for a reply to message DXC116D (see “Message DXC116D from the standby ALCS jobs” on page 12). In this state, ALCS uses system memory and processor resources in order to check the status of the ALCS database every second.

HALT
The state ALCS is in when it has been terminated.

Using ISPF panels to operate ALCS

ALCS provides Interactive System Productivity Facility (ISPF) panels to simplify operations tasks. Figure 3 on page 4 shows the screen for these tasks.

```
                              ALCS operations
Command ===> _____________________________________________________________

Choose one of the following actions, then press ENTER


Actions:
__  1.  Run monitor
    2.  Run communication report file generator
    3.  Run diagnostic file processor
    4.  Run statistical report generator
    5.  Run cross-reference facility
    6.  Create DRIL data file
    7.  Run OCTM offline support program
    8.  Run CPU time report generator

ALCS system . : SMPE      IBM-supplied SMP/E system definitions
```

Figure 3. ISPF panel: ALCS Operations

Table 2 on page 4 shows where you can find more information in this manual about tasks in the ALCS Operations ISPF panel.

Table 2. Where to find more information about the operations ISPF panel

Task	
Run monitor	"Initiating ALCS" on page 5
Run communications report file generator	"Using ISPF panels to run the communication report file generator" on page 26
Run diagnostic file processor	"Running the ALCS diagnostic file processor" on page 26
Run statistical report generator	"Running the ALCS statistical report generator" on page 36
Run cross reference facility	"Running the ALCS cross reference facility" on page 41
Run OCTM offline support program	"Running the ALCS OCTM offline support program" on page 41
Run CPU time report generator	"Running the CPU time report generator" on page 41

Requirements before initiating ALCS

About this task

Before initiating ALCS, do the following:

Procedure

1. If NetView® is in use, ensure that it is started.
2. Ensure that JES is active.
3. Start VTAM. Ensure that:
   - The ALCS application is active. The ALCS application is a logical unit (LU) defined in a VTAM application major node.

- At least one CRAS display and at least one CRAS printer are available and not in use by any other VTAM application. In an ALCS test system, the RO CRAS can be allocated to a system test vehicle (STV) printer.

4. Ensure that all the necessary direct access storage devices (DASDs) that contain configuration data sets, database data sets and general files are online.
5. If any of the following are magnetic tape data sets, ensure that the appropriate tape volumes are mounted, or are ready to be mounted when requested by MVS:
   - ALCS diagnostic file.
   - ALCS update log file.
   - ALCS data collection file (if one was defined in the sequential file generation).
   - ALCS user file (if one was defined in the sequential file generation).
   - Any application real-time sequential files.

   Note: The data collection or user file output can be placed on the diagnostic file. Your system programmer decides this for your installation.
6. For ATA/IATA Synchronous Link Control (SLC) communication, ensure that the communication controller and the Emulation Program (EP) subchannels for SLC are available, with the appropriate EP loaded.
7. For Airlines Line Control (ALC) communication, ensure that the communication controller is online and available, and that the network control program (NCP) is loaded with the appropriate Airlines Line Control Interconnection (ALCI).
8. For X.25 communication, ensure that the communication controller is online and available, and that the appropriate NCP is loaded with the NCP packet switching interface (NPSI).
9. For LU 6.1 communication, ensure that the communication paths are online and available.
10. For LU 6.2 communication, ensure that the Advanced Program to Program Communications/MVS (APPC/MVS) is up and running.
11. For binary synchronous communication (BSC) or Synchronous Data Link Control (SDLC) communication, ensure that the communication controller is online and available, and that the NCP is loaded.
12. For Synchronous Data Link Control (SDLC) communication or TN3270 or TN3270E communication using a workstation terminal emulator, ensure that the workstation is available and that the terminal emulator program and the Online Message Facility is started if required.

    For TN3270 or TN3270E communication, also ensure that the Telnet server is started on z/OS.
13. For World Trade Teletypewriter (WTTY) communication, ensure that the communication controller is online and available, and that the appropriate NCP is loaded with the Network Terminal Option (NTO) program.
14. If any of your applications issue SQL requests, ensure that the local DB2® for z/OS subsystem and that the appropriate database is started and available.
15. If any of your applications issue MQI requests, ensure that the local WebSphere MQ for z/OS subsystem is started and that the appropriate queue manager and queues are available.
16. For Transmission Control Protocol / Internet Protocol (TCP/IP) communication (including connections defined in the ALCS communication generation, applications that use TCP/IP Sockets requests, E-mail, and the ALCS Web Server) ensure that the local TCP/IP subsystem is started.
17. If any of your ALCS applications use either the WebSphere® Application Server for z/OS optimized local adapters (OLA) callable services or the WAS Bridge, ensure that WebSphere Application Server for z/OS is started.
18. If any of your applications use ICSF, ensure that ICSF is started on z/OS.

Initiating ALCS

There are three ways to initiate ALCS:

1. Use the ISPF panels.
2. Submit an MVS job to run ALCS.
3. Use the MVS START command to initiate a cataloged procedure set up by your system programmer.

You can use the ISPF panels to assist in creating JCL. See “Generating sample job streams” on page xvii for an explanation.

Using ISPF panels to start ALCS

```
  File  Options  Help
------------------------------------------------------------------------------
                           ALCS primary menu
Command ===> ____________________________________________________________

Choose one of the following actions, then press ENTER


Actions:
__  1.  Installation
    2.  Generation and database creation
    3.  Maintenance
    4.  Operations
    5.  Application development

ALCS system . . SMPE      IBM-supplied SMP/E system definitions
```

Figure 4. ISPF panel: ALCS Primary Menu

Select option 4 to display the Operations panel which is shown in Figure 5 on page 6.

```
                             ALCS operations
Command ===> ____________________________________________________________

Choose one of the following actions, then press ENTER


Actions:
__  1.  Run monitor
    2.  Run communication report file generator
    3.  Run diagnostic file processor
    4.  Run statistical report generator
    5.  Run cross-reference facility
    6.  Create DRIL data file

ALCS system . . SMPE      IBM-supplied SMP/E system definitions
```

Figure 5. ISPF panel: ALCS Operations

Select option 1 to display the Run ALCS monitor panel which is shown in Figure 6 on page 7.

```
                            Run ALCS monitor
Command ===> ____________________________________________________________

Select a function from the File menu.

ALCS system . : SMPE      IBM-supplied SMP/E system definitions



 F1=Help   F2=Split  F3=Quit   F4=File   F7=Up     F8=Down   F9=Swap
```

Figure 6. ISPF panel: Run monitor

Table 3 on page 7 shows where you can find more information about running the ALCS monitor.

Table 3. Where to find more information about running the ALCS monitor

Install	Use	Test	Change
ALCS Installation and Customization	"ALCS actions on start up" on page 9	*ALCS Installation and Customization*	*ALCS Installation and Customization*

Job control statements to run ALCS

Figure 7 on page 9 shows an example of job control statements to run ALCS. The ALCS monitor uses the MVS dynamic allocation facility (SVC99) to allocate all database and general file data sets. The DASD configuration table generated by the ALCS DASD generation contains these data set names. Thus the MVS JCL used to run ALCS does not contain any DD statements for these data sets. The following paragraphs describe each job control statement.

EXEC Statement

Include an EXEC statement to execute the ALCS online monitor program. Include at least the following parameters:

PGM=DXCMON
Member name of the ALCS online monitor program.

PARM=({*tcbs*|TEST],[*st*],[*dt*],[*ct*],[*qt*],[*pt*],[*ut*],[*state*|X*state*])
Where:

tcbs
Number of CPU loop TCBs that ALCS ATTACHes to exploit multiprocessors. Do not specify a number larger than the number of processors. The format is one of:

nnn
Number of CPU loop TCBs. The minimum value 1, this is also the default. The maximum value is 32.

nnn-mmm
Use this format if you have enabled the ALCS dynamic TCB facility, by specifying DYNTCB=YES on the SCTGEN system generation macro. *nnn* is the number of active CPU loop TCBs when ALCS starts, and *mmm* is the maximum number of CPU loop TCBs. The minimum value for *nnn* or *mmm* is 1 and the maximum value is 32, where *nnn* must not be greater than *mmm*. Use the ZCTCB command to activate and deactivate CPU loop TCBs.

TEST
Run with a test database; that is, access the database in read-only mode, and write updated records to the test data set.

st

Member name of the system configuration table. This is the name specified on the MEMBER parameter of the SCTGEN macro. The default is DXCCDS. *ALCS Installation and Customization* describes the ALCS generation macros.

dt

Member name of the DASD configuration table. This is the name specified on the MEMBER parameter of the DBGEN macro. The default is DXCCDD. *ALCS Installation and Customization* describes the ALCS generation macros.

ct

Member name of the communication load list. The default is COMSLIST. *ALCS Installation and Customization* describes the ALCS generation macros.

qt

Member name of the sequential file configuration table. This is the name specified on the MEMBER parameter of the SEQGEN macro. The default is DXCCDQ. *ALCS Installation and Customization* describes the ALCS generation macros.

pt

Member name of the program configuration table. The program configuration table contains the application program load module list. The default is PROGLIST. *ALCS Installation and Customization* describes the ALCS generation macros.

ut

Parameter for user initialization. ALCS passes this as a parameter to the user initialization exit (see *ALCS Installation and Customization*). It can be any character string up to 8 characters. ALCS does not check or use this parameter.

state

Initial state for ALCS, one of:

- IDLE
- CRAS
- MESW
- NORM
- RESTORE

If you omit this parameter, ALCS stops in a standby state and issues message DXC110R. If you specify an operational system state, by including this parameter, ALCS goes straight to that state, without stopping in standby state and without issuing message DXC110R.

When you are starting an alternate ALCS system for automatic system takeover, include the takeover option X followed by the initial state. That is, one of:

- XIDLE
- XCRAS
- XMESW
- XNORM

JOBLIB or STEPLIB DD Statement

Include a JOBLIB or STEPLIB DD statement to identify the library or libraries that contain the ALCS online monitor load modules. Note that this library must be authorized program facility (APF) authorized.

DXCCLIB DD Statement

Include a DD statement to identify the library or libraries that contain the ALCS configuration table load modules. The DDNAME is DXCCLIB. Note that this library must be APF authorized.

DXCPLIB DD Statement

Include a DD statement to identify the library or libraries that contain the ALCS application program load modules. The DDNAME is DXCPLIB.

DXCHLIB DD Statement

This is an optional statement that identifies:

- The library that contains the LE run-time library routines, required for running HLL applications.
- The ALCS load library containing load modules DXC10OSM and DXCCSATP, and the Debug Tool load library, required for using the C/C++ remote debugger facility.

Note that these libraries must be APF authorized.

You may include these libraries in the MVS linklist or the JOBLIB/STEPLIB statement for the ALCS job. If you prefer not to, then you must include the DXCHLIB DD statement if you are running HLL applications or using the C/C++ remote debugger facility.

SYSABEND, SYSMDUMP, or SYSUDUMP DD Statement

Include one or more of these statements as required. MVS uses these data sets for ALCS termination dumps. For performance reasons, SYSMDUMP is the preferred option.

DXCTEST DD Statement

If you are running with the test database facility (PARM=(TEST,...)) then include this DD statement to identify the VSAM cluster which is the test data set. To get good performance, you may wish to override the default VSAM buffer allocation for the test data set in this DD statement.

```
//ALCS   EXEC PGM=DXCMON,REGION=6000K,
//            PARM=(2,DXCCDSW0,DXCCDDW0,DXCCMLW0,DXCCDQW0,DXCPGLW0,,IDLE)
//STEPLIB  DD DSNAME=DXC.V2R4M1.DXCLMD3,DISP=SHR
//*
//*           AUTHORIZED LIBRARY CONTAINING CONFIGURATION MODULES
//*
//DXCCLIB  DD DSNAME=DXC.V2R4M1.DXCLMD4,DISP=SHR
//*
//*           LIBRARIES CONTAINING ECB-CONTROLLED PROGRAM MODULES
//*
//DXCPLIB  DD DSNAME=DXC.V2R4M1.DXCLMD1,DISP=SHR
//         DD DSNAME=DXC.V2R4M1.DXCLMD2,DISP=SHR
//*
//*
//*SYSMDUMP DD DSN=DXC.V2R4M1.DUMP,DISP=(MOD,CATLG,CATLG),UNIT=SYSDA,
//*            SPACE=(CYL,(2,40),RLSE),DCB=(BLKSIZE=23476,BUFNO=5)
```

Figure 7. Example of JCL to run the ALCS online monitor

Note: If your VFA buffers are above the bar, then you need to specify a MEMLIMIT for ALCS in the PARMLIB SMFPRMxx member. However an alternative solution is to specify MEMLIMIT on either a JOB or an EXEC job control statement.

ALCS actions on start up

On initiation, ALCS loads various tables and sends the following message (number DXC110R) to the MVS operator:

```
Standby state -- Reply with required system state (IDLE, CRAS, MESW, NORM
RESTORE)
```

Note: A run-time parameter to transfer to the required system state might be included in the job stream or cataloged procedure. In this case you do not receive message DXC110R.

When you are ready to complete the initiation of ALCS, reply to message DXC110R by entering the name of the system state in which you want ALCS to operate. Enter one of the following:

```
N or NORM
M or MESW
C or CRAS
I or IDLE
R or RESTORE
```

Alternatively, reply CANCEL to cancel this ALCS job. If you ignore the message, ALCS remains in standby state.

When you have requested one of the four system states, ALCS establishes communication with VTAM and with all VTAM terminals defined to ALCS as being initially active. (The system programmer defines these in the ALCS generation.) ALCS also re-checks the availability of real-time database DASDs.

If it is operating in any of the three states above IDLE state, ALCS also establishes communication with the SLC lines defined as being active. (These are also defined by the system programmer in the ALCS generation.)

ALCS system takeover facility

To start an alternate ALCS for automatic system takeover, you specify a run-time parameter in the job stream or cataloged procedure including the takeover option X followed by the required system state. In this case you do not receive message DXC110R. Instead, on initiation, ALCS loads various tables and sends the following message (number DXC116D) to the MVS operator:

```
Can not obtain exclusive control for database --
ALCS takeover is now waiting -- Reply C to cancel takeover
```

You do not need to reply to this mesage. The alternate ALCS checks the status of the ALCS database every second. When the active ALCS terminates, the alternate ALCS automatically completes initiation and transfers to the required system state.

Alternatively, reply C to cancel this ALCS job.

Changing the current ALCS system state

ALCS provides commands to display (ZDSYS) and alter (ZASYS) the current ALCS system state. Altering the system state is also called ***cycling*** the system.

When you use ZASYS to alter the system state, ALCS makes some application functions available or unavailable, so ZASYS can take several seconds or minutes to complete, depending on the application.

Some error conditions, including application program errors, can prevent ZASYS from completing. ALCS provides parameters for ZASYS that allow you to reset the ALCS system state if this happens.

Terminating ALCS

ALCS is designed for continuous operation.

You can terminate ALCS in any system state. Enter:

```
ZASYS HALT
```

If ALCS is not in IDLE state, this is equivalent to entering ZASYS IDLE followed by ZASYS HALT.

Notes:

1. Both ZASYS IDLE and ZASYS HALT cause ALCS to terminate active entries.
2. You can use the MVS CANCEL command to cancel ALCS but this is likely to cause data corruption when ALCS is in a state higher than IDLE.

3. Because ALCS is designed for continuous 24-hour operation, it does not terminate normally. For abnormal program termination completion codes, see *ALCS Messages and Codes*.

CRAS terminals

When ALCS is started by the MVS operator it is initialized to a predetermined state (usually IDLE). In this state, ALCS is controlled by ALCS operators using computer room agents set (CRAS) terminals. See *ALCS Concepts and Facilities* for an explanation of the different types of CRAS terminals.

Automated operations

When ALCS is set up for automated operations, the Prime CRAS and RO CRAS are assigned to NetView resources.

You can input ALCS commands from the NetView operator ID that you designate as Prime CRAS. (In fact, you can also enter application input messages from this operator ID if you wish.)

You will probably want to route RO CRAS output messages from ALCS to the NetView log. You can also route network status information from ALCS to the NetView hardware monitor.

See *ALCS Concepts and Facilities* for a full description of using automated operations at your site.

Recovery and restart

This section describes the facilities that are available to you to recover from system failures and to restart ALCS.

Each installation normally sets up its own procedures for recovery and restart. Your installation may also use automated operations to simplify these procedures. Automated operations are described in *ALCS Concepts and Facilities*.

Starting alternate ALCS jobs

One way to run ALCS is to have a single ALCS job running under z/OS.

It is also possible to initiate two or more ALCS jobs. Complete the initiation of one of these (the prime job) into an operational system state and leave all the others (the alternate ALCS jobs) in standby state. You can later use one of the ALCS jobs in standby state to take over from the prime ALCS job if it fails.

If possible, put the alternate ALCS on a separate processor. This safeguards against a processor failure, and against a z/OS or VTAM failure. The separate processor must have access to the database and communication facilities.

It is only possible to put ALCS on a separate processor if MVS global resource serialization (GRS) is employed. See “MVS global resource serialization” on page 12.

You can optionally enable one of the alternate ALCS jobs for automatic system takeover, by specifying the takeover option on the system state run-time parameter (See “ALCS actions on start up” on page 9). The alternate ALCS job enabled for system takeover will check the status of the ALCS database every second. When the prime ALCS job terminates, the alternate job automatically commences state change to the required system state. If the alternate ALCS is on a separate processor, or another LPAR in the same sysplex, access to the database and communication network may need to be switched between processors before the alternate ALCS can take over. You may be able to automate this switching by using ALCS system takeover to drive NetView CLISTs or WTO exits. Please contact IBM for more information about this topic.

You can use one of the alternate ALCS jobs - which is not enabled for automatic system takeover - to take over from the prime ALCS job if it fails. However this requires manual intervention as the operator must reply to the DXC110R message.

Message DXC110R from the standby ALCS jobs

This applies to alternate ALCS jobs which are not enabled for automatic system takeover.

As described in "ALCS actions on start up" on page 9, each ALCS job performs some initialization actions then issues message DXC110R to the MVS operator:

```
Standby state -- Reply with required system state (IDLE, CRAS, MESW, or NORM)
```

The prime ALCS job has access to the required database on DASD when you reply to this message.

Only reply to the DXC110R message for the prime ALCS job.

Do not reply to the DXC110R messages issued by the alternate ALCS jobs. (You want these jobs to remain in standby state.)

While the alternate ALCS jobs are waiting for the reply to message DXC110R, MVS can swap them out. The alternate ALCS jobs use very few resources while they are swapped out.

Message DXC116D from the standby ALCS jobs

This applies to alternate ALCS jobs which are enabled for automatic system takeover.

As described in "ALCS actions on start up" on page 9, an alternate ALCS system enabled for automatic system takeover performs some initialization actions and then issues message DXC116D to the MVS operator:

```
Can not obtain exclusive control for database --
ALCS takeover is now waiting -- Reply C to cancel takeover
```

The alternate ALCS job enabled for system takeover (which is not swapped out) checks the status of the ALCS database every second. When the prime ALCS job terminates, the alternate job automatically commences state change to the required system state.

Alternatively the MVS operator may cancel the alternate ALCS job by replying `C` to message DXC116D.

MVS global resource serialization

When MVS global resource serialization (GRS) is available, ALCS uses it to safeguard against multiple ALCS jobs updating the same real-time database.

When you reply to the message DXC110R, ALCS uses GRS to check whether another ALCS is already updating the real-time database data sets. (That is, if there is already another ALCS that is using the same DASD configuration table name.) If so, ALCS issues message number DXC112R to the MVS operator:

```
Unable to obtain exclusive control for database - Reply U to proceed
```

If you receive this message, it is normally because you have replied to message DXC110R for a second ALCS job while the first ALCS job is running.

This message allows the following replies:

U
: ALCS ignores the fact that two jobs are accessing the same database and completes the initialization. For example, you may need to use this reply if the system is in a transient state during system failure.

N (or any other reply)
: ALCS does not complete initialization. It reissues message number DXC110R and remains in standby state.

Attention

In most circumstances, you must answer N to this message. If you do not, there is a risk of these two or more jobs running simultaneously and overwriting the database.

Alternatively, you can reply CANCEL to cancel this ALCS job.

VTAM application name serialization

After leaving standby state, ALCS establishes communication with VTAM. Each ALCS system is defined to VTAM as an application logical unit (LU). If this application LU is not available, or is already in use (for example by another ALCS job), ALCS sends the message number DXC200R to the MVS operator:

```
Open VTAM ACB failed, return code X'code' -- Reply U to retry or C to cancel
```

If you receive this message, it is normally because you have replied to message number DXC110R or DXC112R for a second ALCS job while a prime ALCS job is running.

This message allows the following replies:

U
ALCS tries to establish communication with VTAM again. If it succeeds, ALCS completes the initialization. Otherwise, it reissues the message DXC200R.

C
Cancel this ALCS job.

Only use C if the problem cannot be corrected.

Procedure when the prime ALCS job abends

When a prime ALCS job abends, and there are one or more alternate ALCS jobs in standby state, proceed as follows:

1. Ensure that the alternate ALCS job has access to all the real-time database DASDs (the DASDs must be online to MVS).
2. Reply to the outstanding DXC110R message for one of the alternate ALCS jobs.

   The selected job then becomes the prime ALCS. It completes initialization and starts to process input messages.
3. If this was the only alternate ALCS job, start another alternate ALCS job. When it reaches standby state do not answer message DXC110R for this new job.

Your MVS or ALCS system programmer should set the MVS dispatching priorities of ALCS jobs so that when you reply to DXC110R, the new prime ALCS job takes priority over other jobs on the processor.

ALCS does not read or update the real-time database data sets until after you reply to message DXC110R. After you reply to DXC110R, the ALCS job must have access to other resources such as the communication network before it can be fully operational.

Multiple active ALCS jobs

Sometimes you might need to run two or more active ALCS jobs simultaneously (that is, two or more ALCS jobs in system states IDLE, CRAS, MESW, or NORM). An example is where there is an ALCS production job and test job running on the same machine. (The different jobs should not update the same database.)

When you are running multiple ALCS jobs, your MVS or ALCS system programmer must set the MVS dispatching priorities to reflect the relative importance of each. (The production ALCS should have a higher priority.)

Identifying ALCS jobs

MVS helps you distinguish between the two or more ALCSs by using two unique identifiers:

The VTAM application program minor node name
This is the logical unit (LU) name of an ALCS application. It is sometimes called the "VTAM ACB name".

The ALCS system identifier
This is a single alphabetic character that appears in all ALCS messages and is defined by your ALCS system programmer in the system generation.

Setting threshold values for the long-term pool activity monitor

The ALCS system monitors the rate at which pool file is dispensed from the long-term pools and it uses these dispense rates to predict when there will be no more pool records available for one or more long-term pools. It also compares these dispense rates against threshold values to determine if excessive numbers of pool addresses are being dispensed (which could lead to an early depletion of one or more long-term pools). This monitoring is performed by the ALCS long-term pool activity monitor which continuously monitors pool file usage and issues attention messages to RO CRAS when the dispense rate exceeds a threshold rate or when a long-term pool is close to becoming depleted.

Although the long-term pool activity monitor can play a vital role in monitoring the pools, its effectiveness is completely dependent on the correct threshold values being initialized on the ALCS system (via the ZP00L command). ALCS itself is unable to determine suitable threshold values for each long-term pool, therefore it sets initial threshold values of zero. While the ALCS system is running with these zero threshold values, messages will be output to RO CRAS stating that threshold values have not been set. You must therefore use the ZP00L command to set suitable values to enable ALCS to effectively monitor long-term pool usage. The following provides guidance on how you can gather the information required for setting the appropriate long-term pool threshold values.

ALCS V2.4.1 includes the ZP00L L*n*,DISPLAY command. This provides a display of the current threshold values plus the counts of pool dispensed during each of the last 24 intervals (the default interval time is 10 minutes). The pool dispense information in this display provides a valuable indication of the rate of pool dispensing over short periods of time and over longer periods of time. Use this display over a one week period to determine the peak dispense rates during long periods of pool dispensing and during short periods.

The long-term pool activity monitor is designed to accommodate fluctuations in pool dispensing that occur over a 24 hour period. For example, the count of pool dispensed during a period of high activity on the ALCS system is always much higher than during a period of low activity. Pool dispensing can increase dramatically when a maintenance function is started and can remain high until it completes. Pool dispensing is therefore monitored by ALCS over short periods of time and longer periods of time. Two threshold values must be set for pool monitoring over short periods of time and another two threshold values must be set for pool monitoring over longer periods of time.

It is vital that suitable threshold values are set for the ALCS system. If values are set too low, you will see unnecessary warning messages being output on RO CRAS, and after a period of time will begin to ignore these warnings. If values are set too high, then warning messages will not be output when there is excessive usage of the long-term pools, and you will be unaware that there could be a serious problem on the ALCS system.

Setting the short threshold values

A short rate threshold value must be set which tells ALCS what the maximum dispense rate per second should be for periods of high act ivity and for periods when maintenance functions are running. A short time threshold must be set which specifies the minimum pool that must be available (this threshold value is expressed in hours as the time to exhaustion based on the current short rate). If either of these thresholds is exceeded, ALCS outputs a warning message to RO CRAS. Review the 24 pool dispense counts output by each ZP00L L*n*,DISPLAY command and identify the highest value output during a one week period (for a single interval). Increase that value by approximately 20% and set the short rate

threshold to that value. Set the short time threshold to a value that gives sufficient time for a recoup to be run (for example, if recoup takes two hours to run, set the short time threshold to three hours.

Setting the long threshold values

A long rate threshold value must be set that tells ALCS what the average dispense rate per second should be over a longer period of time (that period of time is the interval time multiplied by 24). A long time threshold must be set which specifies the minimum pool that must be available (this threshold value is expressed in hours as the time to exhaustion based on the current long rate). If either of these thresholds is exceeded, ALCS outputs a warning message to RO CRAS. Review the smoothed pool dispense count output by each ZP00L L*n*,DISPLAY command and identify the highest value output during a one week period. Increase that value by approximately 20% and set the long rate threshold to that value. Set the long time threshold to a value that gives sufficient time for a recoup to be run (for example, if recoup takes two hours to run, set the long time threshold to three hours.

Setting the minimum available records threshold value

A threshold value must be set which indicates the minimum number of available records that should remain in the long-term. If the count of available pool records falls below this value, ALCS outputs a warning message to RO CRAS. Set this threshold to a value that is a percentage of the total addresses in the long-term pool (for example, to 5% of the total pool count). Alternatively, set this value to a count that would indicate that the pool is close to being depleted (and would give sufficient time for a recoup to be run).

Setting the interval time for pool monitoring

ALCS accumulates the number of long-term pool dispenses over 24 consecutive time intervals (the ZP00L L*n*,DISPLAY command outputs a count for each interval and a smoothed count for all intervals). The time interval used for pool monitoring can be any time between 2 and 60 minutes (the default interval is 10 minutes). If the time interval is set quite low (for example, to 4 minutes), ALCS will give warning messages soon after the short rate threshold value is exceeded. If the time interval is set to a longer time (for example, to 45 minutes), ALCS will give warning messages only after the short rate threshold has been exceeded for a number of minutes (therefore ensuring that sudden fluctuations in pool usage do not trigger unnecessary warning messages).

Every minute, ALCS samples the long-term pool counts and calculates the amount of pool depleted during the preceding interval's worth of minutes. This is shown in the ZP00L response as ***pool usage during the last smoothed interval***. ALCS also calculates the dispense rate and time to depletion based on the preceding interval's worth of depleted records, and compares these with the short threshold values.

At the end of every time interval, ALCS calculates the amount of pool depleted during that interval and stores amounts for the most recent 24 intervals. These are shown in the ZP00L response as ***pool usage during the last 24 intervals***. ALCS also calculates the dispense rate and time to depletion based on the preceding 24 intervals' worth of depleted records, and compares these with the long threshold values.

Operational procedures for the communication networks

ALCS requires a VTAM network. In addition, it can optionally use other networks:

- Transmission Control Protocol / Internet Protocol (TCP/IP)
- Synchronous Link Control (SLC)

ALCS can also optionally use MQSeries® queues for sending and receiving messages.

ALCS can also optionally communicate with applications in WebSphere Application Server for z/OS using optimized local adapters (OLA).

Figure 2 on page 2 shows the relationships between ALCS, VTAM, and the possible communication links.

The VTAM network

VTAM must be started before you start ALCS.

You can use VTAM operator commands to control the VTAM network. ALCS provides a command (ZACOM) to establish and terminate sessions between ALCS and VTAM logical units (LUs).

Before a VTAM terminal can use ALCS, a session must be established between the terminal and ALCS. This can be done in one of the following ways:

- ALCS acquires any VTAM terminals defined as initially active in the ALCS generation.
- Use the ZACOM command to acquire a terminal. See "ZACOM -- Alter communication resource information" on page 69 for details.
- Your system programmer can specify to VTAM that it must establish a session whenever ALCS becomes active by defining a default logon application for the terminal.
- The end user can enter a VTAM command to establish the session (for example, a LOGON command).
- Use a VTAM command to establish the session between the terminal and ALCS.

Notes:

1. If a session is to be established automatically at start up, ensure that ***either*** ALCS acquires the terminal ***or*** VTAM initiates the session. If both try to do so, one or both will report an error, and the session may not be established.

Closing down the VTAM network

If you need to close down the VTAM network while ALCS is running, you can do it in one of two ways:

- Issue the VTAM command HALT NET without any additional operands. VTAM and ALCS continue operation, but ALCS sends message number DXC204A to the RO CRAS.

```
VTAM operator has issued HALT -- Halt ALCS with ZASYS HALT command
```

  You can then halt ALCS in a controlled manner and retry the HALT NET command.
- Use the VTAM command HALT NET,QUICK. This command completes immediately, and ALCS terminates abnormally.

  Note: Use this method only in emergencies.

The TCP/IP network

Telnet display and printer terminals can access ALCS using TN3270 or TN3270E, through the Telnet server on z/OS. ALCS controls these terminals in the same way as VTAM 3270 terminals.

For other TCP/IP communication (including connections defined in the ALCS communication generation, applications that use TCP/IP Sockets requests, E-mail, and the ALCS Web Server) ALCS must be connected to an IP stack running in the same MVS image.

ALCS provides a command (ZCTCP) to establish and terminate a connection ALCS and the appropriate TCP/IP address space.

You can use TCP/IP operator commands to control the TCP/IP network.

ALCS starts any TCP/IP connections defined as initially active in the ALCS communication generation.

Use the ZACOM command to start and stop connections defined in the ALCS communication generation. See "ZACOM -- Alter communication resource information" on page 69 for details.

The SLC network

The SLC network consists of up to 255 SLC ***links***, each consisting of one to seven ***channels***, numbered 1 to 7. An SLC link can send and receive messages when at least one channel of the link is ***opened*** and ***started***.

You can control the SLC network using the ZACOM command.

You can perform the following control functions on:

- Each SLC channel independently
- All the channels in a specific SLC link simultaneously
- All SLC links simultaneously

See "ZACOM -- Alter communication resource information" on page 69 for details of the ZACOM command.

Open
Open one or more SLC links or channels. ALCS does this during a system state change from IDLE state to a higher system state for all SLC links that your system programmer has specified as initially active during ALCS generation. Use ZACOM to do this at other times.

Start
Start transmitting and receiving data on one or more SLC links or channels. ALCS does this during a system state change from IDLE state to a higher system state for all SLC links that have been defined as initially active. Use ZACOM to do this at other times.

Stop
Stop transmitting and receiving data on one or more SLC links or channels. ALCS does this during a system state change to IDLE from a higher system state, for all the SLC links that are currently started.

Close
Close one or more SLC links or channels.

Display
Use ZDCOM to display the status of a channel or link in the SLC network. Use ZLKST to display current counts of messages and control blocks sent and received over the SLC network.

Operational procedures for the database

This section describes:

- Backing up and restoring ALCS data sets
- Operating with a duplicated database
- Recovering after hardware or software problems

Backing up and restoring ALCS data sets

It is important that you establish (and use) procedures for regular backup of the ALCS datasets to tape or to other DASD volumes.

There are various ways you can backup and restore data from an ALCS database. Use one of the following:

The ALCS ZDATA command
Use this command to backup and restore individual records or groups of records. For example, you can backup (dump) all fixed records of a particular type. Later you can use the same command to restore (load) some or all of these fixed records.

See "ZDATA -- Load or dump DASD records" on page 132 for a description of this command.

The command ZDATA is not suitable for backing up a complete ALCS real-time database.

The IBM Data Facility Data Set Services (DFDSS), product or equivalent
You can use DFDSS to backup and restore individual DASD data sets, ranges of tracks, complete DASD volumes, or a complete ALCS real-time database.

Data Facility Data Set Services: User's Guide and Reference describes how to use DFDSS.

You can backup data using either ZDATA or DFDSS while ALCS is running in any system state and while it is updating the real-time database. To reload the database using ZDATA LOAD, ALCS must be in IDLE state.

Logging and restoring database updates

During normal operation, ALCS logs copies of records written to the database to the ALCS log file or files. This process is automatic and always active. Later you can use this log file (or files) to bring the database up-to-date after a restore from a backup.

There are three types of logging:

Forward logging
When a database record is updated, ALCS writes the new record to the log file.

Backward logging
When a database record is updated, the old record contents are written to the log file.

Forward and backward logging
When a database record is updated, both the old and the new contents are written to the log file.

The ALCS log file can be on either tape or DASD. When it is held on DASD, you must move closed log files out to tape or other DASD volumes to ensure that sufficient DASD space is always available.

ALCS puts a time-stamp in each logged copy of a record. This lets you restore records from a particular start time (usually the time of the last DASD backup) up to a particular stop time (usually the time when the prime ALCS abended, or when ALCS discovered a corrupted or unusable data set). You use the ZRSTR command to restore the database to an up-to-date form.

ALCS must be in IDLE state when you run the restore function.

Backing up and restoring the real-time database

You should backup the real-time database regularly. With this backup you should also run your restore procedures regularly on a test system to confirm that the backup and restore procedures are correct and the logging options on your system are sufficient. In the event of a DASD hardware failure or corruption of the database you need the backup copy of the database to restore the database to a consistent state.

There are various ways to perform backup and restore. These are described in "Backing up and restoring ALCS data sets" on page 17.

Provided that the backup program allows it (both ZDATA and DFDSS do allow it), the backup can run while ALCS is updating the database. After a restore, the ALCS ZRSTR command can reapply the changes that occur during and after the backup. In this way, a DFDSS restore followed by an ALCS ZRSTR can restore the database to a self-consistent state.

Note: It is not necessary to backup the whole of a large real-time database at the same time. For example, an installation can backup a proportion of the real-time database each day. In this case the database is not self-consistent - some parts of the database are older than other parts. But if it becomes necessary to restore the complete real-time database, you can use ZRSTR to reapply all changes to the database from the time of the start of the oldest backup. This restores the database to a self-consistent state.

See "ZDATA -- Load or dump DASD records" on page 132 and "ZRSTR -- Restore logged records" on page 286 for further details.

When establishing schedules and procedures for database backup, be sure to consider the following:

- Some backup programs do not allow any other program to update the data during the backup. To allow backup without interruption to the ALCS service, choose a backup program (for example DFDSS) that does allow this.
- The backup program competes with ALCS for access to the data. Depending on the backup program, this can affect ALCS performance, or slow down the backup, or both. To minimize these effects, avoid running backups during periods of high activity on the ALCS system.
- If a restore becomes necessary, ALCS is "down" for the duration of the restore plus the time it takes ZRSTR to reapply changes. To reduce this time as far as possible:
  - Choose a backup and restore method that has a fast restore.
  - Take frequent backups; this reduces the number of changes that ZRSTR must reapply.
- If the corruption or loss is restricted to a small number of records, it is possible to avoid a restore. For example, depending on the records, it is possible to:
  - Reinitialize the records; obviously this discards any data that the records contain. This is acceptable if the data is not valuable, or if the cost of losing the data is less than the cost of a system outage.
  - Manually repair the records using, for example, the ALCS `ZAFIL` command or a functionally equivalent program. This command is described in “ZAFIL -- Alter DASD record” on page 85.

Following a full or partial database restore, including the ZRSTR command, which restores the logging updates, there might still be inconsistencies in the database. This happens because ALCS does not log **all** updates. You should use Recoup to identify any remaining inconsistencies.

You should not:

- Run Recoup and a database backup procedure concurrently.
- Run a partial database backup, then Recoup, and then the rest of the database backup.

If you follow either of these procedures you **must** run Recoup to eliminate inconsistencies in the database, before making the system available to end users. The procedures are explained in Table 4 on page 19 and Table 5 on page 20.

Procedures for restoring the real-time database

After a DASD hardware failure part of the database will not be available to ALCS. After corruption of the database it will not be in a consistent state. In either event you will need to do a full or partial restore of the database and then run Recoup.

Table 4 on page 19 shows the recommended procedures to follow to make ALCS available again for the end users and to minimize the risk of inconsistencies in the database in the event of corruption of the database.

Table 5 on page 20 shows the recommended procedures to follow in the event of a DASD hardware failure.

Table 4. Procedure for restoring the database after a software database corruption

Database restore	**Partial (one or more records or record types)**	**Full (all records)**
DFDSS	This procedure is not possible.	This is a recommended procedure: 1. Run DFDSS restore 2. Run ZRSTR 3. Make the system available to the end users 4. Run Recoup

Table 4. Procedure for restoring the database after a software database corruption (continued)		
Database restore	**Partial (one or more records or record types)**	**Full (all records)**
ZDATA	This is a recommended procedure: 1. Run ZDATA with the include parameter to restore the applicable records 2. Run ZRSTR 3. Make the system available to the end users 4. Run Recoup	This procedure is not possible.

Table 5. Procedure for restoring the database after a DASD hardware failure		
Database restore	**Partial (one or more datasets)**	**Full (all data sets)**
DFDSS	This is a recommended procedure: 1. Run a partial DFDSS restore 2. Run ZRSTR 3. Run Recoup 4. Make the system available to the end users	This is a recommended procedure. 1. Run a partial DFDSS restore 2. Run ZRSTR 3. Run Recoup 4. Make the system available to the end users
ZDATA	This procedure is not possible.	This procedure is not possible.

Pool dispensing array for restore -- PDAR

The pool dispensing array for restore (PDAR) is a table structure that contains reserved long term pool records used only for restore.

By reserving these records ALCS avoids possible corruption when a certain pool record that is recognized as free after the restore is used by the application when its contents is changed by updates found on the Log tapes.

At start up the ALCS system must be notified that a restore is to be processed and pool records from the PDAR structure should be used.

See “Job control statements to run ALCS” on page 7 for further details.

If the ALCS system is started with the RESTORE state parameter pool dispensing will take place from the PDAR structure first. If the PDAR structure does not exist the standard ALCS pool dispensing mechanism is used instead. The standard pool dispensing mechanism is also used after the reserved PDAR records are exhausted

If the ALCS system is restarted without the RESTORE state parameter the PDAR structure is ignored and the reserved records found in the PDAR structure are not dispensed by the application.

The PDAR structure is created at system restart unless it already exists. It can also be created, deleted and displayed using the ZPDAR command. See “ZPDAR -- Maintain and display the PDAR Table” on page 256 for further details.

At PDAR creation the number of reserved records for each long term pool interval is determined by system defaults. For each interval the number of reserved records is 1/1000 of the total number of available records for the interval with 500 as the maximum value and 10 as the minimum value.

An installation-wide ECB-controlled exit is provided for the user to change those defaults.

See "Pool dispensing array for restore exit program - APDR" in

the *ALCS Installation and Customization* for further details.

The PDAR structure does not support short term pools. The standard ALCS dispensing mechanism is always used for short term pools.

Duplicated databases

An ALCS database can be fully duplicated. You can configure a duplicate real-time database during ALCS generation. In this case, there are two copies of each real-time data set, here referred to as copy-1 and copy-2. ALCS can continue processing normally if either copy-1 or copy-2 of a data set becomes unusable, for example after a component failure.

Alternatively, you can duplicate the database using the IBM 3990 Dual Copy Facility. In this case, ALCS "sees" only one copy of the database and the hardware takes care of the duplication.

Performance consideration

For best performance, put the copy-1 and copy-2 data sets on separate DASD actuators. If possible, put the copies on separate DASD strings, controllers, and channels.

When making backups of data sets using DFDSS, you should backup copy-2 of each data set rather than copy-1. (The performance impact on ALCS is slightly less if you use copy-2 for the backup.)

Attention

If a data set is duplicated, be sure to backup a copy that is online to ALCS. Use the ZDASD command to verify that the data set is online to ALCS before starting the backup job. If you have to vary the data set offline (using the ZDASD command) during the backup, then discard that backup and run a new backup of the other copy.

Actions when one copy is lost

If only one copy of a data set is lost (for example, because of an equipment failure) you do not need to take any recovery action. ALCS remains operational, but notes which copy of the data set is unusable and does not use it.

When the (previously lost) data set becomes accessible again, use the ZDASD command to vary it online to ALCS. The ZDASD command, which works in any system state, automatically initializes (preformats) and restores data to the offline data set from the current copy during the vary online.

Note: If you need to replace one or more DASD volumes, use the MVS `IDCAMS DEFINE` command to allocate a new dataset before varying them online to ALCS.

ALCS Installation and Customization describes how to initialize data sets.

Actions when both copies lost

The following procedures assume you have made a DFDSS backup of the copy-2 data sets only (as recommended in “Performance consideration” on page 21).

Both copies lost due to hardware error

Assume the following sequence of events:

1. A hardware error destroys copy-2 of a data set. ALCS continues to run with only copy-1 (but it notes that the copy-2 data set is unusable).
2. A second hardware error destroys copy-1 of the data set. ALCS abends because there is now no good copy of the data set.

You cannot just restore the copy-2 data set from the backup. ALCS will not restart because it has recorded that the copy-2 data set is unusable.

Use the following procedure to recover from this situation:

1. Restore the copy-2 data set from the backup, using DFDSS. (Depending on the backup method, you may need to delete the corrupted/destroyed copy-2 data set before you do this restore.)
2. Rename the copy-2 data set to make it the copy-1 data set. You can use the MVS access method services IDCAMS program's ALTER command with the NEWNAME= parameter to do this. (Depending on the state of the corrupted/destroyed old copy-1 data set, you may need to delete the old copy-1 data set before you do this rename.)
3. Start ALCS and wait until it reaches IDLE state.
4. Use the ZRSTR command to restore logged data to bring the database up-to-date. The start time is the time copy-2 was dumped, the end time is the time when ALCS abended.
5. Allocate and initialize a new copy-2 data set.
6. ZDASD VARY ONLINE the new copy-2. ALCS will not use the copy-2 data set until a ZDASD VARY ONLINE completes successfully.

Both copies lost due to software error

Because both copies of the data set were good last time ALCS was running, if you simply restore the copy-2 data set from the backup, ALCS assumes that it can use copy-1 while you are doing the restore.

You need to take the following action:

1. "Hide" the copy-1 data set, so that ALCS cannot allocate it. There are several ways to "hide" the copy-1 data set, including the following:
   - Use IDCAMS ALTER command to rename the copy-1 data set. After an ALCS restart, use ALTER again to "unhide" the data set by renaming it back to its original name. (Note that you need two ALTER commands each time, one to rename the cluster and one to rename the data component of the data set.)
   - Use IDCAMS DELETE command with NOSCRATCH to remove the cluster from the catalog. After ALCS restarts, use IDCAMS DEFINE CLUSTER with RECATALOG to "unhide" the data set.
2. Restore the copy-2 data set from the backup, using DFDSS. (Depending on the backup method, you may need to delete the corrupted/destroyed copy-2 data set before you do this restore.)
3. Start ALCS and wait till it reaches IDLE state.
4. "Unhide" the copy-1 data set.
5. Use the ZDASD VARY ONLINE command to vary the copy-1 data set online. See “ZDASD -- DASD data set functions” on page 123 for details of this command.

Backing up and restoring the configuration data set

You should backup the configuration data set regularly. It is particularly important to do this before and after a change to the DASD configuration. You should use this backup as part of your routine operational procedures to restore onto a test system. This will ensure that your operational procedures are working correctly.

In the event that one or both configuration data sets becomes unreadable use one of the procedures described in “Actions when one copy is lost” on page 21 to recover.

Backing up and restoring general files

You can use any of the methods described in “Backing up and restoring ALCS data sets” on page 17 to backup general files, including ZDASD DUMP and logging. You do not need to backup the Recoup general file (general file zero).

Reconfiguring ALCS

The ALCS generation process, which is performed by your ALCS system programmer, creates a number of configuration tables.

A number of operator commands let you display and alter the online versions of these tables.

Updating the communication configuration

Display

Use the ZDCOM command to display the current contents of the communication configuration table. This is described in “ZDCOM -- Display communication resource information” on page 136.

Change the characteristics of existing communication resources

Use the ZACOM command to make changes to communication resources. This command is described in detail in “ZACOM -- Alter communication resource information” on page 69.

For example, you can:

- Make an ALCS communication resource active or inactive.
- Change the CRAS status of terminals and printers, for example:
  - Assign CRAS status to a terminal.
  - Remove CRAS status from a terminal.
  - Reassign Prime and RO CRAS to other terminals.
  - Change the fallback indicator.
- Alter some of the data in the communication configuration table for a resource, such as:
  - Associated device for the resource
  - "Message being processed" indicator (AAA hold)
  - Input routing for the resource

Adding, deleting, or replacing communication resources

You can use the ZACOM command to load a communication generation load module in which you can add, delete, or replace a resource. You need to work with your ALCS system programmer to perform these tasks. Proceed as follows:

1. Ask your ALCS system programmer for the name of the communication generation load module containing the changes.
2. Ensure that any resources to be deleted or replaced are inactive (use the ZACOM INACT command).
3. Load the communication generation load module with the command:

```
ZACOM LOAD,name
```

Changes made using ZACOM are not retained if you restart ALCS.

Online Communication Table Maintenance (OCTM)

This facility allows the dynamic reconfiguration of the communication network without scheduled outages. See Chapter 4, “Managing the OCTM Operation,” on page 46 for information about using OCTM.

Updating the sequential file configuration

Use the ZDSEQ command to display the current contents of the sequential file configuration table. “ZDSEQ -- Display sequential file status” on page 194 describes the use of this command in detail.

You can update the information about a sequential file in two ways:

Change the definition of a sequential file

Use the ZASEQ command to update an item in the existing sequential file table.

Adding sequential files

To add sequential files to the table, you must:

1. Ask your ALCS system programmer for the name of the sequential file configuration table containing the changes.
2. Load the update with the command:

```
ZASEQ LOAD,name
```

You cannot delete sequential files from the table without restarting ALCS.

Changes made using ZASEQ are not retained if you restart ALCS.

Updating the database configuration

Use the ZDASD command to load and backout changes to the database configuration. This command is described in detail in “ZDASD -- DASD data set functions” on page 123.

Loading a new DASD configuration table

ALCS provides offline facilities to create a new DASD configuration table. See *ALCS Installation and Customization* for details.

Use the ZDASD LOAD command to load these tables on to ALCS. Follow this with ZDASD CONFIRM once the tables have been loaded satisfactorily. If ALCS abends following a ZDASD LOAD but before a ZDASD CONFIRM the changes you are trying to load are lost.

See “ZDASD -- DASD data set functions” on page 123 for a full explanation of the ZDASD command.

Backing out a new DASD configuration table

It is possible to reverse the changes introduced by ZDASD LOAD so long as you have not issued a ZDASD COMMIT. Use the ZDASD BACKOUT command to do this. Its effect is to restore the DASD configuration to the state it was in before the ZDASD LOAD.

Committing to a new DASD configuration table

Some changes which are introduced by a ZDASD LOAD are irreversible, and therefore do not come into effect until the possibility of a backout is eliminated. Use the ZDASD COMMIT command to indicate that backout will not be required.

Displaying the DASD configuration table status

Use the ZDASD REPORT command to determine what DASD configuration loads have been done and to confirm the status of the current load. You cannot perform another ZDASD LOAD until either you have committed or backed out of the previous load.

Loading programs

You can use the ZPCTL command to load and unload application program load modules. The modules can be loaded for system-wide use or for use by a single terminal.

Changes made using ZPCTL are not retained if you restart ALCS.

Loading installation-wide exits

You can use the ZPCTL command to load and unload installation-wide exit program load modules and installation-wide monitor exit load modules. The modules are loaded for system-wide use.

Changes made using ZPCTL are not retained if you restart ALCS.

Chapter 2. Using offline programs to produce reports

This chapter describes how to use some of the offline programs that are supplied with ALCS. These include:

- ALCS communication report file generator
- ALCS diagnostic file processor
- ALCS statistical report generator
- ALCS cross reference facility
- ALCS OCTM offline support program

Running the ALCS communication report file generator

The ALCS communication report file generator (DXCCOMOL) is an offline program that builds the ALCS communication tables in exactly the same way as the ALCS online monitor, and then writes the tables to a sequential data set. This data set is called the ALCS communication report file. Use the communication report file generator to produce reports for the installation.

The communication report file generator creates the report file from the communication load modules that are in the communication load list. Run the communication report file generator in the following situations:

- Before loading an update load module.
- After loading a new or modified communication load module. This is to maintain a current record of the ALCS communication resources.

If the ALCS system is using the OCTM facility (OCTM=YES on the `COMGEN` macro), the communication report includes all the communication resources on the OCTM database as well as those defined in the offline communications generation. When OCTM is being used, the input for DXCCOMOL is the communications generation load module(s), plus a sequential file that contains all the communication resources defined in the OCTM database. The `OCTM Database Backup` function creates this sequential file (this is described in "ZOCTM -- Control online communication table maintenance" on page 238).

For those ALCS customers who are not using the OCTM facility, DXCCOMOL can still be used for creating a communications report. When OCTM is not being used, the input for DXCCOMOL is the communications generation load module(s).

The communication report file contains details of every communications resource and is in a format that allows the MVS IEBPTPCH utility to print selective details of each resource. If required, you can use a utility to sort the contents of the output file prior to printing it with IEBPTPCH (for example, sort the resources into resource ordinal number sequence).

Using ISPF panels to run the communication report file generator

You can use the ISPF panels to run the communication report file generator. The ALCS primary menu panel is shown in Figure 1 on page xvii and the ALCS Operations panel is described in "Using ISPF panels to operate ALCS" on page 3.

See *ALCS Installation and Customization* for general information on installing, customizing, and using ISPF panels.

Running the ALCS diagnostic file processor

The ALCS diagnostic file processor (DXCDTP) is an offline program that prints information from the ALCS diagnostic file, and also TCP/IP trace data from the ALCS TCP/IP trace facility. The information that it prints includes:

- System error dumps from the ALCS online monitor system error routines (see “Format of system error dumps” on page 362).
- Pool file error information from the pool file management and Recoup routines. This is explained in “Problem determination in ALCS” on page 354.
- Trace data from the ALCS trace facility (see “The ALCS diagnostic file” on page 307).
- Trace data from the ALCS TCP/IP trace facility (see “TCP/IP trace facility” on page 385).
- SLC link trace data from the ALCS SLC link trace facility (see “SLC link trace facility” on page 383).
- Messages and other data from the ALCS system test vehicle (STV) (see “System test vehicle” on page 386).

The diagnostic file processor does not print information written to the ALCS diagnostic file by Data Collection (see “Data collection” on page 390). Use the statistical report generator (see “Running the ALCS statistical report generator” on page 36) to print this information.

The diagnostic file processor does not print information written to the ALCS diagnostic file by installation-wide monitor exits using the UWSEQ callable service. Use your own offline program to print this information.

Note: The ALCS diagnostic file processor cannot process an ALCS diagnostic file data set that is still allocated to ALCS. Use the ZSSEQ command (described in “ZSSEQ -- Switch sequential file” on page 295) to close and deallocate the data set so that the ALCS diagnostic file processor can process it.

Global-tag table (DXCDTPGT)

If you specify the GLOBAL print option, the ALCS diagnostic file processor attempts to load (using the MVS LOAD macro) the diagnostic file processor global-tags table DXCDTPGT. See *ALCS Installation and Customization* for an explanation of how to create the diagnostic file processor global-tags table.

A sample diagnostic file processor global-tags table is supplied with IPARS -- ALCS V2 and is called DXCDTPGL. You can use a copy of this to create a global-tags table during the installation process. You must assemble and link edit the diagnostic file processor global-tags table to create a load module with the entry point DXCDTPGT.

If you have not created a global-tags table, or if you have not specified the load module library that contains it when you run the ALCS diagnostic file processor then the load does not work and message `CSV003I REQUESTED MODULE DXCDTPGT NOT FOUND` is produced. This is not necessarily an error; if the load does not work, the ALCS diagnostic file processor prints the global area directories without tags.

Diagnostic file processor control statements

These control statements specify which dumps and trace data (if any) to print, whether or not to print pool usage error information, and so on.

The appropriate use of these control statements can dramatically reduce the amount of printed output from the ALCS diagnostic file processor. This is likely to reduce costs and speed up problem determination. Provided the diagnostic files are retained, it is always possible to run the ALCS diagnostic file processor again to print more (or more detailed) information as and when required.

To avoid producing hardcopy, browse the output from the ALCS diagnostic file processor on a display. You can use, for example, System Display and Search Facility (SDSF) with a 3270 display or a workstation 3270 emulator that can display a full 120-character line.

General rules for coding control statements

Include control statements in the diagnostic file processor SYSIN data set to increase or decrease the printed output according to your requirements. To use the corresponding default, omit any control statement or operand.

The format of diagnostic file processor control statements is:

```
     keyword=operand,...      comments
```

The diagnostic file processor ignores any leading blanks. The first blank following the keyword delimits the statement; the diagnostic file processor ignores any comments that follow.

If more than one statement contains the same keyword, the effect is the same as one control statement that contains all the operands from both statements. For example:

```
CTL=YES,TABLES
CTL=YES,NOSLC
```

is the same as:

```
CTL=YES,TABLES,NOSLC
```

If the SYSIN data set includes conflicting control statements or operands, then the diagnostic file processor uses the last. For example, if the SYSIN data set includes:

```
CTL=YES
CTL=NO
```

the diagnostic file processor uses CTL=NO.

The diagnostic file processor ignores comment statements; that is, statements with an asterisk (*) in the first position.

CTLDUMP, OPRDUMP, and DUMP control statements

These control statements select which dumps (if any) and which parts of the dump(s) to print.

By default, the ALCS diagnostic file processor prints all the dumps from the input diagnostic file or files. However, provided that you keep the diagnostic files, you can print dumps only if you need them for problem determination. Processing with:

```
DUMP=NO
DUMPHEADER=YES
```

gives a list of the dumps on the input data set(s).

The following parameter descriptions list the defaults. These defaults are designed to reduce the volume of printout while still including the information that is most likely to be useful. You may need more (or less) information to solve some problems, and provided that you keep the diagnostic files, you can run an initial printout of a small part of the dump, and ask for additional information only if you need it.

The formats of these control statements are:

```
CTLdump={NO|YES}[,number_range,...][,print_option,...]

OPRdump={NO|YES}[,number_range,...][,print_option,...]

DUMP={NO|YES}[,number_range,...][,print_option,...]
```

Where:

CTLdump
The parameters refer only to control system error dumps (errors that the ALCS online monitor detects).

OPRdump
The parameters refer only to operational system error dumps (errors that application programs detect).

DUMP
The parameters refer to both control system error dumps and operational system error dumps.

NO
Do not print this type of system error dump. The diagnostic file processor prints the dump header (a single line) for each dump that it does not print, unless these lines are suppressed (see "DUMPHEADER control statement" on page 31).

YES
Print this type of system error dump, according to the selections specified in the following parameters:

number_range
Print this type of system error dump only if the system error sequence number is within the specified range or ranges. The diagnostic file processor prints the dump header (a single line) for each dump that it does not print unless these lines are suppressed (see "DUMPHEADER control statement" on page 31). Specify each *number_range* as either a pair of decimal numbers joined by a hyphen (-), or as a single decimal number. Leading zeros are optional.

For example, to print system error dumps 000700, 000800, 000801, and 000802, include a statement of the form:

```
DUMP=YES,700,800-802,...
```

Do not include more than 16 system error sequence numbers (a pair of numbers joined by a hyphen counts as two numbers) in one execution of the diagnostic file processor.

print_option
Do or do not print an optional part of the system error dump for this type of system error. Valid *print_option*s are as follows. Each option has a negative counterpart prefixed with NO (for example, NOGLOBAL); the description includes its description in (parentheses).

GLOBAL
Print (or do not print) the application global area.

IOCBS
Print (or do not print) I/O control blocks (IOCBs).

PROGRAMT
Print (or do not print) program management control tables.

SLC
Print (or do not print) SLC link and channel keypoint records for those SLC links that are open.

TABLES
Print (or do not print) monitor tables. These include the following:

MVS system diagnostic work area (SDWA)
Monitor keypoint record (CTKB)
Resource hold table
CRET table
Macro trace control area
Pool file control tables
Pool file directory records
Control/data areas for APPC, CPU loop TCBs, Monitor exits, WAS, MQ, OCTM, PDU, SQL, TCP/IP
Entry dispatcher work lists
Block list descriptors

ECB descriptors
ECB prefix
Monitor interface area
Monitor work areas.

UTABLES
Print (or do not print) the unformatted monitor tables area dump. This includes:

System configuration table
Sequential file configuration table
DASD configuration table
Monitor tables area.

VFA
Print (or do not print) the virtual file access (VFA) control tables.

VFABH
Print (or do not print) the VFA buffer headers. If NOVFA or NOVFARLT is specified, the diagnostic file processor ignores this option.

VFABUFF
Print (or do not print) the VFA buffers. If NOVFA or NOVFARLT or NOVFABH is specified, the diagnostic file processor ignores this option.

Note: Printing the VFA buffers produces very large volumes of printout that is rarely useful for problem determination.

VFARLT
Print (or do not print) the VFA record locator table. If NOVFA is specified, the diagnostic file processor ignores this option.

ENTRYSTG
Print (or do not print) the storage units, entry macro trace blocks, ECBs, and - if TABLES is specified - the ECB descriptors and ECB prefixes, in the section of the dump headed "ENTRY STORAGE FOLLOWS". That is, the section of the dump that contains all the entries.

Note: NOENTRYSTG does not affect printing storage that belongs to the active entry (if any) that appears at the start of the dump.

SUS
Print (or do not print) the contents of storage units (SUs). These appear in the section of the dump headed "ENTRY STORAGE FOLLOWS". That is, the section of the dump that contains all the entries. If NOENTRYSTG is specified, the diagnostic file processor ignores this option. SUS also determines whether the contents of the storage unit(s) that belong to the active entry (if there is one) are printed.

ENTRYSUM
Print (or do not print) the section of the dump that contains all the entries in summary form. This summary form includes a single line for each entry.

Note: This option does not affect printing storage that belongs to the active entry (if any) that appears at the start of the dump.

REFSTG
Show the contents of 4096 bytes of storage that includes the storage addressed by the program status word (PSW) and 4096 bytes of storage addressed by each of the general registers at the time of error, if that storage is accessible by ALCS.

If NOREFSTG is specified the dump shows the contents of 128 bytes of storage that includes the storage addressed by the PSW, and shows the contents of 64 bytes of storage that includes the storage addressed by each of the general registers at the time of error, if that storage is accessible to ALCS.

Default *print_option*s depend on the type of system error, as follows:

```
Control system              Operational system
error dumps                 error dumps

  GLOBAL                      GLOBAL
  IOCBS                       NOIOCBS
  PROGRAMT                    NOPROGRAMT
  SLC                         NOSLC
  TABLES                      NOTABLES
  NOUTABLES                   NOUTABLES
  VFA                         NOVFA
  VFABH                       NOVFABH
  NOVFABUFF                   NOVFABUFF
  VFARLT                      NOVFARLT
  NOENTRYSTG                  NOENTRYSTG
  NOSUS                       NOSUS
  ENTRYSUM                    ENTRYSUM
  NOREFSTG                    NOREFSTG
```

DUMPHEADER control statement

Use the DUMPHEADER control statement to suppress the printing of dump headers.

DUMPHEADER={YES|DUMP|NO}

Where:

YES
Print all system error dump headers found on the diagnostic files.

DUMP
Print all system error dump headers found on the diagnostic files, except the nodump dump headers.

NO
Print only system error dump headers of dumps selected for printing by the DUMP=YES,...,
CTLDUMP=YES,..., and OPRDUMP=YES,... options also specified.

FORMAT control statement

Use the FORMAT control statement to control formatting of the diagnostic file processor output.

This control statement overrides all other control statements, regardless of where it appears in the SYSIN data set.

The format of this control statement is:

FORMAT={YES|NO}

Where:

YES
Format information on the diagnostic file.

NO
Do not format information on the diagnostic file. Instead, print a hexadecimal dump of each record on the data set. The diagnostic file processor prints the records without checking the record contents; consequently other diagnostic file processor control statements have no effect with **FORMAT**=NO.

Note: This option is provided only for use when errors are suspected in the ALCS diagnostic file processor or ALCS itself.

MESSAGES control statement

Use the MESSAGES control statement to control printing of ALCS information and error messages. The format of this control statement is one of:

```
MESSAGES={NO|YES]
```

Where:

NO
Do not print information and error messages.

YES
Print information and error messages. "Information and error messages" on page 389 describes these messages.

POOLERR control statement

Use the POOLERR control statement to control printing of information about pool file, (these are pool file usage errors, Recoup errors, and release chain (RLCHA macro) errors). The format of this control statement is:

```
POOLerr={YES|NO|RECOUP|MACRO|RLCH}
```

Where:

YES
Print pool file usage error messages, Recoup error messages, and release chain error messages.

NO
Do not print pool file usage error messages, Recoup error messages, or release chain error messages.

RECOUP
Print only Recoup error messages. These are written to the diagnostic file during Recoup chain-chase.

MACRO
Print only pool file usage errors. These are written to the diagnostic file when an error is detected by the GETFC or RELFC monitor-request macro while obtaining or releasing a pool file record.

RLCH
Print only release chain error messages. These are written to the diagnostic file when an error is detected during the release of a chain of records by the ECB-controlled program CRLC.

SLCTRACE control statement

Use the SLCTRACE control statement to control printing of SLC link trace data. The format of this control statement is:

```
SLCTRACE={NO|YES[,start_time-end_time,...]}
```

Where:

NO
Do not print SLC link trace data.

YES
Print SLC link trace data.

start_time-end_time
Only print SLC link trace data that originated within the specified time range or ranges. Specify each *start_time-end_time* as a pair of times joined by a hyphen (-).

Specify each time as *hh*, *hh.mm*, or *hh.mm.ss*, where:

hh
Hours (24-hour clock format)

mm
Minutes

ss
Seconds

ALCS adds trailing zeros if you enter the time in an abbreviated form.

For example, to print SLC link trace output for the period from 12:00:00 to 13:35:00, include a statement of the form:

```
SLCTRACE=YES,12-13.35,...
```

Do not include more than eight SLCTRACE time ranges in one execution of the diagnostic file processor.

If one or more macro trace time intervals are specified on the DXCDTP SLCTRACE control statement, ALCS prints each time interval before the SLC link trace items to which it applies. For example:

```
SLC TRACE TIME INTERVAL 04.09.52-04.10.00
04.09.53.8 SLC W LINK20  KCN1 LCB=ENQ ATSN=00 LEN=004
04.09.54.4 SLC R LINK20  KCN1 LCB=ENQ ATSN=00 LEN=004
```

STV control statement

Use the STV control statement to control printing of STV messages, (that is, input messages from, and output messages to, STV). The format of this control statement is:

```
STV={YES|NO,CRI=(cri,...)}
```

Where:

YES
Print STV messages.

NO
Do not print STV messages.

cri
Up to five selected CRIs, separated by commas. For example CRI=(02011A,02011B,0211C). Only messages for the specified CRIs will be printed.

TCPTRACE control statement

Use the TCPTRACE control statement to control printing of TCP/IP trace data. The format of this control statement is:

```
TCPTRACE={YES|NO}
```

Where:

YES
Print TCP/IP trace data.

NO
Do not print TCP/IP trace data.

TRACE control statement

Use the TRACE control statement to control printing of ALCS trace facility output. The format of this control statement is one of:

```
TRACE={NO|YES|MINimum}[,start_time-end_time,...]
```

Where:

YES
Print ALCS trace facility output in full.

NO
Do not print ALCS trace facility output. Note that any trace start and stop messages are printed even if TRACE=NO is specified.

MINimum
Print one line of trace facility output for each monitor-request macro traced by the ALCS trace facility.

start_time-end_time
Only print trace messages that originated within the specified time range or ranges. Specify each *start_time-end_time* as a pair of times joined by a hyphen (-).

Specify each time as *hh*, *hh.mm*, or *hh.mm.ss*, where:

hh
Hours (24-hour clock format)

mm
Minutes

ss
Seconds

ALCS adds trailing zeros if you enter the time in an abbreviated form.

For example, to print trace messages for the period from 12:00:00 to 13:35:00, include a statement of the form:

```
TRACE=YES,12-13.35,...
```

Do not include more than eight TRACE time ranges in one execution of the diagnostic file processor.

If one or more macro trace time intervals are specified on the DXCDTP TRACE control statement, ALCS prints each time interval before the macro trace items to which it applies. For example:

```
MACRO TRACE TIME INTERVAL 07.32.00-07.40.00
07.32.18 ECB-03F715A0  PROG-CFMS  ADDR-00F2  MACRO-FINPC TERM-0200EC   ...
07.32.18 ECB-03F715A0  PROG-CFMS  ADDR-00FC  MACRO-ENTRC OPERANDS-C1C3D...
```

TRANSLATE control statement

Use the TRANSLATE control statement to determine how much (if any) of the error dump information is to be translated from hexadecimal into character form. This allows dumps to be printed in the most readable form that the printer permits, bearing in mind that some printers have restricted character sets.

The section “Format of system error dumps” on page 362 shows many examples of how the ALCS diagnostic file processor prints storage contents. For storage areas that can contain character data, the ALCS diagnostic file processor prints the storage in hexadecimal notation. However it "interprets" into character format any sequences of 2 or more consecutive bytes containing printable characters (as defined by this control statement).

Interpreting can make dumps easier to read, for example by showing message texts in storage as character strings. However, storage can contain character values by chance, and specifying TRANSLATE=ALL greatly increases the probability that non-character data (such as addresses) is printed with an unhelpful translation.

TRANSLATE does not convert lower-case letters into upper case, but if your printer does not have lower-case letters, you can probably fold them into upper case by other means. If your system is set up to do this, you will not be able to distinguish upper from lower case; if that is important, and you are willing to read the lower case in hexadecimal format, then you should avoid the statement TRANSLATE=(LOWER).

The format of the TRANSLATE statement is:

```
TRANSLATE=(option,...)
```

Where option is:

UPPER
Treat the following characters as printable:

- A through Z (upper case)
- 0 through 9

LOWER
Treat the following characters as printable:

- a through z (lower case)

SPACE
Treat the space (X'40') character as printable.

SPECIAL
Treat the following special characters (shown in hexadecimal, with the character below[1]) as printable:

```
6B 5E 7A 4B 6F 5A 7D 7F 4D 5D 4C 6E 7E 4E 60 5C 61 6D 50 7B 7C 6C 4F
,  ;  :  .  ?  !  '  "  (  )  <  >  =  +  -  *  /  _  1 #  @  %  |
```

and also the "primary currency symbol" (X'5B').

NO
Do not carry out any interpretation. Print everything in hexadecimal notation.

ALL
Same as (UPPER,LOWER,SPACE,SPECIAL).

[1] The actual characters that print in your installation may be different, depending on the National Language character set in use.

The default is TRANSLATE=UPPER. Other subparameters (except, of course, for NO) extend the scope of translation. That is, it is not necessary to specify `TRANSLATE=(UPPER,LOWER)`, `TRANSLATE=LOWER` would have the same effect.

Using ISPF panels to run the diagnostic file processor

You can use the ISPF panels to run the diagnostic file processor. The ALCS primary menu panel is shown in Figure 1 on page xvii and the ALCS Operations panel is described in "Using ISPF panels to operate ALCS" on page 3.

See *ALCS Installation and Customization* for general information on installing, customizing, and using ISPF panels.

Running the ALCS statistical report generator

The ALCS statistical report generator (DXCSRG) is an offline program that prints statistical reports. These reports are based on information that the ALCS data collection facility writes on the ALCS data collection file or the ALCS diagnostic file. These are described in *ALCS Installation and Customization*.

The report is in six parts:

- System load summary
- Message mix by action code (bar chart)
- ECB processing summary
- Record access mix by record type (bar chart)
- Record access mix by record ID (bar chart)
- Program usage mix by called program (bar chart)

Use the ALCS diagnostic file processor to print other information on the diagnostic file.

Note that the ALCS statistical report generator cannot process an ALCS data set that is still allocated to ALCS. Use the ZSSEQ command (see "ZSSEQ -- Switch sequential file" on page 295) to close and deallocate the data set so that the ALCS statistical report generator can process it.

The ALCS statistical report generator invokes the MVS sort utility DFSORT. If you get an abend with a user completion code when running DXCSRG, the user completion code is always a DFSORT user completion code. These are not listed in *ALCS Messages and Codes*; see *DFSORT Application Programming Guide* for details of any errors from DFSORT.

System load summary

The first page of DXCSRG output summarizes the collected data. It indicates the load on the system at the time of data collection. If DXCSRG processes the output of multiple runs of data collection, it produces multiple copies of this page (one for each data collection run).

Duration of collection interval

The duration of the data collection run. Allow enough time to smooth out the instantaneous peaks and troughs in system activity; for example, 1800 seconds (30 minutes).

Collected items

The total for each class of data.

Mean input messages per second

The number of input items for the message type, divided by the duration of the collection interval.

Mean I/O operations per second

The average number of **logical** I/O operations per second that is, reads to or from the VFA buffer. Some of these logical I/O operations create **physical** I/O to or from DASD.

Task wait time

The percentage of total processor time that ALCS has an MVS task wait outstanding. It is not the MVS system wait time. (For example, VTAM can be doing processing for ALCS during this time.)

In a multiprocessor system, it is possible to have MVS task waits outstanding on more than one processor at the same time. In this case task wait time can exceed 100% because the elapsed times of all MVS task waits are added together.

VFA RLT synonym depth

This is the number of entries in the overflow record locator table (RLT) that ALCS addresses before finding the correct entry. It is an indication of the amount of processing required to locate a record in the virtual file access (VFA) area of storage.

Buffers

These figures indicate VFA buffer usage. VFA selects buffers for reuse from an age list. This is a list of buffers in order of elapsed time since the last reference. VFA buffers are either on the age list or in use. Buffers on the age list are either free or contain file pending records. Buffers in use contain records that are either permanently resident or are held. Buffers containing held records stay in use until the records are released with unhold type macros.

Message mix by action code

This bar chart shows messages by action code as a percentage of all messages (except WTTY messages and Type 2, Type 3, Type 4, and Type 5 X.25 PVC messages). The scale of this bar chart varies to suit the highest percentage.

ECB processing summary

This ECB analysis shows the relative system load for the various types of ECBs and action codes.

The action codes can include:

#EOM
"Enter" key pressed (with no data)

#EOU
"Clear" key pressed

#EOI
Printer acknowledgment (ACK) or answerback

DXCSRG prints the following statistics for each type of ECB:

Percent of ECBs
Percentage of all ECBs that are this type.

FINDs per ECB
Average number of times per ECB that an entry reads a record from VFA. This is a **logical** read.

READs per ECB
Average number of times per ECB that ALCS does not find a requested record in a VFA buffer. ALCS reads the record from DASD. This is a **physical** read. This number is included in the number of FINDs per ECB.

FILEs per ECB
Average number of times per ECB that an entry writes a record to VFA. This is a **logical** write.

WRITEs per ECB
Average number of times per ECB that ALCS writes a record from VFA to DASD. This is a **physical** write. This number is included in the number of FILEs per ECB. If the database is duplicated, there are two physical writes for each WRITE counted here.

ENTERs per ECB
Average number of program enters per ECB.

Millisecs life
This is the duration of the ECB measured in milliseconds.

Record access mix by record type and by record ID

The first of these bar charts is for each record type (fixed file and pool file). The second is for each record ID. They show the percentage of all accesses for:

- Total requests to VFA
- Logical reads (FINDs)
- Logical writes (FILEs)
- Physical reads
- Physical writes
- Hiperspace reads

The scale of the bar chart varies to suit the highest percentage, and is based on the total requests to VFA. The total length of the FINDs and FILEs bars should be equal to the length of the VFA bar, and the length of the FINDs bar, for example, does not therefore indicate the percentage of total FINDs in the system.

These two charts are included in the report by default. You can choose to omit them, and enable the DXCSRG job to run faster, by specifying the NORAM parameter on your JCL or by selecting this parameter on the ISPF panels.

Program usage mix by called program

The bar chart shows calls to each program as a percentage of all program calls. This subdivides to show the percentage for each calling program. There is room for up to five calling programs for each called program. DXCSRG omits the names of calling programs if the number of calls is below the minimum that DXCSRG can print. This is the case where the percentage for a called program is apparently greater than the sum of its calling programs.

Using ISPF panels to run the statistical report generator

You can use the ISPF panels to run the statistical report file generator. The ALCS primary menu panel is shown in Figure 1 on page xvii and the ALCS Operations panel is described in “Using ISPF panels to operate ALCS” on page 3.

See *ALCS Installation and Customization* for general information on installing, customizing, and using ISPF panels.

Using the statistical reports

The statistical report generator (DXCSRG) can only report on ALCS resources. Use RMF reports together with DXCSRG output to give a complete picture of system performance. Always collect statistical data at peak time for long enough to include a representative message mix (say, 30 minutes). Use the ZSTAT command (described in “ZSTAT -- Display current system load” on page 297) to find the peak hour for message processing.

For capacity planning purposes, accumulate on a regular basis important statistics such as:

- Peak message rate (from DXCSRG)
- Processor utilization (from RMF)

- Database access rate (from DXCSRG and RMF)

Message profile

The basic unit of measurement in an ALCS system is messages per second. An example of system performance measurement is:

- 65% processor utilization at 105 messages per second.

It is, therefore, important to identify the profile of an average message; that is:

- Number of logical reads (FINDs)
- Number of logical writes (FILEs)
- Number of physical reads
- Number of physical writes
- Number of hiperspace reads
- Number of duplicated writes (if the database is duplicated)

The above information can be collated from the “ECB processing summary” on page 37.

The FINDs and FILEs are logical reads and writes. The number of physical reads and writes depends on the VFA hit rate. Refer to “VFA hit rate” on page 40.

Processor utilization

Use RMF to report on processor utilization. Processor utilization tends to increase and decrease in proportion with the message rate.

If processor utilization is higher than expected, check the following with previous reports:

Task wait time
If there is no change, the reason is probably not in ALCS but somewhere else, such as VTAM or MVS.

Message profile
Check to see whether the number of FINDs, FILEs or ENTERs for "All ECBs" in the ECB processing summary changes (see “ECB processing summary” on page 37). New applications can have this effect, and so can end users using the system in a different way. Refer to “Message mix” on page 398.

VFA hit rate
If the ratios of FINDs to READs, or FILEs to WRITEs changes, this is a change in the VFA hit rate. Refer to “VFA hit rate” on page 40.

System message life

The system message life (in milliseconds) in ALCS does not include time spent in VTAM or any communication link. In the ECB Processing Summary report, "Millisecs Life" is the entry life, and the system message life is the entry life multiplied by the average number of ECBs per message.

Database response time

A large part of the system message life is a result of the database response time. This is the time it takes to read a record from DASD. Message processing is delayed during this time. Multiplying the number of READs per message by the average actuator response time gives an approximate total database response time for a message.

RMF reports on actuator utilization and response time. Actuator response time depends on actuator utilization. Aim to keep DASD actuator utilization below 40%. Operating DASD actuators at greater than 40% utilization can cause sudden and large increases in database response time at peak message processing time. Although DASD writes do not directly delay message processing, they increase actuator utilization, which in turn increases read delays.

VFA hit rate

This is the percentage of physical reads and writes that VFA saves. It is a measure of the effectiveness of the VFA function in ALCS.

Use the "All ECBs" value in the ECB processing summary (see *ALCS Installation and Customization*) to calculate the VFA hit rate as follows:

1. Add together the number of FINDs and FILEs per ECB. If the database is duplicated, double the number of FILEs.
2. Add together the number of READs and WRITEs per ECB. If the database is duplicated, double the number of WRITEs.
3. Subtract the total number of READs and WRITEs per ECB from the total number of FINDs and FILEs per ECB.

The result is the number of physical reads and writes per ECB that VFA has saved. The VFA hit rate is the savings per ECB, expressed as a percentage of the total number of FINDs/FILEs per ECB.

Obtaining a record from a VFA buffer requires fewer instructions than reading it from DASD, so increasing the VFA hit rate reduces the load on the processor as well as on the DASD subsystem. These load reductions in turn improve the message response time.

The above VFA hit rate calculation is based upon ECB-related I/O. It is also important to check the relationship between total DASD I/O and ECB-related I/O. If the former is significantly greater, this may indicate a high level of pool errors.

It is difficult to predict the percentage of physical reads and writes that VFA can save, but 75% is a reasonable expectation. To increase the VFA hit rate:

- Allocate more storage for VFA buffers. After a certain point, expanding the buffers is of very little advantage. To find the right level for your system, set a very high initial level, then reduce the storage in steps, checking the VFA hit rate after each change. (Starting high improves performance at once.)
- Use the "Record Access Mix by Record Type" report to select the record types most frequently written to DASD. Consider changing the VFA option for these record types to delayed file or time-initiated file.
- Use the information on VFA buffer usage (see “System load summary” on page 36) to decide the number of VFA data buffers to allocate for each record size.
- Use short-term rather than long-term pool when you can. Short-term pool is particularly suitable for records containing non-critical data that are released within a minute or so of being dispensed. Remember that long-term pool file records have the "file immediate" VFA option, whereas short-term pool file records have the "delayed file" option. The report "Record Access Mix by Record ID" can help you identify candidates for short-term pool.

  If your application uses the same record ID for both critical and non-critical data, consider changing the application to use different IDs, so that the non-critical data can be stored in short-term pool.

VFA buffers

“Buffers” on page 37 explains VFA buffers. Until your installation has gained experience in monitoring and controlling the VFA hit rate, use the following guideline. Failure to meet any of the following criteria can indicate a need for more buffers:

On age list
: Mean value greater than 90% of the total number of buffers allocated. Minimum value greater than 60% of the total number of buffers allocated.

Free to use
: We have previously recommended that you should try to maintain at least 50 per cent of the buffers free to use. But you may find that increasing the number of VFA buffers *decreases* the number of free to use buffers. This is because when you increase the number of VFA buffers you increase the chance of saving writes on delayed file which, in turn, increases your VFA savings.

Running the ALCS cross reference facility

The ALCS cross reference facility (DXCXREF) is an offline program you can use to search libraries of assembler input source for text strings.

DXCXREF may in fact be used on any partitioned data set containing fixed-length 80-byte records (blocked or unblocked) where the performance of a more general utility would be considered unacceptable.

Because DXCXREF has been designed to work very fast on even the largest partitioned data sets, it does not provide a contextual parsing option. However, if you need to reduce excessive output volumes to manageable proportions, DXCXREF output is formatted to be easily post-processed by a simple REXX command procedure.

Using ISPF panels to search for text strings

You can use the ISPF panels to run the cross reference facility. The ALCS primary menu panel is shown in Figure 1 on page xvii and the ALCS Operations panel is described in "Using ISPF panels to operate ALCS" on page 3.

See *ALCS Installation and Customization* for general information on installing, customizing, and using ISPF panels.

Running the ALCS OCTM offline support program

See "Running the OCTM Offline Support Program" on page 49 for a description of this offline program and how to run it.

Running the CPU time report generator

You can use the ISPF panels to run the CPU time report generator. The ALCS primary menu panel is shown in Figure 1 on page xvii and the ALCS Operations panel is described in "Using ISPF panels to operate ALCS" on page 3.

The CPU time report generator formats and prints a line of output for each data collection ECB item:

```
------------------------------------------------------------
yymmm  hh.mm.ss  aa  iiiiii  rw  di  vh  llll  uuuu  %%%%...
------------------------------------------------------------
```

Where:

yyddd
Creation date (julian) for this entry.

hh.mm.ss
Creation time for this entry.

aa
Action code for this entry if there is one.

iiiiii
CRI for this entry if there is one.

rw
Count of DASD reads and writes for this entry.

di
Count of dispatches for this entry.

vh
Count of waits for record hold for this entry.

llll
ECB life for this entry in microseconds. If the ECB life exceeds X'FFFFFFFF' microseconds (about 71 minutes) then '**********' is printed.

uuuu
Estimated CPU time for this entry in microseconds. If the CPU time exceeds X'FFFFFFFF' microseconds (about 71 minutes) then '**********' is printed.

%%%%
Estimated CPU time as a percentage of ECB life. Each % symbol represents two percent.

Chapter 3. Operating the ALCS system

This chapter describes the different system states of ALCS and explains the procedures for initiating and terminating ALCS.

It also describes how to control some typical ALCS activities, including:

- Controlling the VTAM, TCP/IP, and SLC communication networks
- Recovering from system failure and restarting
- Backing up and restoring the database
- Logging and restoring database updates
- Reconfiguring ALCS

ALCS Concepts and Facilities describes how ALCS relates to the hardware and software in a typical installation. See Figure 8 on page 44 for a summary of an ALCS installation.

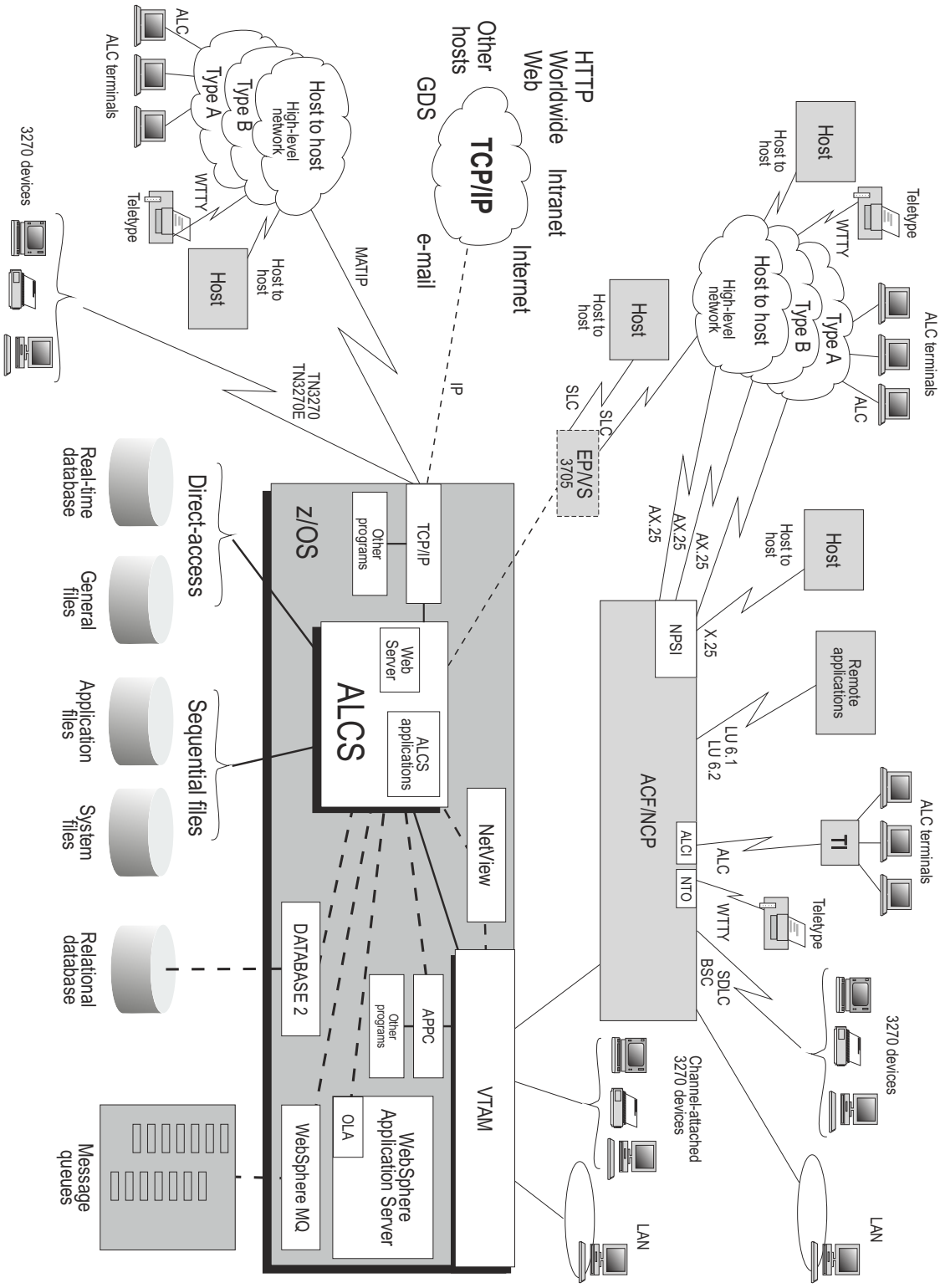


Figure 8. ALCS overview

Reconfiguring ALCS

The ALCS generation process, which is performed by your ALCS system programmer, creates a number of configuration tables.

A number of operator commands let you display and alter the online versions of these tables.

Loading the database

ALCS provides offline facilities, called System Test Compiler (STC), to create data records and to write them to a sequential data set called a data file. See *ALCS Installation and Customization* for details.

Use the ZDATA command to load records from a data file on to the ALCS database. For example, to load all records from the DAT data file, use the command:

```
ZDATA LOAD,SEQF=DAT
```

You can also create a data file using the ZDATA DUMP command. ZDATA DUMP writes selected records from the ALCS database to the specified data file.

"ZDATA -- Load or dump DASD records" on page 132 describes the ZDATA command in detail.

Chapter 4. Managing the OCTM Operation

When the OCTM facility is fully operational, consideration must be given to the ongoing monitoring and maintenance of OCTM. The OCTM facility includes a range of functions that allow this to be done. This section describes these functions and how they should be used.

Displaying OCTM status information

The OCTM database holds various information that is not directly related to the communications resources. This information can be displayed using ZOCTM commands. Three display functions are provided.

1. The ZOCTM STATUS command will display miscellaneous information such as the current OCTM status, number of groups currently allocated, number of physical records in use, etc.
2. The ZOCTM **GROUP**=*group_name* command will display detailed information about a currently allocated communications group. For example, the status of the group, date/time of last change (last COMTC for group), number of resources in group, the CRN of each resource that belongs to the group, etc. The information provided in this display can also be obtained by the CEUS via the COMTC QUERY macro.
3. The ZOCTM GROUPS command will display information about each group that is currently allocated. For example, the group name, number of resources in the group, date/time when group was allocated, date/time of last change to group, etc. The information provided in this display can also be obtained by the CEUS via the COMTC GROUPS macro.

Inhibiting access to the OCTM database

There may be occasions when the CEUS should be inhibited from accessing the OCTM database. Operator commands are therefore provided that can inhibit the usage of OCTM by the CEUS for short periods of time. When maintenance functions are performed on the OCTM database (for example, the OCTM database restore function, see “Running the OCTM database restore function” on page 47), usage of the OCTM facility by the CEUS is automatically stopped while the restore is running. Some ALCS users may wish to inhibit usage of the OCTM database when some of their own maintenance functions are being run. When the OCTM facility is stopped, the OCTM database can not be updated by the CEUS (via the COMTC macro) but the database will still be used for building the communications table during ALCS restart. The operator command ZOCTM STOP inhibits access to the OCTM database and the command ZOCTM START enables access to the OCTM database.

Running the OCTM database backup function

The OCTM database resides in system fixed file, therefore it will automatically be included in the daily backup of the ALCS database. OCTM will though allow a specific backup to be taken of the OCTM database. All the resource definitions in the base and update communication areas of the OCTM database are written by the backup function to a sequential file (called the OCTM sequential file). The operator command ZOCTM BACKUP performs this function. There are various reasons why a backup may be required.

Firstly, when an offline report is required of the communication resources currently defined in the communications generation and the OCTM database. The OCTM sequential file created by the backup function is used as input to the offline ALCS Communications Report program (DXCCOMOL). See “Running the ALCS communication report file generator” on page 26 for a description of this offline program.

Secondly, an OCTM database may need to be specifically created for a regression test system. If a regression test is to be run using a copy of the production system database, the OCTM database on the regression test system may need to be modified. For example, if an input message file containing messages captured from the production system is to be used by the ALCS STV facility, some of the

terminal definitions in the OCTM database must be changed to ***test*** resources (in the communications generation, this is defined by the TEST=YES option on the COMDEF macro). The OCTM sequential file created by the backup function could be used as input to the OCTM Offline Support program (DXCCTMOL). See “Running the OCTM Offline Support Program” on page 49 for a description of this program. This offline program enables changes to be applied to multiple terminal definitions and then writes the updated definitions to an output OCTM sequential file. This OCTM sequential file can be used as input to the OCTM database restore function (Z0CTM RESTORE) to rebuild the OCTM database on the regression test system.

Thirdly, if changes are required to multiple communication resources on the production system OCTM database, the OCTM sequential file created by the backup function could be used as input to the OCTM Offline Support program (DXCCTMOL) which can apply the required changes. For example, if the format of the communications user data area requires changing for all the terminal resources, the offline program can perform all the required reformatting. The OCTM sequential file output by DXCCTMOL contains the updated communication definitions, and this can be loaded onto the production system via the OCTM database restore function.

The following is an example of the SEQGEN definition (in the sequential file generation) that could be used to define the output OCTM sequential file that is required by the database backup function.

```
SEQGEN NAME=CMB,
       TYPE=GEN,
       UNIT=(3380,1),
       DISP=(NEW,CATLG,CATLG),
       DSNAME=.............CMB,
       LABEL=(,,,OUT,RETPD=0),
       RECFM=VB,
       VOLCNT=1,
       BUFNO=2,
       BLKSIZE=13000,
       SPACE=(1000,1000),
       LRECL=12900
```

Running the OCTM database restore function

About this task

The OCTM database restore function reads an OCTM sequential file and replaces the resource definitions in the OCTM database with those on the sequential file. The OCTM sequential file could be created by the OCTM database backup function (see “Running the OCTM database backup function” on page 46) or by the OCTM Offline Support program (see “Running the OCTM Offline Support Program” on page 49).

The steps required to perform the OCTM database restore function are as follows:

Procedure

1. Alter the system state to IDLE.
2. Use the Z0CTM RESTORE command to activate the OCTM database restore function. When the OCTM database restore function completes, the logical records on the OCTM database will contain the restored data.
3. Restart the ALCS system to rebuild the online communication table from the restored OCTM database.

Results

The following is an example of the SEQGEN definition (in the sequential file generation) that could be used to define the input OCTM sequential file that is required by the database restore function.

```
SEQGEN NAME=CMR,
       TYPE=GEN,
       UNIT=(3380,1),
       DISP=(OLD,KEEP,KEEP),
```

```
        DSNAME=.............CMB,
        LABEL=(,,,IN,RETPD=0),
        BUFNO=4
```

Expanding the communications table user data area

The OCTM database backup and restore functions can also be used to rebuild the OCTM database when an expansion of the communications user data area is required. The restore function builds new logical records in the base communications area on the OCTM database to accommodate an enlarged user data area. The steps required to perform the user data area expansion are as follows.

1. The size of the user data area is defined in the USERLEN= parameter on the communications generation COMGEN macro. Update the USERLEN= parameter on the COMGEN macro with the new size and run the offline generation process to create a new communications load module.
2. Create an OCTM sequential file by running the OCTM database backup function.
3. Restart the ALCS system to load the new communications load module (created by the offline generation).
4. When the ALCS system is in IDLE state, activate the OCTM database restore function to restore the OCTM sequential file. When the OCTM database restore function completes, the logical records on the OCTM database will contain the expanded user data area.
5. Restart the ALCS system again to rebuild the online communication table from the restored OCTM database. Each resource definition in the online communication table will now contain an expanded user data area.

Monitoring the output from OCTM Policing

An OCTM policing function monitors the contents of the base and update communications areas on the OCTM database. It checks each communications resource in the update communications area for periods of inactivity since the last COMTC macro was issued. If a period of inactivity is detected, OCTM policing sends a warning message to the RO Cras, indicating that further action is required. The first warning message is sent after 48 hours of inactivity, and further warning messages will be sent every 8 hours after that. These warning messages should be monitored as they indicate that a CEUS user (a system administrator) has submitted change requests to OCTM, but has not taken any action with them for the past 2 days (or longer).

The OCTM policing function does though activate an ECB-controlled exit program (AOCM) which can be used to issue a COMTC macro to automatically activate the next action that is required by the communication resource or group (for example, if the communications resource is waiting to be confirmed, the exit would issue the COMTC CONFIRM macro for that resource). There are four different COMTC macros that can be issued by the AOCM exit program:

- COMTC LOAD
- COMTC CONFIRM
- COMTC COMMIT
- COMTC UNALLOCATE

Implementating the AOCM exit program, with the automatic activation of the next COMTC action, can reduce the need to monitor the RO Cras for the warning messages output by OCTM policing.

Running the Offline Communications Report Program

The offline Communications Report File Generator program (DXCCOMOL) provides a report of the communication resources defined in the offline communications generation. If the OCTM facility is being used, DXCCOMOL can also include in its report, details of the communication resources that are managed by OCTM. The input for DXCCOMOL is the communications generation load module(s), plus the OCTM sequential file (created by the OCTM database backup function). The report provides a useful summary of

all the communication resources in the ALCS online communication table and therefore should be run on a weekly basis.

DXCCOMOL writes details of each communication resource to a sequential data set called the ALCS communication report file. It is in a format that allows the IEBPTPCH utility to print selective details of each resource. Different types of report can be obtained by using a utility to sort the contents of the communication report file prior to printing it with IEBPTPCH (for example, sort the resources into resource ordinal number sequence).

The name of OCTM sequential file data set name is defined to DXCCOMOL in a parameter on the JCL EXEC statement. The data set name can be any length (and can therefore overflow onto a second line), for example:

```
//        EXEC PGM=DXCCOMOL,PARM='COMSANJW,ALCSPROD.OCTM.SEQFILE.BACKU-
//            P.AUGUST'
```

The DXCCOMOL JCL does not require any DD statement for this OCTM sequential file because the offline program uses dynamic allocation to open the data set.

Running the OCTM Offline Support Program

The OCTM offline support program DXCCTMOL provides additional functionality for the management of the OCTM database. It can be used to apply a large number of changes to the communication resources managed by OCTM, it can be used to validate and implement new ranges of CRI addresses and resource ordinals (for the exclusive use of the offline communications generation), and it can also be used to create a dataset that contains an ALCS generation COMDEF macro for every communication resource on the OCTM database.

The following describes the functions provided by the OCTM offline support program DXCCTMOL and how they can be used.

- `Modifying and deleting communication resources`

  Use this offline program to apply a large number of changes to the communication resources managed by OCTM. The communication resource definitions on the OCTM database are normally modified and deleted via the COMTC macro, but when a large number of changes are required, they can be implemented more efficiently by using this OCTM offline support program. These changes can include modifications to multiple communication resources and they can also include the deletion of multiple communication resources.

  The primary input is an OCTM sequential file, created by the OCTM database backup function (activated by the ZOCTM BACKUP command). The primary output from this offline program is a new OCTM sequential file. This new sequential file contains all the communication resources that have not been deleted, and for those that have been modified, the updated communications definitions. The OCTM database restore function (activated by the ZOCTM RESTORE command) can be used to rebuild the OCTM database on the ALCS system from this new OCTM sequential file.

  This offline program activates an installation-wide exit program DXCUTMOL which identifies the modifications that are required to specific communication resources and also identifies the resources to be deleted. This exit is activated by the offline program for every communications resource on the input OCTM sequential file (which also includes recent in-progress change requests). See ALCS Installation and Customization for details of this installation-wide exit program and guidance on how it can be used to apply changes to the communication resources on the OCTM database. When this exit program is assembled and link-edited, the load module must be placed in one of the DXCCTMOL offline program STEPLIB load libraries. This is required because DXCCTMOL invokes the exit program dynamically during program execution.

  Use the DXCUTMOL exit program to apply the following types of change to the communication resources managed by OCTM.

  – Set on the "test" status for communications resources that will be used by the ALCS STV (ZTEST) function

- Change the "initial" status from active to inactive
- Modify the content and format of the communications user data
- Delete all the X.25 PVC and ALC communication resources
- and so on

The DXCCTMOL offline program outputs a report to the PRINT dataset giving the names of the communication resources that have been modified or deleted. When this program has been run, check the report to verify that the list of resources that have been modified or deleted is correct. If not correct, review the code in the DXCUTMOL exit program, and if changes are required, apply those changes and reassemble the exit. Link-edit the exit and re-run the offline program. Check the report and if it is correct, use the output OCTM sequential file created by this offline program to rebuild the OCTM database on the ALCS system.

- `Validating and implementing new CRIRANGE and ORDRANGE parameters`

  Use this offline program to validate and implement new ranges of CRI addresses and resource ordinal numbers (for the exclusive use of the offline communications generation). If the current ranges of CRI addresses and resource ordinals that are defined in the CRIRANGE and ORDRANGE parameters (on the communications generation COMGEN macro) require modification, use this offline program to validate those changes against the communication resources on the OCTM database. Define the new ranges to this offline program in the SYSIN dataset. The offline program identifies the communication resources on the OCTM database that are using any of the CRI addresses or resource ordinals within the new ranges. It reads an OCTM sequential file (created by the OCTM database backup function) and outputs a report to the PRINT dataset giving the name of each communication resource that has a CRI or ordinal within the new ranges. The report outputs the following message for each resource that is within the specified ranges:

  ```
  Resource  crn  deleted by CRI/ORD ranges
  ```

  When you define the new ranges to the offline program (in the SYSIN dataset), code them in one or more records, as follows:

  ```
  CRIRANGE=(080001,08006F,08009F,0800FF)
  CRIRANGE=(090001,0900FF)
  ORDRANGE=(100,150,250,280)
  ```

  Information on the format of these CRIRANGE and ORDRANGE parameters can be found in the description of the communication generation COMGEN macro in ALCS Installation and Customization.

  When you have run this program, check the report (in the PRINT dataset) to verify the list of resources that will be deleted from the OCTM database. If necessary, adjust the ranges of CRI addresses or ordinal numbers and re-run the program. A new OCTM sequential file is created by the program containing an updated OCTM control anchor record (with the NEW ranges of CRI addresses and ordinal numbers defined) plus all the communications resources that have NOT been deleted. Use this new OCTM sequential file to rebuild the OCTM database on the ALCS system (using the OCTM database restore function). Finally, update the CRIRANGE and ORDRANGE parameters on the communications generation COMGEN macro with these new ranges of CRI addresses and ordinal numbers.

- `Creating communications generation COMDEF macros`

  Use this offline program to create a sequential dataset which contains an ALCS generation COMDEF macro for every communication resource on the OCTM database. This enables communication generation decks to be created for all the resources on the OCTM database. Use this feature of the offline program when you are building a new ALCS system that will not use the OCTM facility (and when you require all the communication resources to be defined in the offline communications generation). Code PARM=LIST on the JCL EXEC statement to request this offline program to create the communication generation COMDEF macros. The program reads the input OCTM sequential file (created by the OCTM database backup function) and outputs the COMDEF macros to the LIST dataset.

The following is an example of the JCL that should be used for running the DXCCTMOL offline program.

```
//S1      EXEC PGM=DXCCTMOL,PARM=LIST
//STEPLIB DD  DSN=DXC.V2R4M1.DXCLMD1,DISP=SHR
//        DD  DSN=DXC.V2R4M1.DXCLMD2,DISP=SHR
//INPUT   DD  DSN=old_octm_backup_sequential_file,DISP=OLD
//OUTPUT  DD  DSN=new_octm_backup_sequential_file,
//            DISP=(NEW,CATLG),SPACE=(CYL,(50,50)),
//            DCB=(LRECL=12900,BLKSIZE=13000,RECFM=VB)
//PRINT   DD  SYSOUT=*,DCB=(LRECL=133,BLKSIZE=1330)
//LIST    DD  DSN=octm_list_sequential_file,
//            DISP=(NEW,CATLG),SPACE=(CYL,(50,50)),
//            DCB=(LRECL=80,BLKSIZE=8000,RECFM=FB)
//SYSIN   DD  *
//*
```

The following describes each input and output dataset (in the above DD statements) required by the offline program.

- INPUT dataset
  This is the input OCTM sequential file created by the online OCTM database backup function.
- OUTPUT dataset
  This is the output OCTM sequential file created by the offline program (which can be used as input to the online OCTM database restore function).
- PRINT dataset
  This is the report created by the offline program containing the names of the communication resources that have been modified or deleted.
- LIST dataset
  This is a sequential dataset containing an ALCS generation COMDEF macro for every communication resource on the OCTM database. This dataset is produced when PARM=LIST is specified on the JCL EXEC statement.
- SYSIN dataset
  This contains the new ranges of CRI addresses and ordinal numbers when this offline program is used for validating and implementing new CRI and ordinal ranges. Specify either CRIRANGE or ORDRANGE, or specify both. They should be specified on separate input lines, in the format:

  ```
  CRIRANGE=(nnnnnn,nnnnnn,nnnnnn,nnnnnn,.....)
  CRIRANGE=(nnnnnn,nnnnnn,nnnnnn,nnnnnn,.....)
  ORDRANGE=(nnnnn,nnnnn,nnnnn,nnnnn,....)
  ORDRANGE=(nnnnn,nnnnn,....)
  ```

This offline program terminates with register R15 set to:

R15 = 00
OK

R15 = 08
Invalid CRIRANGE and/or invalid ORDRANGE

R15 = 12
Invalid parameter on EXEC statement

R15 = 16
Unable to load or link DXCUTMOL

For a summary of the functions provided by this offline program, see *ALCS Installation and Customization*.

Chapter 5. Recoup

This chapter briefly describes the ALCS long-term pool space recovery utility (Recoup). Details of:

- The GROUP and INDEX macros
- Descriptor programs
- User-written subroutines and programs

are available in *ALCS Installation and Customization*.

Reasons for using Recoup

Recoup identifies long-term pool file records that are not in use. To do this, it first identifies all the long-term records that are in use. It then indicates the remaining long-term records as being available. Recoup is not used with short-term pool file records (these are returned to the system automatically).

Records from the long-term pools can contain data for long periods (possibly several months or longer). When an application program releases a long-term pool file record, ALCS records the release in control fields in the pool record itself. The record does not become available for reuse immediately. The pool file directory record (PFDR) is not updated.

Recoup makes released long-term pool file records available for reuse. If Recoup is not run, these pool file records never become available, and eventually there are no long-term pool file records available at all. If this happens then your application programs will probably be unusable.

If you have loaded a new DASD configuration table then some of the changes are not brought into effect until Recoup is run. See “ZDASD -- DASD data set functions” on page 123 for an explanation of the ZDASD command.

See *ALCS Concepts and Facilities* for a full explanation of Recoup.

Frequency of running Recoup

The ALCS operator runs Recoup by using the ZRECP command, as described in “ZRECP -- Control Recoup” on page 275.

The optimum frequency for running Recoup depends on the installation, and must be determined by experience. The principal factors that determine this optimum frequency are:

- The number of long-term pool file records in the real-time database.
- The rate at which long-term pool file records are dispensed.
- The number of errors detected each time Recoup is run.

Until experience indicates the contrary, you should run Recoup every two days.

Be aware that ALCS continually monitors the long-term pool dispense rate. ALCS uses this rate to predict when there will be no more records available. If the dispense rate exceeds the threshold rate for the pool file, or the time until pool is exhausted drops below the threshold time for the pool file, ALCS sends an **Attention** message to RO CRAS. If you see this message you should consider running Recoup immediately. You can set the long-term pool dispense threshold values to suit your installation using the ZP00L command, as described in “ZPOOL -- Alter/display pool file status” on page 266.

Running ALCS for the first time

When ALCS is initiated for the first time, the long-term pool file directory records are either all available or all not available. This depends on the LTPOOL operand of the SCTGEN generation macro. This is described in *ALCS Installation and Customization*. If the records are set to INUSE you must run Recoup before altering the system from IDLE state for the first time.

Errors detected by Recoup

Recoup checks all chains of long-term pool file records and builds new pool directories.

There are four possible situations, two of which are errors and are highlighted in the following table:

Table 6. Possible errors in chains		
	The record has been released (or never used)	**The record has not been released**
Recoup finds the record in a chain	**Recoup finds this error (Note 1)**	Record in use
Recoup does not find the record in a chain	Record available for reuse	**ALCS dispense facility finds this error (Note 2)**

The two error conditions are described in the following notes:

Notes:

1. Recoup provides diagnostic information. ALCS does not dispense the record to an application.
2. This is an error condition (**lost address**) that ALCS detects when it next attempts to dispense the record. When it is doing this, ALCS provides diagnostic information and flags the record. It does not, however, dispense the record until at least after the next Recoup run.

Database statistics from Recoup

The operator can request Recoup to write statistical information to the database analysis sequential file for subsequent offline analysis. (See “ZRECP -- Control Recoup” on page 275).

This information can be used to monitor the pool usage for diagnostic and planning purposes.

Database analysis file

For each pool file record that Recoup reads, Recoup writes a 48-byte item to the database analysis sequential file. The sequential file comprises size L2 records. Each record has 24 bytes of header information followed by a maximum of 21 items of 48 bytes each. The `BS0AI` DSECT defines the record format. The count of items in use is held in a fullword field `BS0ITM`.

Table 7 on page 53 shows the format of each 48-byte item. See *ALCS Application Programming Reference - Assembler*, which describes the use of the `BS0AI` DSECT macro.

Table 7. Recoup - statistical information item format		
Label	**Length in bytes**	**Description**
`BS0REC`	2	Expected record ID
	1	Expected RCC
	1	Reserved
	4	Expected file address
`BS0DSC`	4	File address of refer-from record
`BS0ACT`	2	Actual record ID (see note)
	1	Actual RCC (see note)

Table 7. Recoup - statistical information item format (continued)		
Label	**Length in bytes**	**Description**
BS0FLG	1	Switch byte. Use the following symbols to test the switches in this field: BS0FLGD: This record read previously BS0FLGH: I/O error BS0FLGI: Record ID error BS0FLGR: RCC error BS0FLGA: Invalid file address BS0FLGC: Control error
BS0PRI	8	Prime group name
BS0GRP	8	Group name
BS0DSP	4	Dispense time (high-order fullword of TOD clock)
	4	Name of dispensing program
BS0FIL	4	File time (high-order fullword of TOD clock)
	4	Name of filing program
If Recoup has been unable to read a record (because of I/O error, invalid file address, or invalid control information), then this field contains the record ID and RCC of the refer-from record.		

Chapter 6. ALCS command reference

ALCS provides commands that allow:

- The operator and system programmer to monitor and control the operation of ALCS
- The application programmer to test programs

This chapter describes each of these commands.

Explanation of message formats

The format of ALCS messages is:

DXC
IBM product code for ALCS

nnnn
A unique decimal number identifying the message

s
Severity code

CMD
ALCS subcomponent code

i
The ALCS system identifier. This is one alphanumeric character specified by the ALCS generation.

hh.mm.ss
Time stamp

aaaa
Last four characters of the command

text
Text of the message

Conventions used in this book

In this version of ALCS all the responses to commands have been numbered. The numbers are in the range DXC8000 through DXC8999. All the error responses are fully documented in *ALCS Messages and Codes*.

Where the normal response is simple, the message is given in full in *ALCS Messages and Codes*. Where the normal response is complex, for example the normal responses to the command ZDCOM, all the possible options are listed in this book with explanatory text. For example see “ZDCOM -- Display communication resource information” on page 136.

Decimal numbers in ALCS messages

To make large decimal numbers easy to read, ALCS messages use a convention for grouping digits. Numbers up to and including 9999 are displayed without separation into groups, for example:

7
234
1993

Numbers greater than 9999 are displayed with a space separator between groups of three digits, for example:

23 456
2 317 484

Representation of dashes in ALCS messages

In accordance with established typographical conventions ALCS messages display the dash character (-) on equipment that cannot display this character as two hyphens (--).

Online help

Online help for the operator commands is available. To obtain an index of available help enter:

```
ZHELP Z* INDEX or
ZHELP INDEX Z*
```

The index shows which topics have help available. Second level help is also available for some sub-topics.

You can also access help directly if you know that help is available for that topic. For example both of the following entries display the help information for the ZSTAT command:

```
ZHELP ZSTAT
ZSTAT HELP
```

```
DXC8311I CMD M 01.16.03 HELP ZSTAT Display ALCS system load statistics
Use ZSTAT to display or print ALCS system load statistics
Format:  ZSTAT p

Where:   p    Optional - Single character p to print the statistics
              Default is display the characteristics

Related help topics:

ZHELP ZDECB
ZHELP ZDPFC
```

If you want to see some of the related help, for example for ZDECB, you need only overtype the Z of the ZHELP ZDECB and the help information on ZDECB is displayed when you press enter.

See “ZHELP -- Command help facility” on page 202 for a full explanation of how to use the ZHELP command.

Authorization

Some operator commands are restricted to certain types of input terminal, to certain system states, or to both. For example, ZASYS is accepted only from the Prime CRAS terminal or a terminal with Prime CRAS authority, and ZDATA LOAD is accepted only in IDLE state.

When ALCS receives a command, it checks these restrictions. When it detects an error, it sends one of the following error messages to the originating terminal:

```
DXC8004I CMD i hh.mm.ss xxxx
Not authorized to request this function -- Prime CRAS only
DXC8019I CMD i hh.mm.ss xxxx  Prime CRAS or AT1 -- AT16 only
DXC8005I CMD i hh.mm.ss xxxx  CRAS only
DXC8003I CMD i hh.mm.ss xxxx  Wrong system state
```

where *xxxx* are the last 4 characters of the command.

Validity checking

If the command contains a format error, you may receive one of the following error responses:

```
DXC8001I CMD hh.mm.ss xxxx  Unable -- Invalid command format
DXC8002I CMD hh.mm.ss xxxx  Unable -- Unknown command
DXC8012I CMD hh.mm.ss xxxx  Unable -- Invalid keyword parameter
DXC8013I CMD hh.mm.ss xxxx  Unable -- Invalid positional parameter
DXC8009I CMD hh.mm.ss xxxx  Unable -- Keyword invalid or omitted
DXC8010I CMD hh.mm.ss xxxx  Unable -- Keyword too long
```

```
DXC8008I CMD hh.mm.ss xxxx  Unable -- Parameter too long
DXC8031I CMD hh.mm.ss xxxx  Unable -- Too many parameters
DXC8007I CMD hh.mm.ss xxxx  Unable -- Unmatched parentheses
DXC8011I CMD hh.mm.ss xxxx  Unable -- Unmatched quote
```

where *xxxx* are the last 4 characters of the command.

Other error responses are listed in *ALCS Messages and Codes*.

Destination of responses

The response to an operator command is sent either to the originating terminal, or to a printer, as follows:

1. When the command produces a single response, ALCS sends it to the originating terminal.
2. When the command produces more than one response, ALCS sends the first response to the originating terminal followed by a message:

   ```
   Progress messages on printer -- CRN-crn  CRI-cri
   ```

   Where:

   crn
   Communications resource name (CRN) of the printer where the command sends subsequent (progress) responses.

   cri
   Communications resource identifier (CRI) of the printer to which the command sends subsequent (progress) responses.

   If the originating terminal has an associated printer that is active and usable then this is the CRN and CRI of the associated printer. If not then this is the CRN and CRI of RO CRAS.
3. Some responses are sent out through the ALCS scrolling function, so the scrolling header appears on the screen. This is described in "Scrolling displays" on page 58.
4. ALCS generates some commands itself to perform system functions. The responses to these commands are sent to the RO CRAS. These commands include:

   ZACOM - to perform CRAS fallback if the RO CRAS is lost

Printer shadowing

Printer shadowing is an ALCS facility that allows up to 16 printers to receive a copy of a message sent to a particular printer. The operator can control printer shadowing using the ZACOM command.

Printer redirection

You can also use the ZACOM command to send a message to a different printer than the associated printer. See "Control ALCS terminal message queues" on page 79 for details of both printer shadowing and printer redirection.

3270 Display layout

Figure 9 on page 58 shows an example of a 3270 display being used to enter ALCS operator commands.

```
zdkey pf3
DXC8667I CMD M 01.43.15 DKEY
* in column 1 indicates installation default
I in column 1 indicates ZAKEY inhibited
* PF03   OA
zakey pf3 zstat
DXC8082I CMD M 01.43.25 AKEY  OK
zdkey pf3
DXC8667I CMD M 01.43.31 DKEY
* in column 1 indicates installation default
I in column 1 indicates ZAKEY inhibited
* PF03 % ZSTAT



        ALCS  V2 Production System                  date
```

Figure 9. Display being used for operator commands.

Note: *date* is the ALCS local date in the format defined by the SCTGEN DATEFORM generation parameter. If the DATEFORM parameter defines a date format that is too long to fit on the bottom line of the 3270 display then *date* is in the format *dd.mm.yy*.

Explanation of the display layout

The main part of the display is available for entering commands and receiving replies from ALCS. At the bottom of the display is a line showing the name of the current ALCS system and the current date. (The exact text that appears at the bottom of the screen is defined in the system configuration table.)

In Figure 9 on page 58, the operator has entered three commands in sequence:

```
zdkey pf3
zakey pf3 zstat
zdkey pf3
```

The operator has entered these commands in lower case text but they could equally well be entered in upper case.

ALCS has replied to each command. The replies can be in upper or lower case. A reply can occupy one or more lines but the first line always starts in the same way: DXC*nnnns* CMD *i hh.mm.ss xxxx*

hh.mm.ss is the current ALCS local time and *xxxx* are the last 4 characters of the command.

Note: You can re-enter a command that you have previously entered by moving the cursor up to the earlier command. You can overtype parts of the command, for example to change the parameters. ALCS replies to the command on the line below the cursor.

Scrolling displays

The responses to some commands might be too large to fit on to a single screen. For example, Figure 10 on page 59 shows a typical ZDSEQ response (see “ZDSEQ -- Display sequential file status” on page 194 for a description of the fields).

You can use the ZSCRL command to scroll to different parts of the output file which contains the response.

You can scroll the display up, down, left, or right. Alternatively, you can request the screen display to start at a particular line number or column number in the output file.

The top two lines of the screen are called the header. You can use the ZSCRL command to display or suppress the header.

See “ZSCRL -- Scroll information on the screen” on page 289 for more details.

```
              Lines      1 to     16 of     26 Columns  1 to 62 of   62
              Active:     5  4  3  2 *1*     More:           down
DXC8233I CMD M 04.19.26 DSEQ
File Device   Status  Type Volume Data-set-name
LOG  *DUMMY*  OPEN    LOG         XAN.XA2MNT.SEQF.LOG.M0000000
DIA  3380     OPEN    DIAG IASC03 XAN.XA2MNT.SEQF.DIA.M0009000
TPF  3380     CLOSED  OUT         XAN.XA2MNT.SEQF.TPF
RTA  *DUMMY*  OPEN    RT          XAN.XA2MNT.SEQF.RTA.M0000000
DCL  USES RTA
PIJ  USES RTA
RTL  USES RTA
TDR  TAPE     CLOSED  IN          DBF.TAPE
TDD  3380     CLOSED  IN          XAN.DBF.TAPE
PID  3380     CLOSED  IN          XAN.XA2MNT.PILOT.D
PIM  3380     CLOSED  IN          XAN.XA2MNT.PILOT.M
PIX  3380     CLOSED  IN          XAN.XA2MNT.PILOT.X
OUT  3380     CLOSED  OUT         XAN.XA2MNT.SEQF.OUT
TTT  *DUMMY*  CLOSED  OUT
TUT  3380     CLOSED  IN          XAN.XA2MNT.SEQF.TUT
VPH  *DUMMY*  CLOSED  OUT
More...
```

Figure 10. Sample ZDSEQ display

The first two lines of the display provide the following information:

Lines
Shows the line numbers of the first and the last lines displayed and the total number of lines of data in the file. For example Figure 10 on page 59 shows the first 16 lines of a total of 26 lines.

Columns
Shows the numbers of the first and the last columns displayed and the column number of the end of the longest line in the file. For example Figure 10 on page 59 shows all 62 columns of a response that is 62 columns wide.

Active
Figure 10 on page 59 shows a single file, created by one operator command that resulted in output that was too large to fit on to one screen. ALCS allows up to five different files of such output. ALCS numbers these output files from one to five. The file currently displayed on the screen is the ***active file***. This is shown as *1* in Figure 10 on page 59. You can switch between output files with the ZSCRL command.

More:
Indicates the directions that have displayable data beyond the screen boundaries. For example Figure 10 on page 59 shows that you can display more lines by scrolling down.

The last line of an active file display always ends with either:

`More...`, indicating that more data exists downward, or
`Bottom`, indicating that the end of the data has been reached.

NetView display

Figure 11 on page 60 shows ALCS messages which have been recorded on the NetView status monitor (Statmon) log. It is possible for the Statmon log to display messages from other sources, for example MVS system messages, on the same screen.

```
 STATMON.BROWSE     ACTP  NETWORK LOG FOR 09/16/93 (91249) COLS 037 114 15:13
 HOST: HOST31              *1*     *2*     *3*     *4*        SCROLL ==> 0001
  ---4----+----5----+----6----+----7----+----8----+----9----+---10----+---11---
ALCSCMD PYEZC01,IALC02,ZDCOM N=ROC,D,P
DXC8045I CMD V 17.04.03 DCOM
Output on printer -- CRN-'PYEP0743'  CRI-'020006'
DXC2999I SYS V  ****** COPIED FROM PRINTER CRI - 020006 CRN - PYEP0743
DXC2999I SYS V  DXC8900I CMD V  17:04.03 DCOM
DXC2999I SYS V  Resource CRN      CRI    Ordinal  Routing  Status   CRAS  f/b
DXC2999I SYS V  3270-PRT PYEP0743 020006 00000006 RES0     ACTIVE   ROC    F
DXC2999I SYS V  Associated resource CRN - None
DXC2999I SYS V  Messages on queue       - 0
DXC2999I SYS V  Printer buffer size     - 1920
DXC2999I SYS V  DBCS support            - No
DXC2999I SYS V  SNA between brackets    - No
DXC2999I SYS V  Resource is unusable    - No
DXC2999I SYS V  System sends allowed    - Yes
DXC2999I SYS V  Message shadowing       - Yes
DXC2999I SYS V  Shadow CRN(s)           - PYEPJD7  PYEPPI1
DXC2999I SYS V  Message re-direction    - No
DXC2999I SYS V  Re-direction CRN        - None
 CMD==>
 1=HELP 2=END 3=RET 4=TOP 5=BOT 6=ROLL 7=BCK 8=FWD 9=RPTFND 10=LFT 11=RGT 12=AL
```

Figure 11. Sample NetView display

Confirmation of commands

Some commands could have very serious effects if used incorrectly. For these commands ALCS prompts you to confirm your request. You do this by entering the complete command again in the same form as before. For example:

User	zatim add 0200
System	DXC8515I CMD M 22.20.15 CONF Local date change Reenter within 30 seconds to confirm
User	zatim add 0200
System	DXC8648I CMD M 22.20.17 ATIM ...(normal ZATIM response)

If you do not reenter the command in the same form within 30 seconds, the original request is ignored.

Some commands only require confirmation when certain parameters are specified (this is the case with ZATIM illustrated above). Other commands (such as ZPURG), require confirmation for every request.

Commands, or command and parameter combinations that require confirmation, are identified in this chapter in the command description.

Reading syntax diagrams

This book uses "railroad" syntax diagrams, which are designed to be easy to read and interpret without ambiguity.

Read the syntax diagram from left to right, and top to bottom, following the path of the line:

>>— indicates the beginning of a command.
—> indicates that the command syntax continues on the next line.
>— indicates that a command is continued from the previous line.
—>< indicates the end of a command.

Parameters

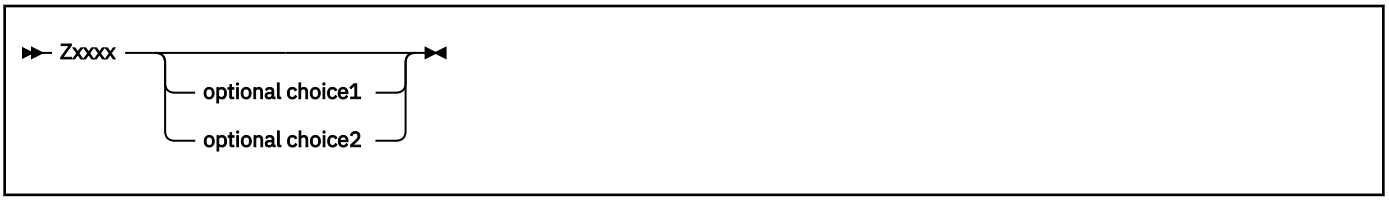
Within a syntax diagram, parameter options are shown as follows:

Required items appear on the horizontal line:

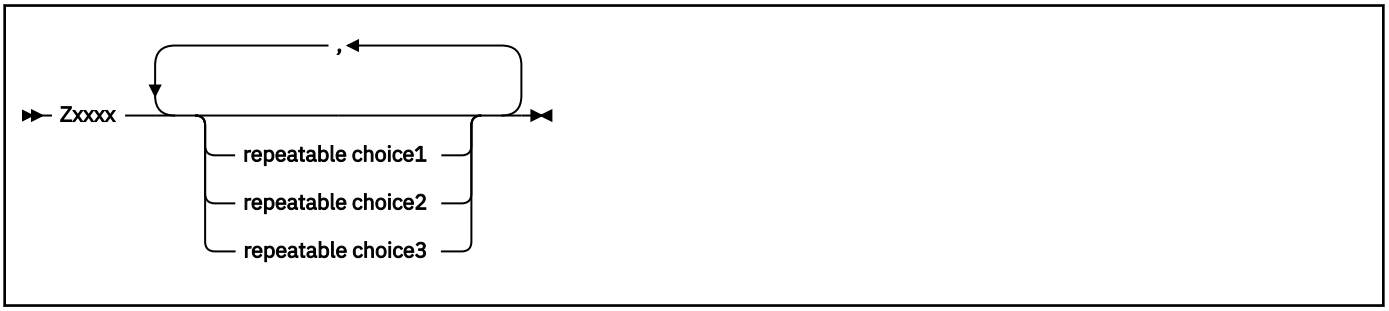
Optional items appear below the main path:

If you must choose from two or more items, they appear in a stack:

If the choices are optional they appear in a stack below the main path:



An arrow returning to the left above the line indicates that an item can be repeated one or more times. If the arrow contains a comma, you must enter each item separated by a comma:



A default parameter appears above the main path:

Unless stated otherwise, keywords and parameters must be entered in the order that they appear along the path:

In some cases a command requires information (in this example a comment) on the next line of the screen. This is indicated by the (NL) on the syntax diagram:

Each syntax diagram starts with the name of the command, followed by one or more parameters. These parameters can contain keywords and variables as follows:

Keywords
Words that ALCS recognizes, for example LOAD and DUMP.

Variables
Indicate that you must enter a value. Variables are shown in *italics*, for example *module*.

Keywords with variables
Some keywords require you to enter a value, for example, SEQ=*seq*.

Keywords containing variables
These occur in a very few commands. For example, AT*nnn* requires you to enter a number, for example AT9 or AT123. The command descriptions explain any cases that are not immediately clear to you.

How to enter an ALCS command

The rules for typing ALCS commands are quite flexible:

Case does not matter

You can enter commands and parameters in upper case or lower case or any mixture of the two. ALCS treats the following as exactly equivalent:

```
ZDPRG FRED
zdprg fred
ZDPRG fred
Zdprg Fred
zDPRG fREd
```

Keyword abbreviations

You can use abbreviated versions of most keywords. Often these abbreviations are a single letter. Keywords appear in the syntax diagrams in capitals if they cannot be abbreviated, or in mixed case with the capitals showing the acceptable short form.

For example, the keywords Conv and sTop mean that you can enter any of the following:

```
ztrac conv,stop
ztrac c,t
ztrac c,stop
```

Other abbreviations (such as ZTRAC CON) are ***not*** accepted.

Separation of keywords and variables

All commands start with a keyword (Zxxxx) which is the name of the command. You must follow this with one or more spaces. You can separate the other items of the command (variables and other keywords) using spaces, or a comma.

The syntax diagrams show the comma as a separator, but you can use a space instead.

Equals sign

The syntax diagrams show an equals sign (=) in certain commands. If your terminal does not have this character, or if you prefer, you can use a hyphen (-) instead.

ALCS ***command responses*** use the hyphen in most cases where an equals sign would be equally appropriate.

Parentheses

The syntax diagrams show certain values enclosed in parentheses. ALCS accepts either parentheses or slashes (but not a parenthesis paired with a slash).

Symbolic CRAS CRN

Wherever you need to specify a communication resource name (CRN), you can do so either with the name that was specified during ALCS generation or, if the terminal has one, with a symbolic CRAS CRN such as PRC, ROC, AT*nnn*, or AP*nnn*.

ALCS parameter descriptions

In the descriptions of the parameters in ALCS commands (these descriptions appear after the syntax diagrams) the following conventions are used:

Keywords are shown either as Keyword or as Keyword. Variables are shown as *variable*. Default values are underlined (shown respectively as Keyword and *variable*).

Symbols describing parameter syntax

The following symbols have special significance in describing the parameters. You must not enter them in the command.

Braces
{ }

Brackets
[ ]

Ellipsis
...

OR symbol
|

Braces

Braces indicate a series of options, where you must choose one:

```
{A|B|C}
```

means that you must select one of A, , B, or C.

Brackets

Brackets indicate a series of options, where you can optionally choose one:

```
[A|B|C]
```

means that you can select one of A, , B, or C, or omit the parameter altogether.

Ellipses

Ellipses (...) indicates that an item or group can be repeated more than once in succession.

Symbols used in commands

The following symbols can form part of the ALCS command:

Asterisk
*

Comma
,

Hyphen
-

Equal sign
=

Parentheses
( )

Slashes
/ /

Period
.

See also "How to enter an ALCS command" on page 62.

Terminals with restricted character sets

When a response includes a character that is not in the character set of the terminal, the response does not appear exactly as shown. For example, on some terminals responses are always in uppercase. See "How to enter an ALCS command" on page 62 for information on acceptable alternatives which these terminals can accept and display.

Overview of ALCS commands

This section gives an overview of operator commands grouped by function. The rest of the chapter describes each of the commands in alphabetical order.

The commands are grouped as follows:

- General commands
- Screen commands
- DASD commands
- Sequential file commands
- Communication resource commands
- Test and trace commands

The following tables list the commands in each group.

General commands

Table 8. General commands	
ZAACV/ZDACV	Alter/display activity control variables
ZACOR/ZDCOR	Alter/display ALCS system storage
ZAPRG/ZDPRG	Alter/display program
ZASER/ZDSER	Alter/display system error options
ZASYS/ZDSYS	Alter/display system state
ZATIM/ZDTIM	Alter/display time
ZCSQL	Control the connection between ALCS and DB2
ZDCLR	Control data collection
ZDECB	Display information about long-lived ECBs
ZDUMP	Request manual dump

Table 8. General commands (continued)	
ZHELP	Command help facility
ZPCTL	Load/unload/display application load module
ZPURG	Purge an entry
ZSTAT	Display system load

Screen commands

Table 9. Screen commands	
ZAKEY/ZDKEY	Alter/display terminal function key settings
ZCMSP	Alter/display 3270 screen maps
ZRETR	Recall previously issued command
ZSCRL	Scroll information on the screen
ZSNDU	Send or receive an unsolicited message

DASD commands

Table 10. DASD commands	
ZAFIL/ZDFIL	Alter/display DASD record
ZDASD	DASD database functions
ZDATA	Load/dump data from/to sequential file
ZDPDU	Display status of long term pool directory
ZGAFA	Get available pool file address
ZPOOL	Alter/display pool file counts and dispense options
ZRECP	Start, stop, and control Recoup
ZRELO	Dump/restore records to/from sequential file
ZRSTR	Restore logged records

Sequential file commands

Table 11. Sequential file commands	
ZASEQ/ZDSEQ	Alter/display sequential file configuration table
ZCSEQ	Close general sequential file
ZSSEQ	Switch sequential file

Communication commands

Table 12. Communication commands	
ZACOM	Load a communication generation module Assign or transfer CRAS status Alter communication resource information Control a printer terminal Set or clear BATAP variables for AX.25 and MATIP Type B message processing Control an LU 6.1 link Control an SLC link or channel, or all links
ZCMQI	Control the communication between ALCS and WebSphere MQ for z/OS
ZCTCP	Control the connection between ALCS and other devices using TCP/IP
ZCWAS	Control and display the connection between ALCS and WebSphere Application Server for z/OS
ZDCOM	Display communication resource information Display BATAP variables for AX.25 and MATIP Type B message processing Display the status of an SLC channel Display communication system information Display message queues
ZLKST	Display statistics for an SLC link or channel
ZLOGF	Log off a VTAM controlled terminal from ALCS
ZLOGN	Log on to ALCS with a different user ID
ZMAIL	Use the ALCS e-mail facility
ZOCTM	Control online communication table maintenance
ZRCRS	Send a message to a CRAS terminal
ZROUT	Alter/display input routing for this terminal

Test and trace commands

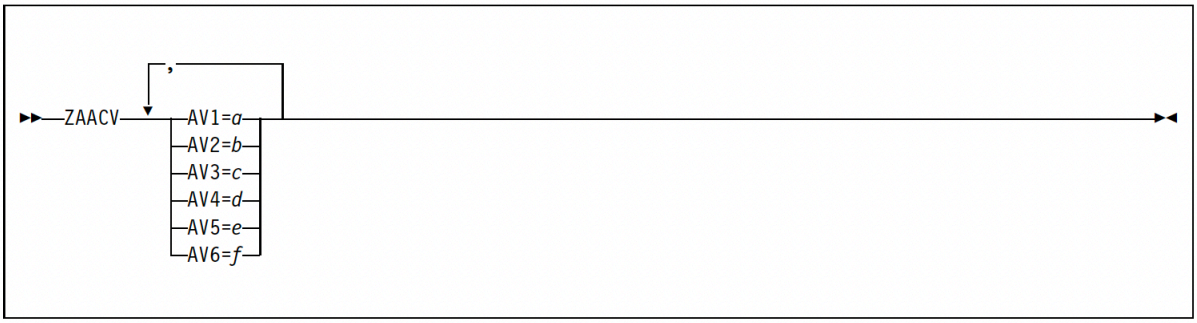
Table 13. Test and trace commands	
ZDRIV	Invoke application program
ZLTST	Invoke an SLC link test
ZLKTR	Control the SLC link trace facility
ZTEST	Control the ALCS test facility
ZTRAC	Control the ALCS trace facility

ZAACV -- Alter activity control variables

Use the ZAACV command -- ***from a Prime CRAS authorized terminal only*** -- to alter parameters that control the creation of new entries. This command can help optimize the performance of ALCS. When ALCS restarts, the parameters revert to the values specified at generation time.

Attention

Altering these parameters can cause system problems. Consult your system programmer before using this command. The format of the command is:



Where:

AV1=*a*
The ALCS online monitor is to service the input list only if the number of active entries is less than, or equal to, *a*. The minimum value for this variable is 1. The maximum value is one less than the maximum number of active entries permitted in the system.

AV2=*b*
The ALCS online monitor is to create a new entry only if the existing number of entries is less than, or equal to, *b*.

If this limit is exceeded, ALCS queues any entry that issues a create-type macro. Eventually, the number of entries reduces to *b* and the entry can execute the create-type macro. The minimum value for this variable is 1. The maximum value is one less than the maximum number of active entries permitted.

AV3=*c*
The ALCS online monitor is to service the input list, create an entry, or issue ACF/VTAM RECEIVEs only if the number of available storage units is greater than, or equal to, *c*. The minimum value for this variable is 1. The maximum value is one less than the number of storage units defined in the ALCS generation.

AV4=*d*
The ALCS online monitor is to service the input list only if the number of available IOCBs is greater than, or equal to, *d*. The minimum value for this variable is 1. The maximum value is one less than the number of IOCBs defined in the ALCS generation.

AV5=*e*
The maximum number of entries that Recoup can create. The minimum value is 1 and the maximum is one fewer than the maximum number of active entries allowed in the system.

AV6=*f*
The percentage of free entries to be made available to batch type programs (Recoup, file maintenance, and so on) that use the LODIC ECBCREATE macro:

- To regulate the number of entries that they create
- To stop creating new entries when the number of entries already existing reaches *f* percent of control variable 1 or 2 (whichever is smaller).

See the description of the LODIC macro in *ALCS Application Programming Reference - Assembler*, for details. The minimum value for this variable is 1, the maximum is 99.

Normal response

```
DXC8586I CMD i hh.mm.ss  AACV
Input max active entries ....(AV1) ........a
Create max total entries ....(AV2) ........b
Input min free storage units (AV3) ........c
Input min free I/O blocks ...(AV4) ........d
Recoup max creatable ECBs ...(AV5) ........e
Entry create high percentage (AV6) ........f
```

Where *a,b,c,d,e*, and *f* are the new values.

ZACOM -- Alter communication resource information

You can use the ZACOM command to :

- Load a communication configuration load module or load list (“Load a communication configuration load module or load list” on page 69).
- Assign or transfer CRAS status (“Assign or transfer CRAS status” on page 71).
- Alter information about one or more communication resources (“Alter information about communication resources” on page 73).
- Set or clear BATAP variables for AX.25 and MATIP Type B messages (“Set or clear BATAP variables for AX.25 and MATIP Type B message processing” on page 77).
- Control an LU 6.1 link (“Control ALCS LU 6.1 message queues” on page 79).
- Control an ALCS printer (“Control ALCS terminal message queues” on page 79).
- Control an SLC link or channel (“Control an SLC link or channel” on page 81).

Each of these operations is described in the following sections.

Load a communication configuration load module or load list

Use the ZACOM command -- ***from a Prime CRAS authorized terminal only*** -- to manage the loading of communication configuration load modules. If your ALCS system is using the communication configuration data set (CDS2), you can also load and backout communication configuration load lists (these contain the names of communication configuration load modules). Additional functions are performed during the loading of communication configuration load modules when your system is using CDS2.

- **Loading a communication configuration load module**

  Load a communication configuration load module that will add, delete, and replace resources in the existing communication configuration table according to actions requested in the communication configuration load module. Ensure that any communication resources that you want to update or delete are made inactive before you load the communication configuration load module.

  Note: Before loading a communication configuration load module, run the ALCS communication report file generator to validate the communication configuration definitions. Check, for example, that there are no duplicate CRI addresses or ordinal numbers. “Using ISPF panels to run the communication report file generator” on page 26 describes how to run the communication report file generator.

  If your system is using CDS2, additional functions are provided, and changes made to the communication configuration table are preserved across an ALCS system restart. The `ZACOM LOAD` command adds the name of the communication configuration load module to the communication configuration load list on CDS2. After a communication configuration load module has been loaded, if you are satisfied that it has not impacted the availability or performance of the communications network, use the `ZACOM CONFIRM` command to confirm that load. This will ensure that the load module is preserved across future ALCS restarts. If you are not satisfied with the communications network changes introduced by the communication configuration load module, use the `ZACOM BACKOUT` command to remove it on the next ALCS restart.

- **Loading a communication configuration load list**

  Load a communication configuration load list on CDS2 using the `ZACOM LOAD LIST` command (a load list contains the names of the communication configuration load modules that must be loaded on the next restart of the ALCS system). ALCS does not use this list until you confirm it and restart the system.

  After a new communication configuration load list has been successfully loaded, if you require ALCS to use that list for loading communication configuration load modules on the next ALCS restart, use the `ZACOM CONFIRM LIST` command to confirm the load list.

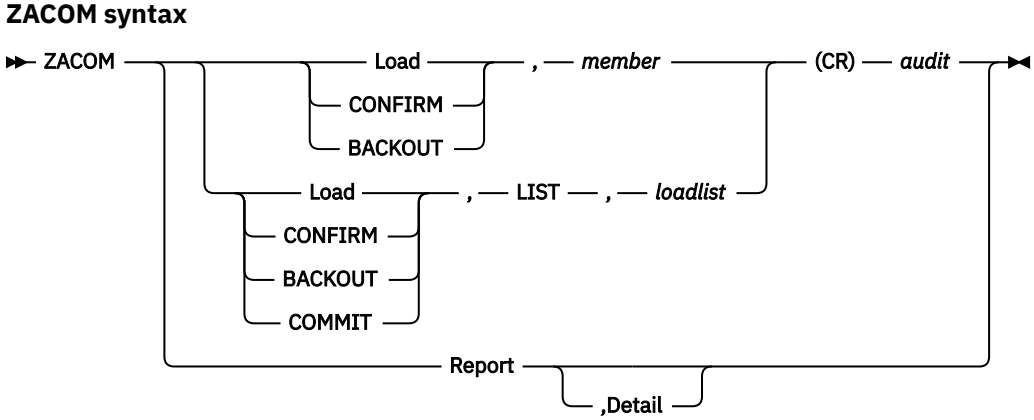
After the next restart of the ALCS system, if you are not satisfied with the communication network changes that were introduced by the new communication configuration load list, use the ZACOM BACKOUT LIST command to remove that load list (and fallback to the previous load list). To activate the previous load list, restart the ALCS system.

If you are satisfied with the communication network changes that were introduced by the new communication configuration load list, use the ZACOM COMMIT LIST command to prevent a backout of that load list.

ZACOM Command Format

The format of the command is:

ZACOM syntax



Where:

Load,*member*
Load the communication configuration load module *member*. Use it to update the existing ALCS communication configuration table. This update will not be preserved across an ALCS restart. If CDS2 is in use on the ALCS system, the name of the load module *member* is inserted in the CDS2 load list.

CONFIRM,*member*
Confirm that the communication configuration load module *member*, previously loaded, is safe to preserve across an ALCS restart. Use the confirm function only when CDS2 is in use on the ALCS system. The load module *member* is marked as confirmed in the CDS2 load list, and if a restart of the ALCS system occurs, the load module *member* will be used to update the communication configuration table during the restart.

Note: You must confirm load modules in the same order you load them.

BACKOUT,*member*
Backout (stop using) the communication configuration load module *member*, previously loaded. The backout function, which can be used only when CDS2 is in use on the ALCS system, marks the load module *member* as backed out in the CDS2 load list. The backout of the load module from the communication configuration table does not occur until after the next ALCS restart.

member
The name of the communication configuration load module (up to 8 characters); this must be unique.

Load,LIST,*loadlist*
Load the communication configuration load list *loadlist* onto CDS2. ALCS will not use this load list until it has been confirmed. You cannot load a new communication configuration load list until you either commit or backout a previous load list.

CONFIRM,LIST,*loadlist*
Confirm that the communication configuration load list *loadlist*, previously loaded onto CDS2, is safe to use at the next restart of the ALCS system.

BACKOUT,LIST,*loadlist*
Backout the communication configuration load list *loadlist* on CDS2. This load list, which must have been previously loaded, and may have been previously confirmed, will not be used during the next restart of the ALCS system. The previous load list that was loaded onto CDS2 will now be used during the next restart of the ALCS system. Alternatively, a replacement load list could now be loaded for usage during the next ALCS restart.

COMMIT,LIST,*loadlist*
Commit the communication configuration load list *loadlist* on CDS2. The load list must have been previously confirmed. When a load list has been committed, it can not be backed out.

loadlist
The name of the communication configuration load list load module (up to 8 characters); this must be unique.

audit
Audit-trail information. Up to 80 characters of text.

Report
Display a status report of the communication configuration load list(s) plus the loaded, confirmed, or committed communication configuration load modules on CDS2.

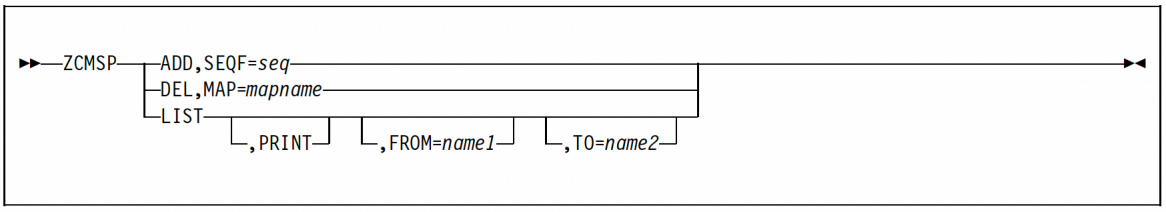
Report,Detail
Display a status report of the communication configuration load list(s) plus all communication configuration load modules on CDS2. This includes backed out (unloaded) and never loaded modules.

Assign or transfer CRAS status

Use the ZACOM command to transfer or remove CRAS status, or to assign or remove CRAS authority to or from a resource. See Table 14 on page 73 and Table 15 on page 73 for the level of CRAS authorization that can perform each function.

Changes made in this way are not preserved across a system restart.

The format of the command is:



Where:

PRC=*crn*
Transfer Prime CRAS status from one IBM 3270 display or NetView display to another, where *crn* is the CRN of the new Prime CRAS, which must already be an alternate CRAS AT1 to AT16. The old Prime CRAS acquires this alternate CRAS status. Specify *crn* as 1 through 8 alphanumeric characters.

If any other CRAS authorities exist for either resource, then these are removed before CRAS status is exchanged.

ROC=*crn*
Transfer RO CRAS status from one IBM 3270 printer or NetView printer to another, where *crn* is the CRN of the new RO CRAS, which must already be an alternate CRAS AT1 to AT16 or an alternate CRAS printer AP1 to AP16. The old RO CRAS acquires this alternate CRAS or alternate CRAS printer status. Specify *crn* as 1 through 8 alphanumeric characters.

AT*nnn*=*crn*
Give a display or printer alternate CRAS status, where *nnn* is the alternate CRAS number (a value 1 through 255) and *crn* is the CRN of the display or printer. Specify *crn* as 1 through 8 alphanumeric characters or * for the originating terminal. The specified resource must not already have alternate

CRAS printer status. If another display or printer is already defined as AT*nnn*, that display or printer acquires the CRAS status of the resource with CRN of *crn*.

If any other CRAS authorities exist for either resource, then these are removed before CRAS status is exchanged. AT1 to AT16 can be assigned to IBM 3270 terminals or NetView resources only.

AP*nnn*=*crn*
Give a printer alternate CRAS printer status, where *nnn* is the alternate CRAS printer number (1 to 255) and *crn* is the CRN of the printer. Specify *crn* as 1 through 8 alphanumeric characters or * for the originating terminal. The specified resource must not already have alternate CRAS status. If another printer is already defined as AP*nnn*, that printer loses its CRAS status. AP1 to AP16 can be assigned to IBM 3270 printers or NetView resources only.

NOCRAS=*crn*
Remove CRAS status from a display or printer, where *crn* is the CRN of the display or printer. Specify *crn* as 1 through 8 alphanumeric characters or * for the originating terminal.

crI=*cri*
The CRI of the resource to be assigned the specified CRAS authority (6 hexadecimal digits).

crN=*crn*
The CRN of the resource to be assigned the specified CRAS authority (1 through 8 alphanumeric characters or * for the originating terminal).

CrasAuth=*auth*
CRAS authority to be assigned to the specified resource. One of the following:

PRIME|PRC
Assign Prime CRAS authority to an IBM 3270 display or NetView display. The resource must not already have Prime CRAS status. There is no restriction on the number of terminals that can be assigned Prime CRAS authority.

AT*nnn*
Assign alternate CRAS authority to a display or printer, where *nnn* is the alternate CRAS number (a value 1 through 255). AT1 to AT16 authority can be assigned to IBM 3270 terminals or NetView resources only. The resource must not already have alternate CRAS printer status. There is no restriction on the number of terminals that can be assigned alternate CRAS authority.

NOCRAS|NONE
Remove CRAS authority from a resource. If the resource has CRAS status then this is not removed. (To remove CRAS status use the ZACOM **NOCRAS**=*crn* command.)

Restrictions

Assigning CRAS status

Prime CRAS status can only be assigned to a terminal which already has alternate CRAS status AT1 through AT16.

RO CRAS status can only be assigned to a terminal which already has alternate CRAS status AT1 through AT16 or alternate CRAS printer status AP1 through AP16.

Prime CRAS or RO CRAS status can not be removed from a terminal without assigning it to another terminal at the same time.

Assigning CRAS status to a terminal requires the appropriate SAF authorization for the user-ID currently logged on to that terminal. If there is no SAF decision then authorization to perform the function is granted. In any case, the requesting terminal must have the appropriate CRAS status or CRAS authority, as shown in Table 14 on page 73.

Table 14. Input terminal restrictions for assigning CRAS status			
Function	**Prime CRAS**	**AT1 to AT16**	**AT17 to AT255**
PRC	Yes	Yes	No
ROC	Yes	Yes	No
AT1 to AT16	Yes	Yes	No
AT17 to AT255	Yes	Yes	Yes
AP1 to AP16	Yes	Yes	No
AP17 to AP255	Yes	Yes	Yes
NOCRAS AT1 to AT16	Yes	Yes	No
NOCRAS AT17 to AT255	Yes	Yes	Yes

Assigning CRAS authority

Assigning CRAS authority to a terminal other than the originating terminal requires the appropriate SAF authorization for the user-ID logged on to that terminal. If there is no SAF decision then authorization to perform the function is granted. In any case, the originating terminal must have Prime CRAS status or Prime CRAS authority.

Assigning CRAS authority to the originating terminal requires the appropriate SAF authorization for the user-ID currently logged on to that terminal. If there is no SAF decision then the requesting terminal must have the appropriate CRAS status or CRAS authority, as shown in Table 15 on page 73.

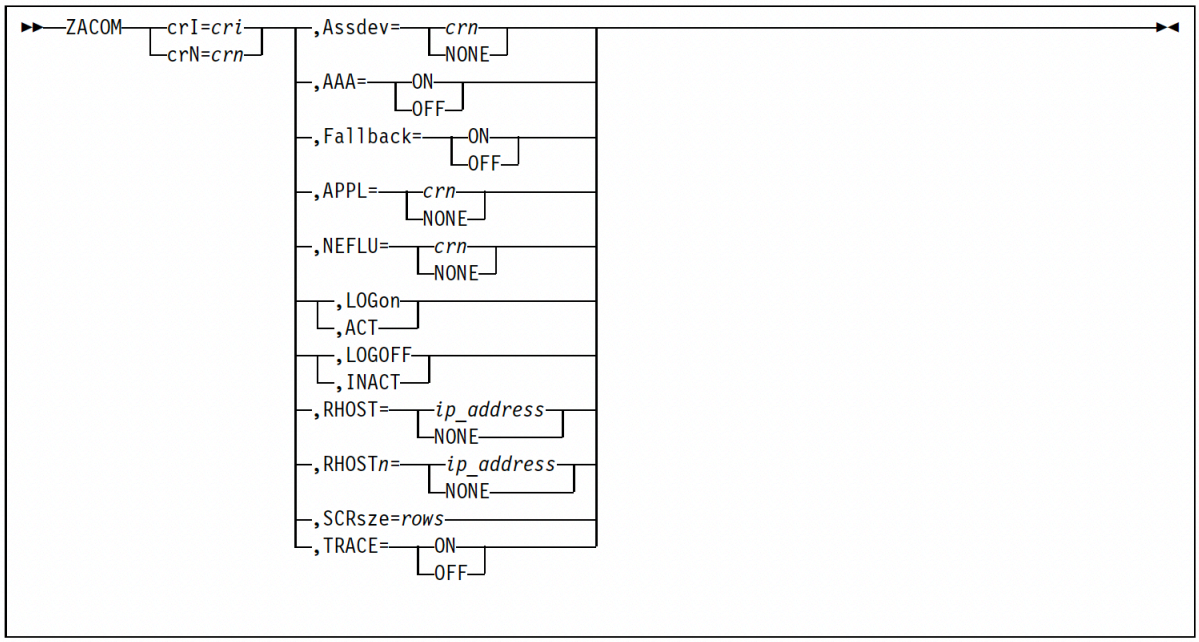
Table 15. Input terminal restrictions for assigning CRAS authority to the originating terminal if no SAF decision			
Function	**Prime CRAS**	**AT1 to AT16**	**AT17 to AT255**
CRASAUTH=PRIME	N/A	Yes	No
CRASAUTH=AT1-AT16	Yes	Yes	No
CRASAUTH=AT17-AT255	Yes	Yes	Yes

For further information see *ALCS Concepts and Facilities*.

Alter information about communication resources

Use the ZACOM command to alter information about one or more communication resources. See Table 16 on page 77 for the level of CRAS that can perform each function. Changes made in this way are not preserved across a system restart.

The format of the command is:



Where:

crI=*cri*
The CRI of the resource to be modified (6 hexadecimal digits).

crN=*crn*
The CRN of the resource to be modified (1 to 8 alphanumeric characters).

Assdev={*crn*|NONE}

crn
CRN of the resource that is to become the associated resource.

NONE
Remove any currently associated resource.

AAA={ON|OFF}
Some applications use a facility called ***terminal hold*** (also called "AAA hold" and the "message being processed indicator"). While terminal hold is in effect, these applications reject any further input messages from that terminal. Terminal hold is usually set on and off by an application input message editor but it is sometimes useful to be able to do it from a CRAS terminal. For example, to reject input messages from a terminal while changing its status, (AAA=ON) or to restore a terminal if an application fails to reset the hold (AAA=OFF).

AAA=0FF also transmits a message to the specified terminal. The message resets the terminal interlocks so that the agent can enter another input message.

ON
Set terminal hold on for the resource. An application that uses terminal hold rejects input messages from the terminal.

OFF
Set terminal hold off for the resource. An application that uses terminal hold accepts input messages from the terminal.

Fallback={ON|OFF}
For a terminal, set the ***fallback indicator*** on or off. The CRAS= parameter of the COMDEF generation macro defines the initial state of the fallback indicator. This is described in *ALCS Installation and Customization*.

The fallback indicator determines whether a CRAS printer (AT1 to AT16 or AP1 to AP16) can become the new RO CRAS if the current RO CRAS fails.

ON
Set the fallback indicator ON for the resource. The terminal is a fallback candidate.

OFF
Set the fallback indicator OFF for the resource. The terminal is not a fallback candidate.

APPL={*crn*|NONE}
Alter input routing for a terminal.

crn
Set input routing for the resource to an ALCS application, where *crn* is the CRN of the application.

NONE
Remove input routing for the resource.

NEFLU={*crn*|NONE}
For an ALCI connected ALC terminal, set or remove the owning ALCI logical unit (LU). ALCS only allows this command when the terminal is inactive. See the COMGEN and COMDEF generation macros in *ALCS Installation and Customization* for more information about ownership of ALCI terminals.

crn
CRN of the owning ALCI logical unit (LU).

NONE
Remove the owning ALCI LU for the resource.

LOGon|ACT
Activate the resource as follows:

- For a VTAM resource, initiate VTAM logon processing. The result is to establish a VTAM session between ALCS and the specified resource.
- For an ALCI LU (LDTYPE=VTAMALC and TERM=NEFLU), all the resources controlled by the specified ALCI LU become active when the session is established.
- For an X.25 PVC (LDTYPE=X25PVC), all the resources controlled by the specified X.25 PVC become active when the session is established.
- For an ALCS link (LDTYPE=ALCSLINK and TERM=LU61) initiate VTAM logon processing for all parallel sessions owned by that link.
- For a parallel session within an ALCS link (LDTYPE=ALCSLINK and TERM=PARSESS) initiate VTAM logon processing for the parallel session only.
- For an ALC terminal, set the resource active in the relevant ALCS communication table entry.
- For an ALC terminal, set the resource active in the relevant ALCS communication table entry. If the owning ALCI LU, X.25 PVC, SLC link, TCP/IP server, WAS resource, or MQ resource is active, the terminal becomes available for sending and receiving messages.
- For an other-system terminal, set the resource active in the relevant ALCS communication table entry. If there is an active message-router link to the owning system then the other-system terminal becomes available for sending and receiving messages.
- For an application owned by another system, set the resource active in the relevant communication table entry. If there is an active message router link to the owning system then the remote application becomes available for processing input messages.
- For an ALCS application, set the resource active in the ALCS communications table. This makes it available to process input messages.
- For an APPC resource (LDTYPE=APPC), allocate a session between the local LU and the partner LU defined for this resource. And on that session, allocate a conversation between the local TP and the partner TP.

For an APPC resource defined with COMDEF PRTCOL=TYPE1 it does not become active until the conversation is allocated successfully and ALCS receives an inbound ALLOCATE request for a second conversation. Once the outbound and the inbound conversations are allocated, ALCS waits for data to arrive on the inbound conversation.

For an APPC resource defined with COMDEF PRTCOL=TYPE2 it becomes active as soon as the conversation is allocated successfully.

- For a TCP/IP client connection (LDTYPE=TCPIP,TERM=CLIENT), start a connection between ALCS and the remote server port.

  For a TCP/IP server connection (LDTYPE=TCPIP,TERM=SERVER), start waiting for connections on the local server port.
- For an MQ resource (LDTYPE=MQ), set the resource active in the relevant ALCS communication table entry. If there are any messages on the MQSeries request queue, start processing them.
- For a WAS resource (LDTYPE=WAS), set the resource active in the relevant ALCS communication table entry.

LOGOFF|INACT

Inactivate the resource as follows:

- For a VTAM resource, initiate VTAM logoff processing. The result is to terminate the VTAM session between ALCS and the specified resource.
- For an ALCI LU, (LDTYPE=VTAMALC and TERM=NEFLU), all the resources controlled by the specified ALCI LU become inactive when the session is terminated.
- For an X.25 PVC, all the resources controlled by the specified PVC become inactive when the session is terminated.
- For an ALCS link, initiate VTAM logoff processing for all parallel sessions owned by that link.
- For a parallel session within an ALCS link, initiate VTAM logoff processing for the parallel session only.
- For an ALC terminal, set the resource inactive in the ALCS communication table. This makes the terminal unavailable for sending and receiving messages. If it is a terminal controlled by an ALCS LU **do not** remove the owning ALCI LU.
- For an ALCS application, set the resource inactive in the ALCS communications table. This makes it unavailable to process input messages. No further input messages are passed to the application for processing.
- For an other-system terminal, set the resource inactive in the relevant ALCS communication table entry. This makes it unavailable for sending and receiving messages.
- For an application owned by another system, set the resource inactive in the relevant communication table entry. This makes it unavailable for processing input messages.
- For an APPC resource (LDTYPE=APPC), deallocate any conversations with the partner TP defined for this resource.
- For a TCP/IP client connection (LDTYPE=TCPIP,TERM=CLIENT), stop the connection between ALCS and the remote server port.
- For a TCP/IP server connection (LDTYPE=TCPIP,TERM=SERVER), stop any current connections on the local server port and stop waiting for new connections.
- For a TCP/IP dynamic server connection, stop that particular connection on the local server port.
- For an MQ resource (LDTYPE=MQ), set the resource inactive in the relevant ALCS communication table entry.
- For a WAS resource (LDTYPE=WAS), set the resource inactive in the relevant ALCS communication table entry.

RHOST={*ip_address*|NONE}
For an ALC terminal connected through TCP/IP, set or remove the remote host or gateway. For a TCP/IP connection where the ALCS application is the client, set or remove the IP addresses of the remote system. Specify the IP address in dotted decimal notation, for example: 9.180.147.125. ALCS allows this command only when the terminal or TCP/IP connection is inactive. See the COMGEN and COMDEF generation macros in *ALCS Installation and Customization* for more information about ownership of TCP/IP ALC terminals.

RHOST*n*={*ip_address*|NONE}
For an ALC terminal connected through TCP/IP, set or remove the primary and up to three alternate IP addresses of the remote host or gateway. For a TCP/IP connection where the ALCS application is the client, set or remove the primary and up to three alternate IP addresses of the remote system. RHOST1 refers to the primary IP address (a synonym for RHOST), RHOST2, RHOST3, and RHOST4 refer to the alternate IP addresses. Specify the IP address in dotted decimal notation, for example: 9.180.147.125. ALCS allows this command only when the TCP/IP connection is inactive.

SCRsze=*rows*
For an ALC terminal, change the number of lines on the display screen. Specify a decimal number in the range 12 through 255.

TRACE={ON|OFF}
For a TCP/IP connection, start (ON) or stop (OFF) tracing TCP/IP activity.

See the description of the TCP/IP trace facility in "TCP/IP trace facility" on page 385 for information about the trace output on the ALCS diagnostic file.

See "ZDCOM -- Display communication resource information" on page 136 for details of how to display the results of the request.

Restrictions

Restrictions depend on the ZACOM function. Table 16 on page 77 shows the level of CRAS authority that can perform each function, a blank entry indicates that the function is not allowed on that level of CRAS.

Table 16. ZACOM -- Input terminal restrictions

Function	Prime CRAS authority	AT1 to AT16 authority	AT17 to AT255 authority
Assdev	Yes	Yes	
AAA=ON	Yes		
AAA=OFF	Yes	Yes	
APPL	Yes		
LOGon|ACT	Yes	Yes	
LOGOFF|INACT	Yes	Yes	
Fallback=ON|OFF	Yes	Yes	
NEFLU	Yes		

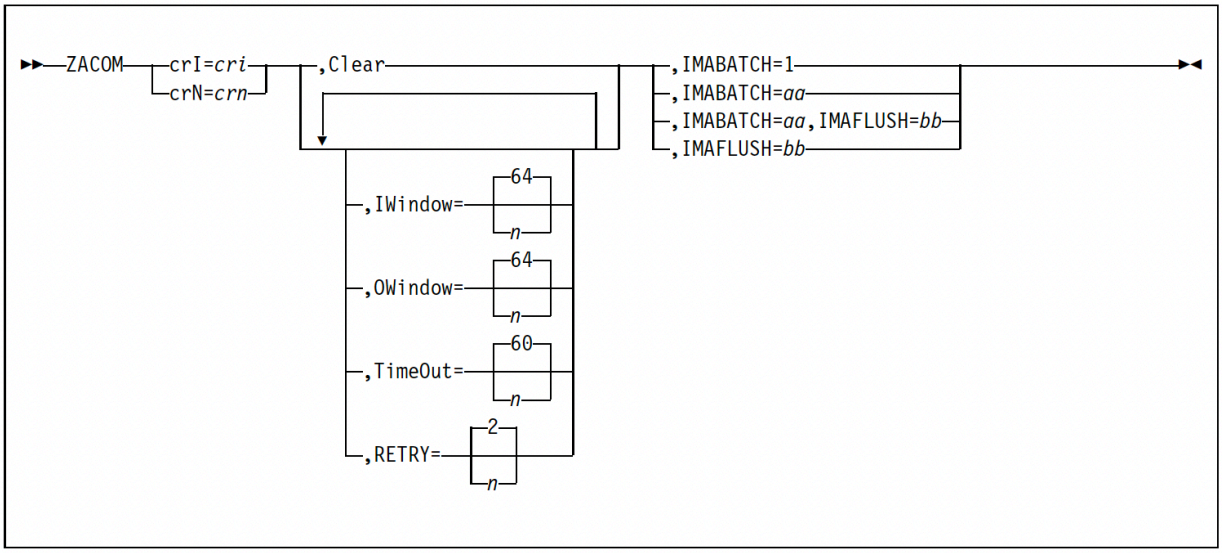
Set or clear BATAP variables for AX.25 and MATIP Type B message processing

Use the ZACOM command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal only*** -- to clear or set BATAP variables.

Note: These changes are preserved over an ALCS restart.

BATAP is a protocol used for end-to-end control of AX.25 and MATIP Type B messages between ALCS and a high level network for example SITA or ARINC. The BATAP protocol uses a number of variables which are specified in the ALCS communication generation for the relevant X.25 PVC or TCP/IP resource. Your installation must agree the values of these variables with the controller of the high level network.

The format of the command is:



Where:

cri
The CRI of the X.25 PVC or TCP/IP resource (6 hexadecimal digits).

crn
The CRN of the X.25 PVC or TCP/IP resource (1 to 8 alphanumeric characters).

Clear
Reset the BATAP control fields to their initial values.

IWindow=[64|*n*]
BATAP input window size. The maximum number of input messages awaiting acknowledgment; it is 1 or 2 numeric digits (default 64, minimum 1, maximum 64).

OWindow=[64|*n*]
BATAP output window size. The maximum number of output messages awaiting acknowledgment; it is 1 or 2 numeric digits (default 64, minimum 1, maximum 64).

TimeOut=[60|*n*]
BATAP timeout value. The length of time (in seconds) to await an acknowledgment; it is 1 to 3 numeric digits (default 60, minimum 1, maximum 120). BATAP sets a timer when it transmits a message and clears the timer when it receives an acknowledgment. If BATAP does not receive an acknowledgment within the timeout period, it resets the timer and retransmits the message. BATAP continues retransmitting the message until either it receives an acknowledgment or the X.25 PVC or TCP/IP resource is inactivated using the ZACOM INACT command.

ZACOM INACT is described in “Alter information about communication resources” on page 73.

RETRY=[2|*n*]
BATAP retry control value. The number of times that BATAP retransmits a message without receiving an acknowledgment before it sends an attention message to the RO CRAS; it is 1 or 2 numeric digits (default 2, minimum 1, maximum 64).

aa
IMA batch value. The maximum number of IMAs to be batched. One or two numeric digits (minimum 1, maximum 20).

Default is IMABATCH=1 indicating that IMAs are not batched.

Contact your SITA representative to determine a suitable value.

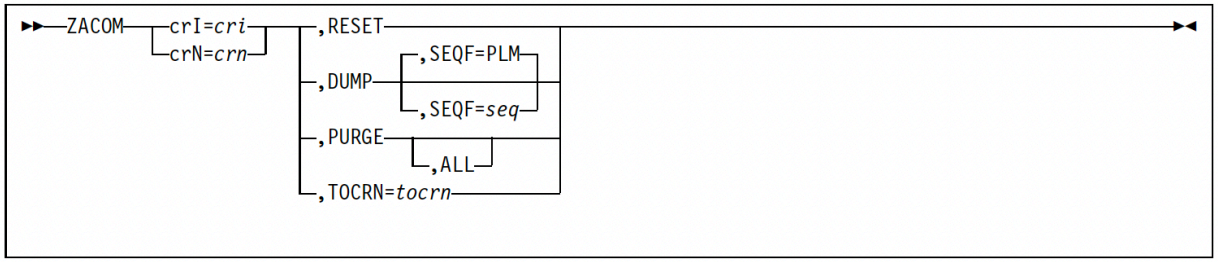
bb
IMA flush value. The maximum number of seconds before a batched IMA is sent, irrespective of the value of IMABATCH. One or two numeric digits (minimum 1, maximum 30).

Contact your SITA representative to determine a suitable value. The IMA flush value plus the network navigation time must not exceed the value of SITA's BATAP T1 timer.

Control ALCS LU 6.1 message queues

Use the ZACOM command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal only*** -- to control an ALCS LU 6.1 link.

The format of the command is:



Where:

cri
CRI of the resource (6 hexadecimal digits).

crn
CRN of the resource (1 to 8 alphanumeric characters).

RESET
Reset the link as though it has just been activated. The message queue is left intact.

DUMP[,SEQF={PLM|*seq*}]
Copy all the messages on the queue to the sequential file with symbolic name *seq* and then discard them. The default sequential file name is PLM.

PURGE
Discard the current message or the first message on the queue.

PURGE,ALL
Discard the current message and all messages on the queue.

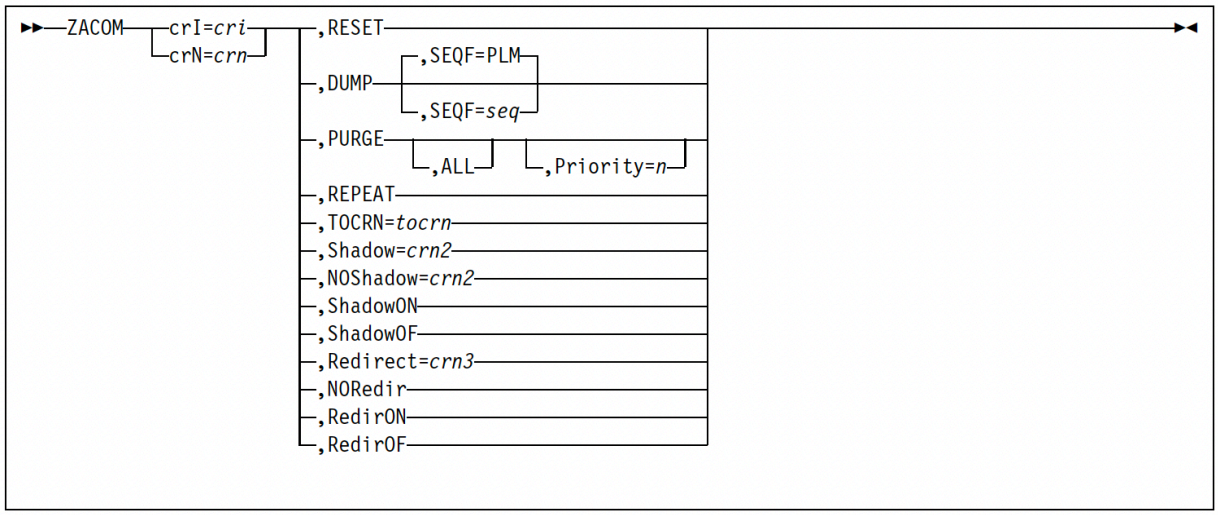
TOCRN=*tocrn*
Remove all messages from the queue for this link, and append them to the queue for the link with CRN *tocrn*. This process is called ***queue swing***.

Control ALCS terminal message queues

Use the ZACOM command -- ***from any CRAS authorized terminal*** -- to control an ALCS printer or display.

Use the ZACOM command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal only*** -- to control printer shadowing and redirection.

The format of the command is:



Where:

cri
CRI of the resource (6 hexadecimal digits).

crn
CRN of the resource (1 to 8 alphanumeric characters).

RESET
Re-initialize the ALCS printer as though it has just come on line. The message queues are left intact. Use this command to recover from a printer hung condition.

DUMP[,SEQF={*seq*|PLM}]
Copy all messages (except SLMTC messages but including unsolicited messages) on queue for the printer or display terminal to the sequential file with symbolic name *seq* and then discard them. The default sequential file name is PLM.

PURGE[,ALL][,Priority=*n*]
Discard the message currently being sent to the printer or display terminal. If there is no current message, discard the first message on the highest priority queue that is not empty. You can use this command to discard a message that is causing problems on the printer and to discard unsolicited message queues to the printer or display terminal.

ALL
If there is a current message being sent to the printer or display terminal, discard it; and in any case, discard all messages that are on queue.

Priority=*n*
The priority of the queue to purge, a decimal number in the range 0 through 14. The status of the resource must be either inactive or unusable.

REPEAT
Repeat the last message sent to the printer.

TOCRN=*tocrn*
Remove all messages (except SLMTC messages) on queue for the printer or display terminal with CRN *crn* or CRI *cri*, and append them to the queues for the printer with CRN *tocrn*.

Shadow=*crn2*
Add the printer with CRN *crn2* to the list of shadow printers for the printer with CRN *crn* or CRI *cri*.

NOShadow=*crn2*
Remove the printer with CRN *crn2* from the list of shadow printers for the printer with CRN *crn* or CRI *cri*.

ShadowON
Start shadowing messages for the printer with CRN *crn* or CRI *cri*. If there are no shadow printers, ALCS responds with an attention message.

ShadowOF
Stop shadowing messages for the printer with CRN *crn* or CRI *cri*.

Redirect=*crn3*
Define the printer with CRN *crn3* as the redirection printer for the printer with CRN *crn* or CRI *cri*. Only one redirection printer can be defined for a printer.

NORedir
Remove redirection printer for the printer with CRN *crn* or CRI *cri*.

RedirON
Start redirecting messages for the printer with CRN *crn* or CRI *cri*.

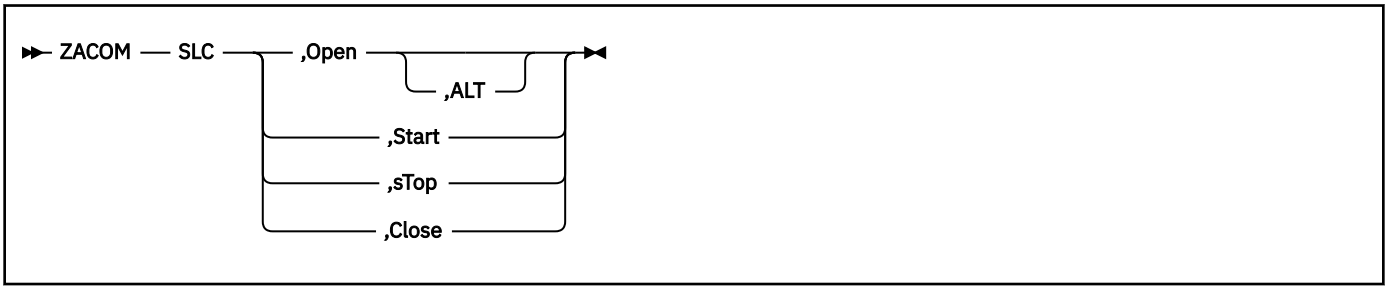
RedirOF
Stop redirecting messages for the printer with CRN *crn* or CRI *cri*.

See "Printer shadowing" on page 57 for more information on printer shadowing and see "Printer redirection" on page 57 for more information on printer redirection.

Control an SLC link or channel

Use the ZACOM command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal only*** -- to open, start, stop, or close an SLC link or channel, or all SLC links.

The format of the command to open, start, stop, or close ***the entire SLC network*** is:



Where:

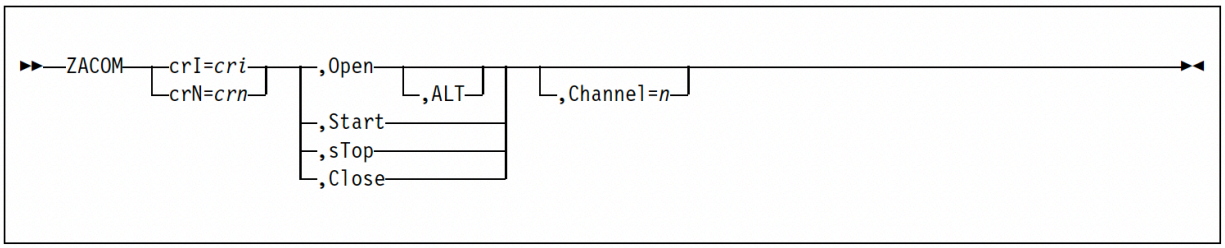
SLC
Open, start, stop or close all SLC links.

ALT
When ALCS opens an SLC channel, it normally uses the prime EP subchannels for the send and receive sides to dynamically allocate telecommunications lines.

Specify ALT to use the alternate EP subchannels instead. For more information, see the description of the COMDEF generation macro in *ALCS Installation and Customization*.

Note: Always use the ALT= parameter to open an SLC channel using the alternate EP subchannel address as ALCS reverts to using the prime subchannel address after a system restart, a system state change, or another ZACOM SLC command.

The format of the command to open, start, stop, or close one or more channels on an SLC link is:



Where:

cri
CRI of the link (6 hexadecimal digits).

crn
CRN of the link (1 to 8 alphanumeric characters).

Channel=*n*
Number of the channel within the link; a number from 1 to 7. Omit this parameter to open, close, start or stop all channels on the link.

ZACOR -- Alter storage

Use the ZACOR command -- ***from any CRAS authorized terminal*** -- to alter application global area storage above the bar.

Changes to the application global area are preserved across a system restart only if the changed storage is in a keypointable global record.

The format of the command is:

Where:

address
Storage address in hexadecimal digits. (You can omit leading zeros.)

data
Hexadecimal data to write at the specified address. It must not cross a doubleword boundary, and must be an even number of digits, up to a maximum of 16.

comments
On the ***next line***, indicated by (NL) in the syntax diagram, enter at least 6 and up to 40 characters describing the reason for the change.

Normal response

ALCS sends a response to the RO CRAS, and a copy, without the comments, to the originating terminal.

```
DXC8200I CMD i hh.mm.ss BCOR  address
oooo    dddddddd dddddddd              *cccccccc      * ALTERED TO
        eeeeeeee eeeeeeee              *ffffffff      *
CRN=crn comments
```

Where:

address
Storage address of the storage altered.

oooo
Low 4 hexadecimal digits of the storage address of the storage displayed. (For alterations not on a fullword boundary, this is the address of the fullword containing the first altered byte.)

dd...dd
Storage contents (hexadecimal) before the alteration.

ee...ee
Storage contents (hexadecimal) after the alteration.

cc...cc
Storage contents (character) before the alteration.

ff...ff
Storage contents (character) after the alteration.

crn
CRN of the originating terminal.

comments
Reason for the change.

ZBC0R-- Alter storage above the bar

Use the ZBC0R command -- ***from any CRAS authorized terminal*** -- to alter application global area storage above the bar.

The format of the command is:

Where:

address
Storage address in hexadecimal digits. (You can omit leading zeros.)

data
Hexadecimal data to write at the specified address. It must not cross a doubleword boundary, and must be an even number of digits, up to a maximum of 16.

comments
On the ***next line***, indicated by (NL) in the syntax diagram, enter at least 6 and up to 40 characters describing the reason for the change.

Normal response

ALCS sends a response to the RO CRAS, and a copy, without the comments, to the originating terminal.

```
DXC8249I CMD i hh.mm.ss BCOR  address
oooo    dddddddd dddddddd              *cccccccc      * ALTERED TO
        eeeeeeee eeeeeeee              *ffffffff      *
CRN=crn comments
```

Where:

address
Storage address of the storage altered.

oooo
Low 4 hexadecimal digits of the storage address of the storage displayed. (For alterations not on a fullword boundary, this is the address of the fullword containing the first altered byte.)

dd...dd
Storage contents (hexadecimal) before the alteration.

ee...ee
Storage contents (hexadecimal) after the alteration.

cc...cc
Storage contents (character) before the alteration.

ff...ff
Storage contents (character) after the alteration.

crn
CRN of the originating terminal.

comments
Reason for the change.

ZAFIL -- Alter DASD record

Use the ZAFIL command -- ***from any CRAS authorized terminal*** -- to alter up to 16 bytes of a DASD record. If the system has a duplicated database ZAFIL modifies both the primary and the duplicate copies.

The format of the command is:

Where:

file_address
File address; 8 or 16 hexadecimal digits.

type
Any of the following:

GF-*nnn*
nnn is the 3-digit general file number.

#xxxxx
Fixed record type (for example, #WAARI).

LsLTpool
Long-term pool record, where Ls identifies the record size.

LsSTpool
Short-term pool record, where Ls identifies the record size.

Lspool
Allocatable pool record, where Ls identifies the record size.

ordinal
Record ordinal number; a decimal number.

dis
Starting displacement within the record. It can be up to 4 hexadecimal digits.

data
Hexadecimal data to write at the specified displacement in the record. It must be an even number of digits, up to a maximum of 32 if the starting displacement is a fullword boundary. The maximum reduces by 2 digits for each byte displacement from a fullword boundary.

comments
On the ***next line***, indicated by (NL) in the syntax diagram, enter at least 6 and up to 40 characters of comments describing the reason for the change.

Normal response

ALCS sends a response to the RO CRAS, and a copy, without the comments, to the originating terminal.

```
DXC8202I CMD i hh.mm.ss AFIL  file_address
oooo    dddddddd dddddddd dddddddd dddddddd  *ccccccccccccccc*ALTERED TO
        eeeeeeee eeeeeeee eeeeeeee eeeeeeee  *fffffffffffffffff*
CRN--crn comments
```

Where:

file_address
File address.

oooo
Displacement within the record of the data displayed. (For alterations not on a fullword boundary, this is the displacement of the fullword containing the first altered byte.)

dd...dd
Record contents (hexadecimal) before the alteration.

ee...ee
Record contents (hexadecimal) after the alteration.

cc...cc
Record contents (character) before the alteration.

ff...ff
Record contents (character) after the alteration.

crn
CRN of the originating terminal

comments
Reason for the change.

Message number DXC8202I is used when the file address has 8 hexadecimal digits (4 bytes); message number DXC8231I is used when the file address has 16 hexadecimal digits (8 bytes).

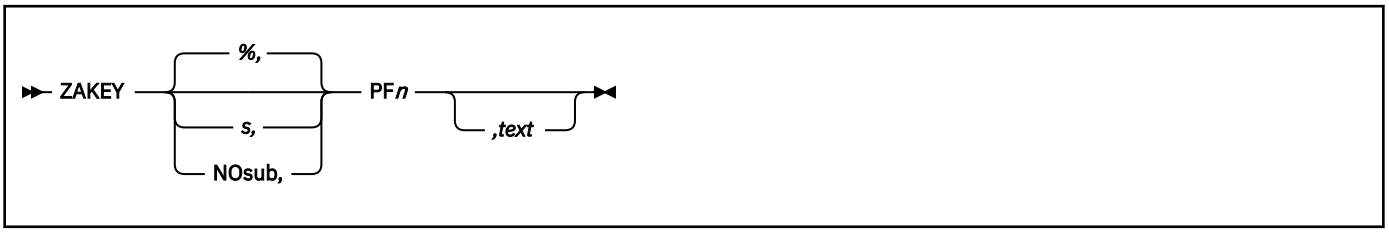
ZAKEY -- Alter terminal program function key settings

Use the ZAKEY command -- ***from any 3270 terminal*** -- to alter the program function (PF) key settings for your terminal.

The default PF key settings are defined in an installation-wide exit. You can alter them in various ways. The simplest way is to specify that when you press a specific PF key (for example, PF7) the application receives a text message (for example, ZSCRL UP) from that terminal.

You can also use substitution (see "Substitution" on page 87 below) so that when you enter a message and press the PF key, the application receives a more complete message from the terminal.

The format of the ZAKEY command is:



Where:

s
Substitution trigger character, the default is %. See "Substitution" on page 87.

NOsub
Inhibit substitution.

PF*n*
Program function key number *n*. Specify PF1 or PF01 for program function key one, and so on. Allowed range is PF1 to PF24.

text
Text of the input message or ALCS command that this key produces. This text can contain significant newline characters. It can also contain ***substitution tokens*** (see "Substitution" on page 87). A new-line character or a comma in *text* is treated as a space character.

The maximum length of *text* is 79 characters.

If you omit *text*, ALCS resets the key to its default setting.

Restriction

You cannot use PF keys to enter the ZLOGF, ZROUT, ZDUMP commands, or any of the trace commands.

Substitution

Before pressing the defined function key, you can type in text for inclusion in the message passed to the application. This is called ***substitution***.

If you have inhibited substitution, by specifying NOsub in the corresponding ZAKEY command, this input text is discarded. Pressing the function key then has the same effect as entering the input message or command *text* exactly as it appears in the ZAKEY command.

If you have specified (or defaulted) a substitution trigger character, *text* can contain substitution tokens. Pressing the function key has the same effect as entering the input message or command *text* after replacing each substitution token with the corresponding word from the input text.

There are three sorts of substitution:

Period substitution (*s.*)
ALCS replaces *s*. with the next word from the text typed in before pressing the PF key.

Asterisk substitution (*s)**
ALCS replaces *s** with the remaining words from the text typed in before pressing the PF key.

Number substitution (*sn*)
ALCS replaces *sn* with the *n* ' th word from the text typed in before pressing the PF key.

It is possible to combine period, asterisk and number substitution in the same PF key. The following examples describe how to use substitution using the default substitution trigger %.

Period substitution

ZAKEY command:

```
ZAKEY % PFn AAAA %. BBBB %.
```

if you type XXX YYYYY and press PF*n*, the following message is sent to the application:

```
AAAA XXX BBBB YYYYY
```

Asterisk substitution

ZAKEY command:

```
ZAKEY % PFn AAAA %* BBBB
```

If you type: XXX YYYYY and press PF*n*, the following message is sent to the application:

```
AAAA XXX YYYYY BBBB
```

Number substitution

ZAKEY command:

```
ZAKEY % PFn AAAA %2 BBBB %1
```

If you type: XXX YYYYY and press PF*n*, the following message is sent to the application:

```
AAAA YYYYY BBBB XXX
```

Combined period, asterisk and number substitution

ZAKEY command:

```
ZAKEY % PFn AAA %1 BBB %2 CCC %.%..
DDD %* EEE%.FFF
```

If you type: `HELLO JOHN, AND GOOD LUCK` and press PF*n*, the following message is sent to the application:

```
AAA HELLO BBB JOHN CCC HELLOJOHN.DDD AND GOOD LUCK EEEFFF
```

Note: Your system programmer may have implemented an installation-wide exit that prevents you from changing some or all of the PF keys from their default settings.

In all cases, if there is no input text corresponding to a substitution token then ALCS substitutes a null (zero length) character string.

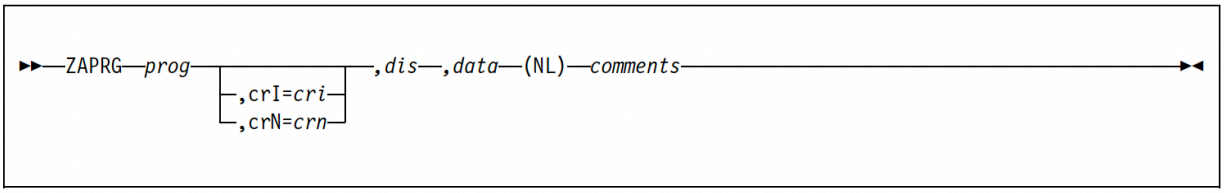
ZAPRG -- Alter program

Use the ZAPRG command -- ***from any CRAS authorized terminal*** -- to alter up to 16 bytes of a test version (loaded with ZPCTL LOAD,TEST) of an ALCS application program.

Only a Prime CRAS authorized terminal can alter another terminal's test version of a program. Any CRAS terminal can alter its own test version of a program. The copy of the program on the load module library is not modified.

Alterations to programs by ZAPRG are not retained over an ALCS restart, or if the program is unloaded and reloaded.

The format of the command is:



Where:

prog
4-character name of the ALCS application program.

crI=*cri*|(crN=*crn*
Not required if ZAPRG is entered from the terminal whose test version of the program is to be altered.

The terminal whose test version of the program is to be altered, where:

cri
CRI of the terminal (6 hexadecimal digits).

crn
CRN of the terminal (1 to 8 alphanumeric characters).

dis
Displacement in program. It is the listing address printed as LOC in the assembly listing; you can omit leading zeros.

data
Hexadecimal data to write at the specified location. It must be an even number of digits, up to a maximum of 32 if the starting displacement is a fullword boundary. The maximum reduces by 2 digits for each byte displacement from a fullword boundary.

comments
On the ***next line***, indicated by (NL) in the syntax diagram, enter at least 6 and up to 40 characters of comments describing the reason for the change.

Normal response

ALCS sends a response to the RO CRAS, and a copy, without the comments, to the originating terminal.

```
DXC8220I CMD i hh.mm.ss APRG
progvv Module module  Size size Address address
Owning CRN crn
oooo    dddddddd dddddddd dddddddd dddddddd  *ccccccccccccccc* ALTERED TO
        eeeeeeee eeeeeeee eeeeeeee eeeeeeee  *fffffffffffffff*
CRN crn2 comments
```

Where:

progvv
Program name and version number.

module
Load module name.

address
Address of the program storage.

crn
CRN of the terminal that owns the test program.

size
Size of the program.

oooo
Displacement within the program of the data displayed. (For alterations not on a fullword boundary, this is the displacement of the fullword containing the first altered byte.)

dd...dd
Program contents (hexadecimal) before the alteration.

ee...ee
Program contents (hexadecimal) after the alteration.

cc...cc
Program contents (character) before the alteration.

ff...ff
Program contents (character) after the alteration.

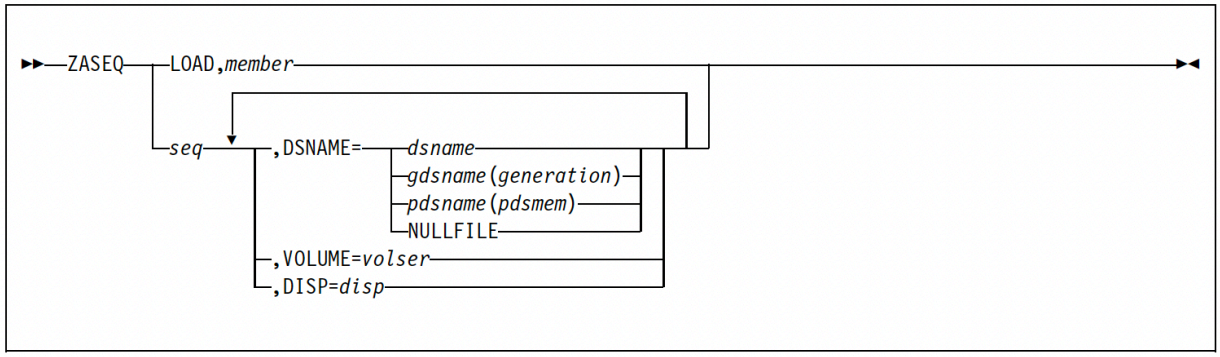
crn2
CRN of the originating terminal.

comments
Reason for the change.

ZASEQ -- Alter sequential file definition

Use the ZASEQ command -- ***from any CRAS authorized terminal*** -- to load a sequential file configuration table, or to alter the definition of a general sequential file. Changes made in this way are not preserved across an ALCS restart.

The format of the command is:



Where:

LOAD
Load a sequential file configuration table. ZASEQ updates the existing sequential file configuration as follows:

- For existing general sequential files, it updates the existing definition.
- For new general sequential files, it adds the new definition.
- For other sequential files, it ignores the new definition. "ZDSEQ -- Display sequential file status" on page 194 lists the different types of sequential file.

member
Load module name of the sequential file configuration table.

seq
Symbolic name of the general sequential file. ZASEQ updates the definition of the general sequential file according to the parameter or parameters that follow.

DSNAME={*dsname*|*gdsname*(*generation*)|*pdsname*(*pdsmem*)|NULLFILE}
New data set name, where:

dsname
Data set name of a physical sequential data set.

pdsname
Data set name of a partitioned data set (PDS or PDS-E). The maximum length for *pdsname* is 34 characters.

pdsmem
Name of a PDS member.

gdsname
Data set name of a general data set (GDS), up to a maximum of 34 characters.

generation
Relative generation number in the form *+nn* or *-nn*.

To make a data set into a dummy data set, specify *dsname* as `NULLFILE`.

The data set name can optionally contain one or more special character strings. When ALCS allocates the data set, it replaces the string as follows:

String	Replacement value
-GDAT	Greenwich mean time Julian date in the format yyddd.
-LDAT	Local time Julian date in the format yyddd.
-GTIME	Greenwich mean time in the format hhmmss.
-LTIME	Local time in the format hhmmss.

These dates and times are the MVS system dates and times, not ALCS dates and times.

Note: MVS does not allow data set name qualifiers to begin with numeric characters. For example, do not specify DSNAME as:

```
QUAL1.QUAL2.-GDAT.-GTIME.QUAL3
```

Instead, specify it as (for example):

```
QUAL1.QUAL2.D-GDAT.T-GTIME.QUAL3
```

VOLUME=*volser*
New volume serial for the sequential file.

DISP=*disp*
New disposition of the sequential file, for example:

```
DISP=(OLD,KEEP,KEEP)
DISP=(NEW,CATLG,CATLG)
```

Restriction

ZASEQ only updates general sequential file definitions.

Normal responses

To ZASEQ LOAD:

```
DXC8726I CMD i hh.mm.ss ASEQ
Sequential file configuration table member load complete
```

The ZASEQ command with all other parameters produces a display of the new sequential file configuration on the originating terminal:

```
DXC8729I CMD i hh.mm.ss ASEQ Sequential file seq update complete
File    Device     Status    Type     Volume    Data-set-name
seq     device     status    type     volser    dsname
DISP=disp                          b Blocks read/written
⋮
```

A display of the previous sequential file configuration and a copy of the new configuration (if update was successful) are sent to RO CRAS:

```
DXC8734I CMD i  hh.mm.ss ASEQ  Sequential file seq before update
File    Device      Status     Type     Volume    Data-set-name
seq     device      status     type     volser    dsname
DISP=disp                             b  Blocks read/written

DXC8729I CMD i  hh.mm.ss ASEQ  Sequential file seq update complete
File    Device      Status     Type     Volume    Data-set-name
seq     device      status     type     volser    dsname
DISP=disp                             b  Blocks read/written
⋮
```

See “ZDSEQ -- Display sequential file status” on page 194 for an explanation of the fields.

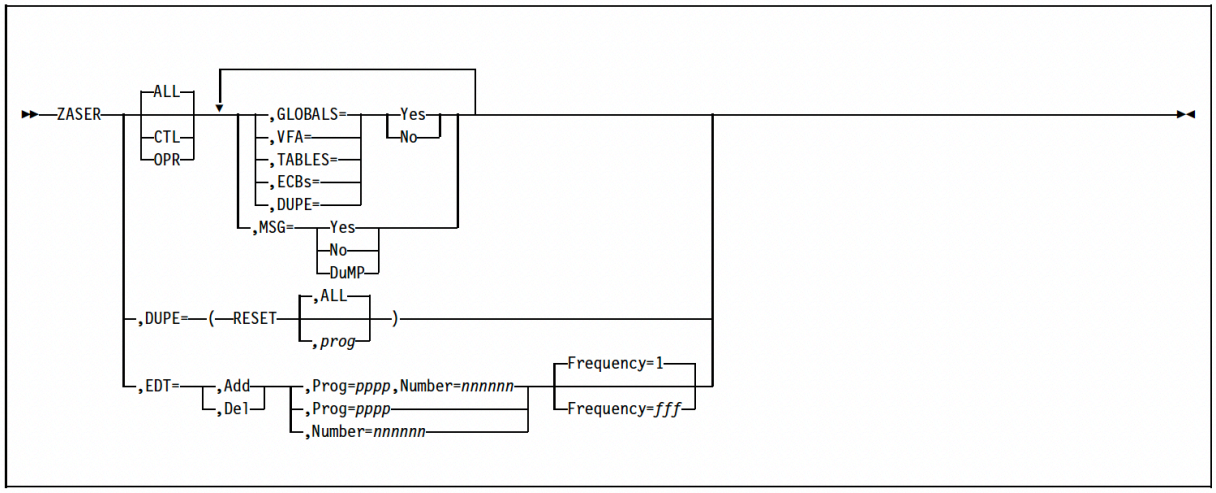
ZASER -- Alter system error options

Use the ZASER command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal only*** -- to alter the system error options. Changes made in this way are not preserved across an ALCS restart.

The system error options determine the action taken in case of error. The initial settings are specified in the ALCS system generation options. This is described in *ALCS Installation and Customization*.

Use the ZASER command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal only*** -- to add or delete an entry to/from the Exception Dump Table. Changes made in this way are preserved across an ALCS restart.

The format of the command is:



Specify one or more keyword parameters in any order, separated by commas or spaces, to alter one or more system error options. Only those options included in the input message are modified.

Where:

CTL
Alter options on control (CTL) type dumps.

Modify the Exception Dump Table for control (CTL) type dumps.

OPR
Alter options on operational (OPR) type dumps.

Modify the Exception Dump Table for operational (OPR) type dumps.

ALL
Alter options on both types of dump (default).

Modify the Exception Dump Table for both types of dump (default).

GLOBALS={Yes|No}

Yes
Dump the global area

No
Do not dump the global area

VFA={Yes|No}

Yes
Dump the VFA contents (VFA buffers, VFA buffer headers, VFA record locator table, and VFA control area).

No
Do not dump the VFA contents.

TABLES={Yes|No}

Yes
Dump all the online monitor tables (formatted and unformatted), VFA control area, and IOCBs.

No
Dump only selected sections of the formatted online monitor tables.

ECBs={Yes|No}

Yes
Dump entry storage for all entries.

No
Dump entry storage for the failing entry only. This option is ignored during catastrophic error processing.

MSG={Yes|No|DuMP}

Yes
Send a message to the RO CRAS for each system error.

No
Do not send a message to the RO CRAS for each system error.

DuMP
Send a message to the RO CRAS for each system error, but not for duplicate system errors.

The message is always written to the ALCS diagnostic file.

DUPE={Yes|No|(RESET[,ALL|*prog*])}

Yes
Take a dump for each system error, including duplicate system errors.

No
Suppress duplicate system error dumps at the same displacement in the same program.

(RESET,ALL)
Remove all items from the duplicate dump table. (This is the default.)

(RESET,*prog*)
Remove any items related to program *prog* from the duplicate dump table.

Edt=Add
Add an entry to the Exception Dump Table.

Edt=Del
Delete an entry from the Exception Dump Table. .

pppp
Program name. This can be a specific program name (4 characters) or a generic program name (1 to 3 characters). For example XP selects all program names starting with the characters XP.

nnnnnn
System error code. This is either one, two, three, or six characters. ALCS will pad this code with leading zeroes to create a six character error number. This parameter is required when you specify CTL or OPR with Edt=ADD or Edt=DEL.

fff
Frequency. The frequency with which ALCS will take a dump for this system error in the Exception Dump table. If set to 1 a dump will be taken on every occurrence. If set higher, a dump will not result

on every occurrence. This number specifies how often a full dump will result instead of a nodump. It is a decimal number in the range 1 through 999 (1 is the default).

You can use the Exception Dump table to override the duplicate dump system error option DUPE=NO. ALCS takes a dump for each system error which matches an entry in the Exception Dump Table, including duplicate system errors.

Normal responses

ALCS sends a response to the RO CRAS and to the originating terminal.

The response to ZASER DUPE=RESET without any other options:

```
DXC8581I CMD i hh.mm.ss ASER
Duplicate dump table reset OK
```

The response to ZASER EDT=ADD or ZASER EDT=DEL:

```
DXC8273I CMD i hh.mm.ss ASER
Exception dump table update OK
```

To ZASER with parameter options:

```
DXC8582I CMD i hh.mm.ss ASER  System error dump option modified to
Option                CTL-Dumps      OPR-Dumps
Duplicate dumps....... aaa          aaa
Monitor tables dump... aaa          aaa
Entry storage dump.... aaa          aaa
Global area........... aaa          aaa
Dump message.......... aaa          aaa
Nodump message........ aaa          aaa
VFA buffers dump...... aaa          aaa
```

Where *aaa* are the new options.

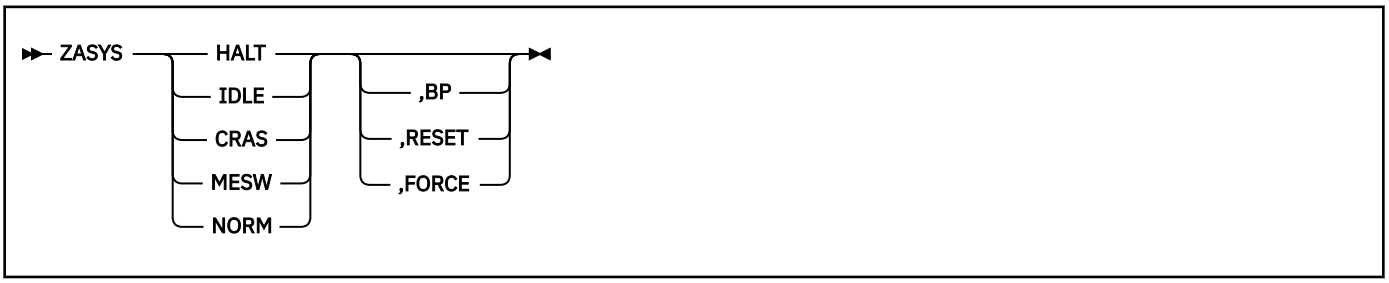
ZASYS -- Alter system state

Use the ZASYS command -- ***from a Prime CRAS authorized terminal only*** -- to alter the ALCS system state.

Notes:

1. It can take several minutes for ZASYS to complete normally, depending on the application design. Allow for this delay and ***consult your system programmer*** before you repeat the ZASYS command with RESET or FORCE.
2. If Recoup is running, altering the system state (which, however, requires the BP parameter) automatically causes Recoup to pause.

The format of the command is:



Where:

HALT
Halt ALCS. You can use ZASYS HALT in any system state.

IDLE
Change to IDLE state.

CRAS
Change to CRAS state.

MESW
Change to message switching state (MESW).

NORM
Change to normal (NORM) state.

BP
Bypass the check for active utilities. A change of system state while utility programs are running can cause the utilities to terminate abnormally. ALCS therefore checks for active utilities before altering the system state. Consult your system programmer before you use BP.

RESET
Use this parameter when a state change entry ended abnormally (exited) with the state change incomplete. RESET sets the state change indicator to the required state.

FORCE
Use this parameter only after an unsuccessful ZASYS with RESET, when a state change entry is held up; that is the state change is incomplete, but the entry is still present. FORCE changes the system state to the required value, and forces the previous state change entry to exit.

Note: ALCS only accepts ZASYS commands with the RESET or FORCE parameters when a system state change is in progress.

Normal responses

```
DXC8235I CMD i hh.mm.ss ASYS
ALCS State change from xxxx to xxxx starting


DXC8028I CMD i hh.mm.ss ASYS OK
```

Where *xxxx* is one of the following:

HALT
Halt state.
SYSTEM IN HALT STATE means that ALCS is about to terminate.

IDLE
Idle state.

CRAS
CRAS state.

MESW
Message switching state.

NORM
Normal state.

ALCS sends the following message to the RO CRAS when state change starts:

```
DXC2003I SYS hh.mm.ss ALCS state change from SS1-'xxxx' to SS2-'xxxx' starting
```

ALCS sends the following message to the RO CRAS when state change completes:

```
DXC2004I SYS hh.mm.ss ALCS in SS1-'xxxx' state
```

ZATIM -- Alter time

Use the ZATIM command -- ***from a Prime CRAS authorized terminal only*** -- to alter or set the time and date, or to change the difference between local time and Greenwich Mean Time (GMT). The change is preserved across a system restart.

Note: ZATIM does not affect the system time maintained by MVS.

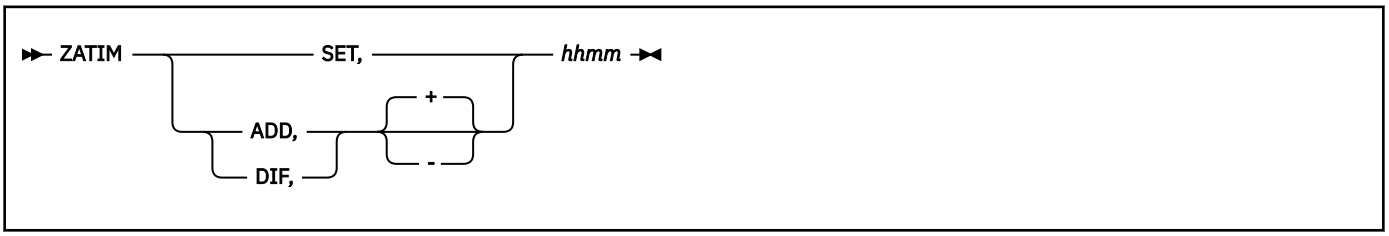
Restrictions

The following rules apply to a change to either the local time date or the GMT date.

1. A negative date changed is not allowed.
2. A positive date change is allowed provided you confirm the action (see "Confirmation of commands" on page 60).

These restrictions can be changed by installing the installation-wide ECB-controlled exit ATM1 on your system.

The format of the command is:



Where:

ADD,[+|-]*hhmm*
Add (or subtract) the value of *hhmm* (hours and minutes, 24-hour clock format, complete with leading zeros) to (or from) the local time and GMT. The maximum value for *hhmm* is 2359. If you need to change the date, refer to your installation guidelines.

SET,*hhmm*
Set the local time to *hhmm* (hours and minutes, 24-hour clock format, complete with leading zeros), and reset GMT according to the current difference. Express® midnight as 0000. The date changes at 0000.

Note: If you enter ZATIM SET at 2359 and you have the installation-wide ECB-controlled exit ATM1 installed at your site it is possible that if the time changes to 0000 before you re-enter the time to confirm the changes then ALCS will produce an unpredictable result. See *ALCS Installation and Customization* for a full explanation of ATM1.

DIF,[+|-]*hhmm*
Set the difference between local time and GMT to *hhmm* or -*hhmm* (complete with leading zeros). DIF is local time minus GMT. Specify *hhmm* if local time is in front of GMT; specify -*hhmm* if local time is behind GMT. That is, you need to specify a negative difference if you are West of the Greenwich Meridian.

ZATIM DIF changes local time, not GMT.

Normal response

```
DXC8648I CMD i hh.mm.ss  ATIM

            yyddd   yyyy.mm.dd   hh.mm.ss
ALCS local  yyddd   yyyy.mm.dd   hh.mm.ss   date
MVS  local  yyddd   yyyy.mm.dd   hh.mm.ss   date
ALCS GMT    yyddd   yyyy.mm.dd   hh.mm.ss   date
MVS  GMT    yyddd   yyyy.mm.dd   hh.mm.ss   date
```

Where:

MVS
MVS local time and GMT (system GMT is the time of day (TOD) clock time). MVS maintains these times; ZATIM cannot alter them.

ALCS
ALCS local time and GMT. These times can be the same as the system times or they can be different; ZATIM can alter them.

yyddd
Year, and day number in the year (the Julian date)

yyyy.mm.dd
Year, month and day

hh.mm.dd
Hours, minutes and seconds (24-hour clock format)

date
Date in the installation default format.

See *ALCS Installation and Customization* for further details on your installation default date format.

ZCICF -- Display information about the ICSF subsystem

Use the ZCICF command to provide information about the ICSF Subsystem. It ca be issued from any Prime CRAS authorized or alternate CRAS authorized terminal.

The format of the command is:

Where:

QALG
Displays information about the algorithms available in ICSF, CPACF and the crypto cards.

QICF
Displays status information about ICSF, CPACF and also the count of encrypt calls and decrypt calls as recorded by DXCICSF

Full details of the data returned are provided in the “Utilities – ICSF Query Algorithm” chapter of the *ICSF Application Programmer’s Guide*. A rule array count of zero is used.

Normal response to ZCICF QALG

```
DXC5491I CME i hh.mm.ss  CICF
ICSF Algorithm Information
Name      Size      Key/Sec   Impl.
name1     size1     desc1     impl1
name2     size2     desc2     impl2
  .         .         .         .
  .         .         .         .
  .         .         .         .
```

Normal response to ZCICF QICF

```
DXC5490I CME i hh.mm.ss  CICF
ICSF status
FMID               swlevel
Status field 1     a
Status field 2     b
CPACF              c
AES                d
DSA                e
RSA Signature      f
RSA Key Management g
RSA Key Generate   h
Accelerators       i

Count of Encryption calls cnt1
Count of Decryption calls cnt2
```

Where:

swlevel,a,b,c,d,e,f,g,h,i
Values returned from ICSF.

Full details of the data returned are provided in the “Utilities – ICSF Query Algorithm” chapter of the *ICSF Application Programmer’s Guide*. A rule array count of one is used — rule ICSFSTAT.

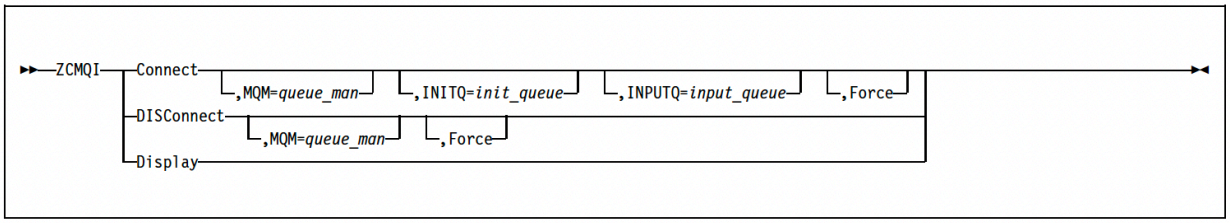
cnt1
The number of encryption calls made by applications.

cnt2
The number of decryption calls made by applications.

ZCMQI -- Control connection to WebSphere MQ for z/OS

This command is used to control ALCS connection to the WebSphere MQ for z/OS queue manager. You can display information from any CRAS authorized terminal but you can only connect and disconnect the queue manager from a Prime CRAS authorized terminal.

The format of the command is:



Where:

Connect
Connect to the WebSphere MQ for z/OS queue manager. The following three parameters are used to specify the characteristics of the WebSphere MQ for z/OS queue manager.

MQM=*queue_manager*
WebSphere MQ for z/OS queue manager name, 1 through 48 alphanumeric characters. Default is the name defined in the ALCS generation (SCTGEN MQMM parameter). Specify "MQM=" if you want to use the system-wide default queue manager instead of the ALCS default queue manager. The system-wide default is established as part of the WebSphere MQ for z/OS queue manager installation process.

INITQ=*init_queue*
WebSphere MQ for z/OS initiation queue name, 1 through 48 alphanumeric characters. Default is the name defined in the ALCS generation (SCTGEN MQMI parameter). Specify "INITQ=" if you do not want to use the default.

INPUTQ=*input_queue*
WebSphere MQ for z/OS input queue name, 1 through 48 alphanumeric characters. Default is the name defined in the ALCS generation (SCTGEN MQMQ parameter). Specify "INPUTQ=" if you do not want to use the default.

Force
Bypass the checks that ALCS normally makes before attempting to establish or terminate a connection.

DISConnect
Disconnect from the WebSphere MQ for z/OS queue manager. If you use:

```
ZCMQI DISConnect
```

then ALCS will disconnect from any WebSphere MQ for z/OS queue manager. If you use:

```
ZCMQI DISConnect,MQM=queue_manager
```

then ALCS will check that it is connected to the specified queue manager before disconnecting.

Display
This is allowed from any CRAS authorized terminal. Use it to display the characteristics of the WebSphere MQ for z/OS queue manager with which ALCS is connected. The resulting display includes:

```
WebSphere MQ for z/OS queue manager name.
```

```
Name of the initiation queue (if one is being used).
Name of the input queue (if one is being used).
```

Normal responses

Normal response to ZCMQI DISPLAY

```
DXC8625I CMD i hh.mm.ss CMQI
Currently connected to WebSphere MQ for z/OS
Queue manager      queue_manager
Initiation queue   init_queue
Input queue        input_queue
```

Normal response to ZCMQI CONNECT

```
DXC8626I CMD i hh.mm.ss CMQI
Now connected to WebSphere MQ for z/OS
Queue manager      queue_manager
Initiation queue   init_queue
Input queue        input_queue
```

Normal response to ZCMQI DISCONNECT

```
DXC8627I CMD i hh.mm.ss CMQI
Now disconnected from WebSphere MQ for z/OS
Queue manager      queue_manager
```

Where:

queue_manager
WebSphere MQ for z/OS queue manager name.

init_queue
WebSphere MQ for z/OS initiation queue name.

input_queue
WebSphere MQ for z/OS input queue name.

ZCMSP -- Control 3270 mapping

Use the ZCMSP command to control the mapping support for all the IBM 3270 terminals in your installation.

Screen mapping provides the user with a ready made screen display. The whole of a transaction can be entered on one display by filling in the blanks, instead of entering the data as a sequence of messages.

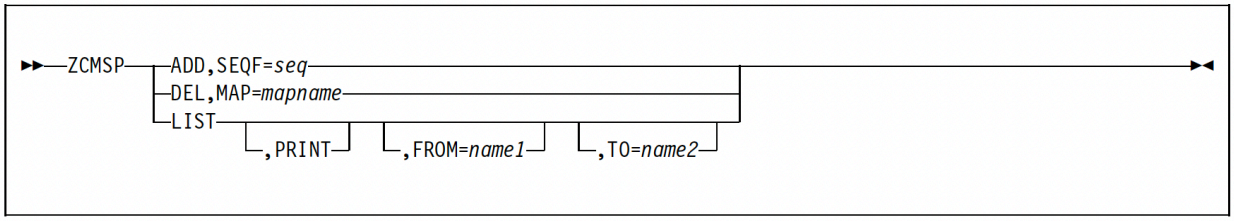
The application programmer codes a map description to define the screen layout, using a sequence of MAP3270 macroinstructions. See *ALCS Application Programming Guide* for details.

Your system programmer uses these map descriptions to build maps on a sequential file.

Use ZCMSP -- ***from a Prime CRAS authorized terminal only*** -- to load maps to the real-time database so that application programs can use them, or to delete a map from the database.

Use ZCMSP -- ***from any CRAS authorized terminal*** -- to display or print the names of the maps stored on the database.

The format of the command is:



Where:

ADD,SEQF=*seq*
Loads maps to the database. *seq* is the name of the sequential file that contains the list of maps.

DEL,MAP=*mapname*
Delete the map with the name *mapname* from the ALCS database.

LIST [,PRINT] [,FROM=*name1*] [,TO=*name2*]
List the names of the maps that ALCS has stored on the database.

PRINT
Print the list on the associated printer or on the RO CRAS if there is no associated printer.

FROM=*name1*
List maps starting with the map name *name1* (or the next map name in EBCDIC collating sequence).

TO=*name2*
List maps ending with the map name *name2* (or the previous map name in EBCDIC collating sequence).

If you specify FROM and TO, ALCS displays all map names between and including *name1* and *name2*.

ZCSEQ -- Close a general sequential file

Use the ZCSEQ command -- ***from any CRAS authorized terminal*** -- to close a general sequential file that is open but not in use.

Note: This status (called RESERVE, see "ZDSEQ -- Display sequential file status" on page 194) is not necessarily an error. Before you close the file, consult your system programmer to make sure that the condition is an error.

The format of the command is:

Where:

seq
Symbolic name of the general sequential file.

Force
Unconditionally close and unallocate a sequential file.

Restriction

ZCSEQ can only close general sequential files. To close the active data set of a real-time or system sequential file (for example, so that an offline program can process it), see "ZSSEQ -- Switch sequential file" on page 295.

You can only use the `Force` parameter from a Prime CRAS authorized terminal. When you use the `Force` parameter any process which is using the sequential file will terminate abnormally.

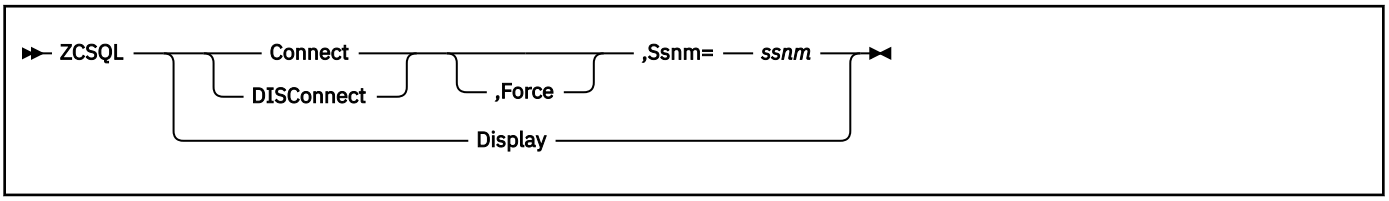
This parameter requires confirmation.

ZCSQL -- Control the connection between ALCS and DB2

Use the ZCSQL command -- ***from a Prime CRAS authorized terminal only*** -- to establish or terminate a connection between ALCS and a specified DB2 subsystem in the same MVS system.

You can also use ZCSQL -- ***from any CRAS authorized terminal*** -- to display the current connection (if any) between an ALCS and a DB2 system.

The format of the command is:



Where:

Connect
Connect ALCS to the DB2 subsystem specified by the Ssnm= parameter.

ALCS normally checks to see if it is already connected to DB2. It omits this check if you specify the Force parameter.

If you issue ZCSQL `Connect` when DB2 is not started, and DB2RECONNECT=YES is specified on the SCTGEN generation macro, then ALCS will automatically try to connect again when DB2 starts.

DISConnect
Terminate a connection to the specified DB2 subsystem. ALCS can be connected to only one DB2 subsystem at any one time; the Ssnm= parameter confirms which DB2 subsystem is being disconnected.

Before terminating a connection, ALCS normally checks whether:

- It is already connected to DB2
- The name specified by Ssnm= is the name of the system to which ALCS is currently connected
- Any entry has issued an SQL call and is waiting for a response from DB2.

ALCS omits these checks when you specify the Force parameter.

If you previously issued ZCSQL `Connect` when DB2 was not started, and DB2RECONNECT=YES is specified on the SCTGEN generation macro, then ZCSQL `DISConnect` stops ALCS from automatically trying to connect again when DB2 starts.

Ssnm=*ssnm*
Name of the DB2 system to be connected or disconnected (1 to 4 alphanumeric characters). Names that are less than 4 characters are padded on the right with blanks to 4 characters.

Force
Bypass the normal checks before establishing or terminating a connection to DB2.

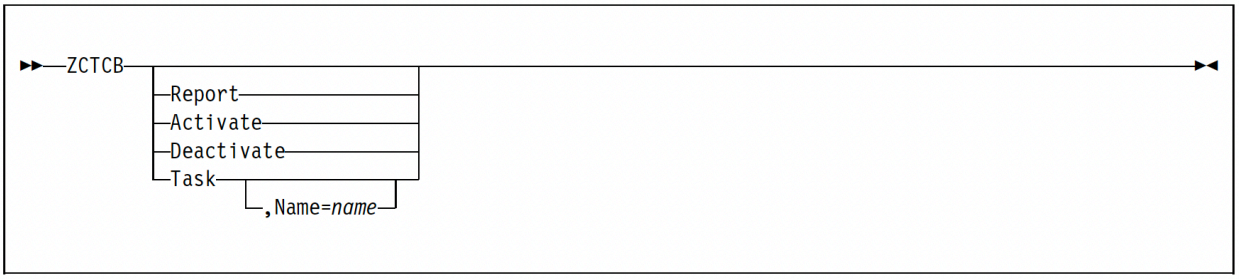
Display
Display the DB2 subsystem (if any) currently connected to ALCS.

ZCTCB -- Control CPU loops and display CPU loops or MVS tasks

Use the ZCTCB command - ***from a Prime CRAS authorized terminal only*** - to control the number of active ALCS CPU loops.

Use the ZCTCB command - ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal only*** - to obtain information about the ALCS CPU loops or MVS tasks used by ALCS.

The format of the command is:



Where:

Report
Obtain information about ALCS CPU loops (this is the default).

Activate
Activate an ALCS CPU loop.

Deactivate
Deactivate an ALCS CPU loop.

Task
Obtain information about MVS tasks used by ALCS.

Task,Name=*name*
Obtain information about MVS tasks used by ALCS with this task name. This is the module name from the ZCTCB TASK display.

Normal responses

Normal response to ZCTCB ACTIVATE

```
DXC5182I CME i  hh.mm.ss CTCB  One CPU loop activated
⋮
```

Normal response to ZCTCB DEACTIVATE

```
DXC5182I CME i  hh.mm.ss CTCB  One CPU loop deactivated
⋮
```

Normal response to ZCTCB REPORT

```
DXC5188I CME i  hh.mm.ss CTCB  CPU loop Status
EVCB       TCB        Save area  Perf table Status     Busy
aaaaaaaa   bbbbbbbb   cccccccc   dddddddd   eeeeeeee   fffff
⋮
```

Where:

aaaaaaaa
4-byte event control block (EVCB) for the CPU loop.

bbbbbbbb
Address of CPU loop TCB.

cccccccc
Address of CPU loop save area.

dddddddd
Address of ALCS performance monitor table if the performance monitor is enabled, otherwise zero.

eeeeeeee
Status of the CPU loop. The status is either `ACTIVE` or `INACTIVE`.

fffffff
Estimated CPU loop TCB busy percentage in format *nn.nn* if the performance monitor is enabled and activated, otherwise zero. This estimated percentage includes blocked I/O (test database) and interrupts.

Normal response to `ZCTCB TASK` or `ZCTCB TASK,NAME=`*name*

```
DXC5190I CME i  hh.mm.ss CTCB  ALCS tasks
TCB       RB        Module    EP        Lvl Grp CPU clock
aaaaaaaa  bbbbbbbb  cccccccc  dddddddd  e   f   gggggggggggggggg
⋮
```

Where:

aaaaaaaa
Address of task control block (TCB).

bbbbbbbb
Address of program request block (PRB).

cccccccc
Contents directory entry (CDE) module name.

dddddddd
Entry point address.

e
TCB level in a range from 0 to *n*. The level for the ALCS initializer TCB is 0 and the level for a CPU loop TCB is 1.

f
ALCS task identification:

1
ALCS initializer task

2
ALCS CPU loop task

3
ALCS sequential file task

4
ALCS MQSeries task

5
ALCS TCP/IP task

6
ALCS LU6.2 task

7
ALCS SQL task

8
ALCS NetView PPI task

9
ALCS SLC task

A
ALCS WAS task

gggggggg
ALCS CPU loop TCB accumulated CPU time in format *hhh:mm:ss.mmmuuu* (copied from TCB field TCBTTIME).

ZCTCP -- Control connection between ALCS and a TCP/IP address space

Use the ZCTCP command - ***from a Prime CRAS authorized terminal only*** -- to establish or terminate a connection between ALCS and a specified TCP/IP address space in the same MVS system.

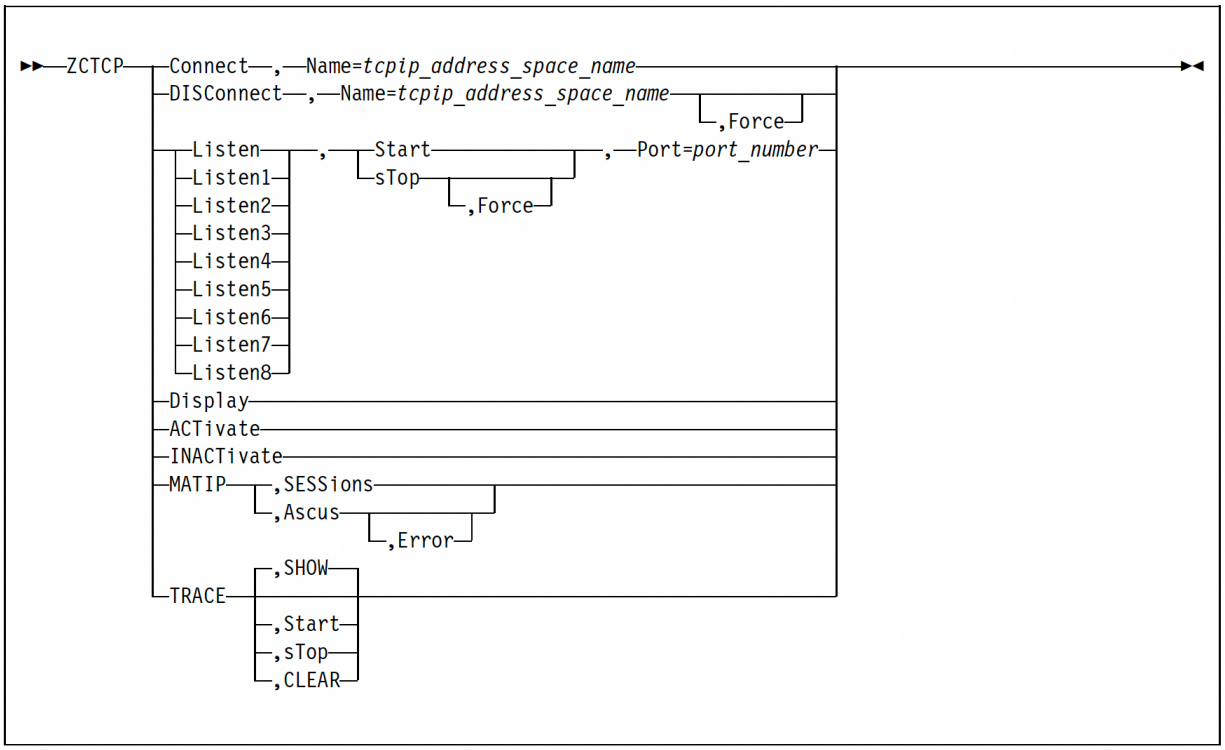
Use the ZCTCP command - ***from a Prime CRAS authorized terminal only*** -- to start or stop one of the ALCS TCP/IP concurrent servers (ALCS Listeners).

You can also use the ZCTCP command - ***from any CRAS authorized terminal*** - to display the current connection (if there is one) between ALCS and a TCP/IP address space.

Use the ZCTCP command - ***from a Prime CRAS authorized terminal*** or ***Alternate CRAS AT1-AT16 authorized terminal*** only to start and stop writing TCP/IP trace data to the diagnostic file, and to clear the system TCP/IP trace block.

Use the ZCTCP command - ***from any CRAS authorized terminal*** - to display the contents of the system TCP/IP trace block.

The format of the command is:



Where:

Connect
Connect ALCS to the TCP/IP address space specified by *tcpip_address_space_name*.

Force
Bypass the normal checks before terminating a connection to TCP/IP or stopping the concurrent server (Listener).

Note: Force stops all concurrent servers.

DISConnect
Terminate a connection to the specified TCP/IP address space. ALCS can be connected to only one TCP/IP address space at any one time; *tcpip_address_space_name* confirms which TCP/IP address space is being disconnected.

Name=*tcpip_address_space_name*
Name of the TCP/IP address space to be connected or disconnected. Specify the name of the TCP/IP job or started task. one to eight alphanumeric characters.

Listen,Start
Same as **Listen1,Start**.

Listen*n*,Start
Start one of the ALCS TCP/IP concurrent servers (Listeners). Up to eight concurrent servers can be started at the same time, each using a different port number. *n* is the index number of the concurrent server (1 to 8).

Listen,sTop
Same as **Listen1,sTop**.

Listen*n*,sTop
Stop one of the ALCS TCP/IP concurrent servers (Listeners). *n* is the index number of the concurrent server (1 to 8).

Port=*port_number*
The TCP/IP port number which the concurrent server (Listener) will use. A decimal number in the range 1 through 65 534; the default is 21.

Display
Display information about TCP/IP:

- The name of the TCP/IP address space (if there is one) that is currently connected to ALCS.
- The virtual IP address (if there is one) that ALCS uses.
- The status of the concurrent servers (Listeners).

ACTivate
Start all TCP/IP communication resources defined with `COMDEF ISTATUS=ACTIVE` in the ALCS communication generation.

INACTivate
Stop all TCP/IP communication resources.

MATIP,SESSions
Display all TCP/IP connections that currently have MATIP sessions.

MATIP,Ascus
Display all ASCUs (agent set control units) that are currently accessed through MATIP sessions.

MATIP,Ascus,Error
Display the ASCUS that are currently accessed through MATIP sessions and are in ERROR state

TRACE,SHOW
Display the contents of the system TCP/IP trace block. The format of the trace data is shown in “TCP/IP trace facility” on page 385.

TRACE,Start
Start TCP/IP tracing to the ALCS diagnostic file. You can run the ALCS diagnostic file processor later as a batch job.

TRACE,sTop
Stop TCP/IP tracing to the ALCS diagnostic file

TRACE,CLEAR
Clear the system TCP/IP trace block.

Normal responses

Normal response to ZCTCP with no parameters

```
DXC8825I CMD i hh.mm.ss CTCP
Currently connected to TCP/IP address space AS-'name'
Virtual IP address vipa_address
Listener n listener_status
⋮
```

Or:

```
DXC8826I CMD i hh.mm.ss CTCP
Not currently connected to TCP/IP
Virtual IP address vipa_address
```

Where:

name
Name of the TCP/IP address space which ALCS is connected to.

vipa_address
One of:

Dotted decimal IP address specified on the TCPVIPA parameter of the SCTGEN system generation macro.
`not defined` if TCPVIPA was not specified.

n
Index number of the concurrent server (1 to 8).

listener_status
One of:

`started on port PO-'number'`
`not started`

Normal response to `ZCTCP MATIP,SESSIONS`

```
DXC8999I CMD i hh.mm.ss CTCP  MATIP sessions
CRN      CRI    STATUS          TYPE CODE   MPX HDR ID
crn      cri    status          type code   mpx hdr id
⋮
```

Where:

crn
CRN of the resource.

cri
CRI of the resource.

status
Status of MATIP session, one of:

`CLOSED`
`STARTING`
`ACTIVE`
`RECONFIGURING`
`STOPPING`
`TERMINATING`

type
Type of MATIP session, one of:

A
Type A terminal-to-host session

I
Type A IATA host-to-host session

S
Type A SITA host-to-host session

B
Type B session

code
Translate code for the MATIP session, one of:

ALC
Padded ALC 6-bit

IATA7
ATA/IATA 7-bit (CCITT#5 ASCII)

IATA5
Padded ATA/IATA 5-bit (CCITT#2)

NONE
ALCS does not translate messages.

mpx
MATIP multiplex type for the session, one of:

10
Single ASCU, or
Host-to-host single flow

01
Group of ASCUs with 2 bytes identification per ASCU (A1+A2), or
Host-to-host multiple flows

00
Group of ASCU with 4 bytes identification per ASCU (H1+H2+A1+A2)

hdr
MATIP header type for the session, one of:

10
No header

01
ASCU header is A1+A2, or
Host-to-host header is flow ID

00
ASCU header is H1+H2+A1+A2, or
Host-to-host header is session ID + flow ID

id
MATIP ID for the session.

For Type A terminal-to-host sessions, this is the 2-character MATIP session ID.
For Type A host-to-host sessions, this is the 2-character MATIP session ID followed by the flow ID if there is one.
For Type B, this is the 2-character sender HLD followed by the 2-character recipient HLD.

Normal response to `ZCTCP MATIP,ASCUS`

```
DXC8999I CMD i hh.mm.ss CTCP  MATIP sessions
CRN      CRI    TYPE H1 H2 A1 A2 STATUS  REMOTE IP ADDRESS
crn      cri    type h1 h2 a1 a2 status  ip-addr
⋮
```

Where:

crn
CRN of the resource.

cri
CRI of the resource.

type
Type of MATIP session, one of:

A
Type A terminal-to-host session

I
Type A IATA host-to-host session

S
Type A SITA host-to-host session

B
Type B session

h1 h2
MATIP H1,H2 address values. The COMDEF HEX parameter corresponds to the MATIP H1,H2 address values.

a1
MATIP A1 address values. The COMDEF TCID parameter corresponds to the MATIP A1 address value.

a2
MATIP A2 address values. The COMDEF IA parameter corresponds to the MATIP A2 address value.

status
Status of ASCU, one of:

OK
ERROR

ip-addr
Remote IP address

Normal response to ZCTCP MATIP,Ascus,Error is the same as for ZCTCP MATIP,Ascus, except that STATUS is always ERROR.

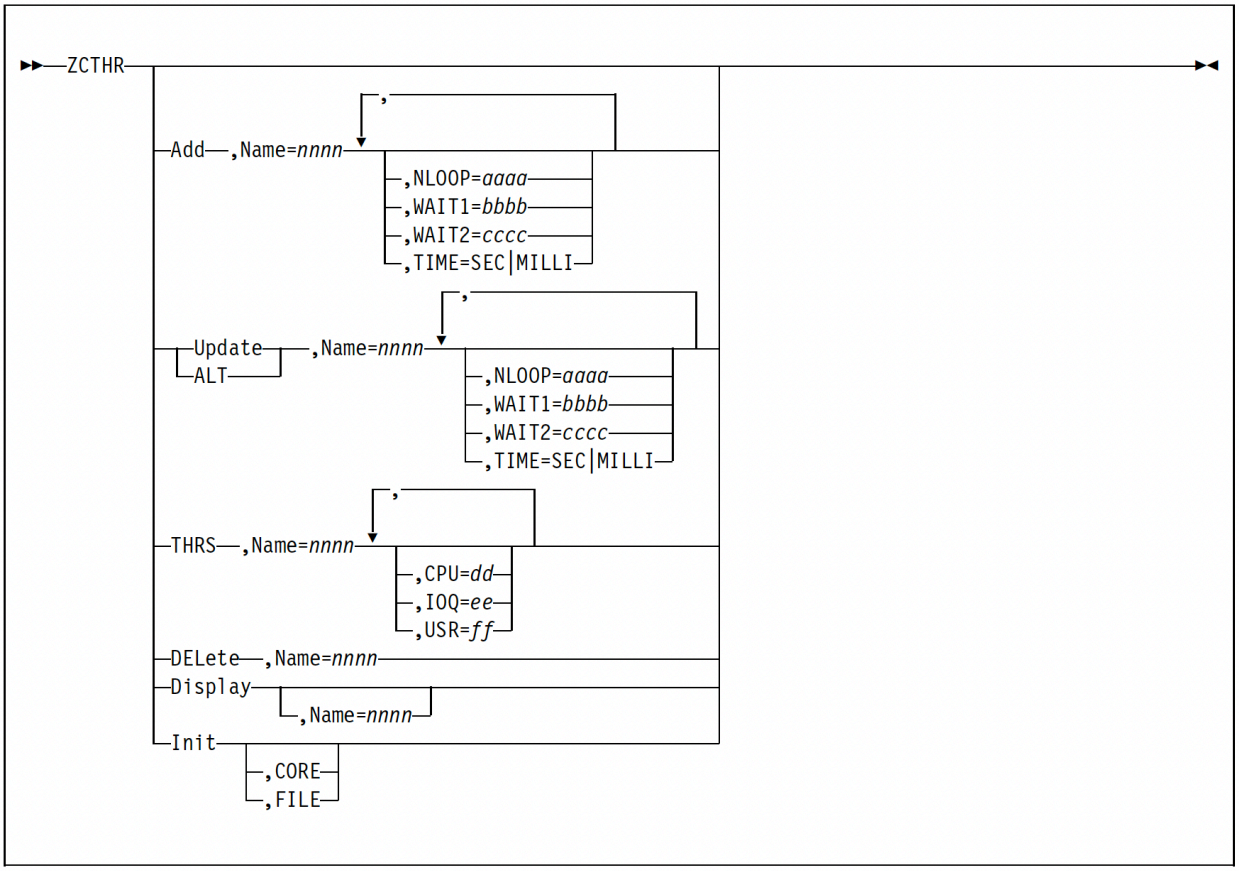
Internet RFC (Request for Comments) number 2351 describes the MATIP protocol.

ZCTHR -- Control and display the ALCS Throttle

Use the ZCTHR command - ***from a Prime CRAS authorized terminal only*** - to control the ALCS throttle.

Use the ZCTHR command - ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1-AT16 authorized terminal only*** - to display information about the ALCS throttle.

The format of the command is:



Where:

Add
Add throttle application.
This change is preserved over an ALCS restart.

DELete
Delete throttle application.
This change is preserved over an ALCS restart.

Update
Update normal values of a throttle application.
This change is preserved over an ALCS restart.

ALT
Update restricted values of a throttle application.
This change is preserved over an ALCS restart.

THRS
Update thresholds of a throttle application.
This change is preserved over an ALCS restart.

Display
Display throttle applications.

Init
Initialize throttle table.

CORE Initialize the table in memory only.
FILE Initialize the table on file and in memory.

Name=*nnnn*
Throttle application name of one to four characters.

NLOOP=*aaaa*
Normal and restricted NLOOP value. Specify a number in the range 0 through 9999.

WAIT1=*bbbb*
Normal and restricted WAIT1 value. Specify a number in the range 0 through 9999.

WAIT2=*cccc*
Normal and restricted WAIT2 value. Specify a number in the range 0 through 9999.

Time=SEC|MILLI
The units of the time to wait, where:

SEC
specifies wait time unit in seconds. It is the default.

MILLI
specifies wait time unit in milliseconds.

CPU=*dd*
CPU threshold. Specify a number in the range 0 through 99.

IOQ=*ee*
I/O queue time threshold. Specify a number in the range 0 through 99.

USR=*fff*
User threshold. Specify a number in the range 0 through 999.

For detailed information about NLOOP, WAIT1, WAIT2, CPU, IOQ, and USR refer to “ALCS Throttle” on page 397.

Normal responses

Normal response to ZCTHR Add:

```
DXC5095I CME i  hh.mm.ss  CTHR  Throttle Application added
```

Normal response to ZCTHR Delete:

```
DXC5096I CME i  hh.mm.ss  CTHR  Throttle Application deleted
```

Normal response to ZCTHR Update:

```
DXC5088I CME i  hh.mm.ss  CTHR  Throttle Application updated
```

Normal response to ZCTHR ALT:

```
DXC5088I CME i  hh.mm.ss CTHR  Throttle Application updated
```

Normal response to ZCTHR THRS:

```
DXC5088I CME i  hh.mm.ss CTHR  Throttle Application updated
```

Normal response to ZCTHR Init:

```
DXC5092I CME i  hh.mm.ss CTHR  Throttle Table initialized
```

Normal response to ZCTHR Display:

```
DXC5086I CME i  hh.mm.ss CTHR
           ----Normal-----------   --Threshold-  ------Restricted------
Appl Cond  Time Nloop Wait1 Wait2   Usr Cpu Ioq   Time Nloop Wait1 Wait2
nnnn cond  tttt aaaa  bbbb  cccc    fff dd  ee    tttt aaaa  bbbb  cccc
```

Where:

nnnn
Throttle application.

cond
Condition either NORM (NORMAL) or REST (RESTRICTED).

tttt
Wait time unit. Either SEC(seconds) or MSEC(milliseconds).

aaaa
NLOOP value.

bbbb
WAIT1 value.

cccc
WAIT2 value.

dd
CPU threshold.

ee
I/O queue time threshold.

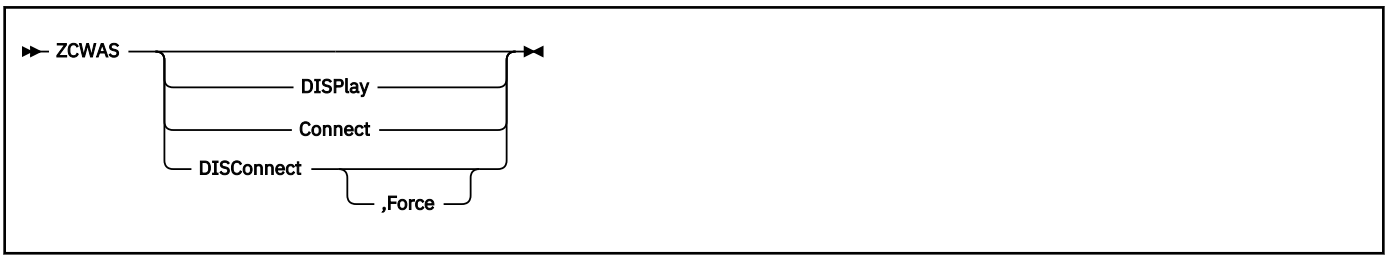
fff
User threshold.

ZCWAS -- Control connection between ALCS and WebSphere Application Server for z/OS (WAS)

Use the ZCWAS command -- ***from a Prime CRAS authorized terminal only*** -- to establish or terminate a connection between ALCS and WAS.

Use the ZCWAS command -- ***from any CRAS authorized terminal*** -- to display the status.

The format of the command is:



Where:

Connect
Connect ALCS and WebSphere Application Server for z/OS.

DISConnect
Disconnect ALCS after inactivating all active WAS resources. If threads are active in WAS OLA callable services, a disconnect command without the Force option does not complete.

Force
Forcibly disconnect ALCS even though threads are still active in WAS OLA callable services. The Force option may be used only after a disconnect command without the Force option has failed to complete.

DISPlay
Display status information. This is the default for ZCWAS with no additional parameters.

Normal responses

Normal response to ZCWAS DISPlay when connected:

```
DXC5250I CME i  hh.mm.ss  CWAS
WAS interface connected
Total  number of threads ...: td1
Active number of threads ...: td2
```

Normal response to ZCWAS DISPlay when disconnected:

```
DXC5251I CME i  hh.mm.ss  CWAS
WAS interface disconnected
Total  number of threads ...: td1
Active number of threads ...: td2
```

Normal response to ZCWAS DISPlay when disconnecting:

```
DXC5252I CME i  hh.mm.ss  CWAS
WAS interface disconnecting
Total  number of threads ...: td1
Active number of threads ...: td2
```

Where:

td1
Total number of threads (used by application programs issuing WAS OLA callable services).

td2
Active number of threads (used by application programs issuing WAS OLA callable services).

Normal response to ZCWAS Connect

```
DXC5255I CME i  hh.mm.ss  CWAS WAS interface now connected
```

Normal response to ZCWAS DISCONNECT

```
DXC5256I CME i  hh.mm.ss  CWAS WAS interface now disconnecting
```

Forcing a disconnection

This section illustrates the use of ZCWAS DISCONNECT,FORCE.

```
zcwas display
DXC5250I CME A  07.12.10 CWAS
WAS interface connected
Total  number of threads ...: 5
Active number of threads ...: 1
```

An active thread exists in the WAS OLA callable services. Issue a ZCWAS DISC to disconnect the thread.

```
zcwas disc
DXC5256I CME A  07.12.37 CWAS WAS interface now disconnecting
```

To illustrate the Force option assume this active thread does not terminate. This is shown by another ZCWAS DISPlay command.

```
DXC5252I CME A  07.13.02 CWAS
WAS interface disconnecting
Total  number of threads ...: 5
Active number of threads ...: 1
```

Issuing the ZCWAS DISCONNECT again but including the Force option finishes the disconnection of the active thread.

```
zcwas disc,f
DXC2020E DMP A  07.14.18
Check entry and retry -- If problem persists call supervisor
Problem reference information follows
SE-000082 CTL-000066 PROG-TSTD AT-7ED418FA CRN-PYEYTC33
VOLUME=KIPC06 DSNAME=XAN.V24.DIA.M0015000
MSG='WAS SUB TASK ABEND - ECB TERMINATED'
```

ZDACV -- Display activity control variables

Use the ZDACV command -- ***from any CRAS authorized terminal*** -- to display the current settings of the activity control variables. “ZAACV -- Alter activity control variables” on page 67 explains how to set these variables.

The format of the command is:

Normal response

```
DXC8586I CMD i hh.mm.ss DACV
Input max active entries ....(AV1) ........a
Create max total entries ....(AV2) ........b
Input min free storage units (AV3) ........c
Input min free I/O blocks ...(AV4) ........d
Recoup max creatable ECBs ...(AV5) ........e
Entry create high percentage (AV6) ........f
```

Where *a,b,c,d, e,* and *f* are the current values of the activity control variables.

ZDASD -- DASD data set functions

Use the ZDASD command -- ***from a Prime CRAS authorized terminal only*** -- to:

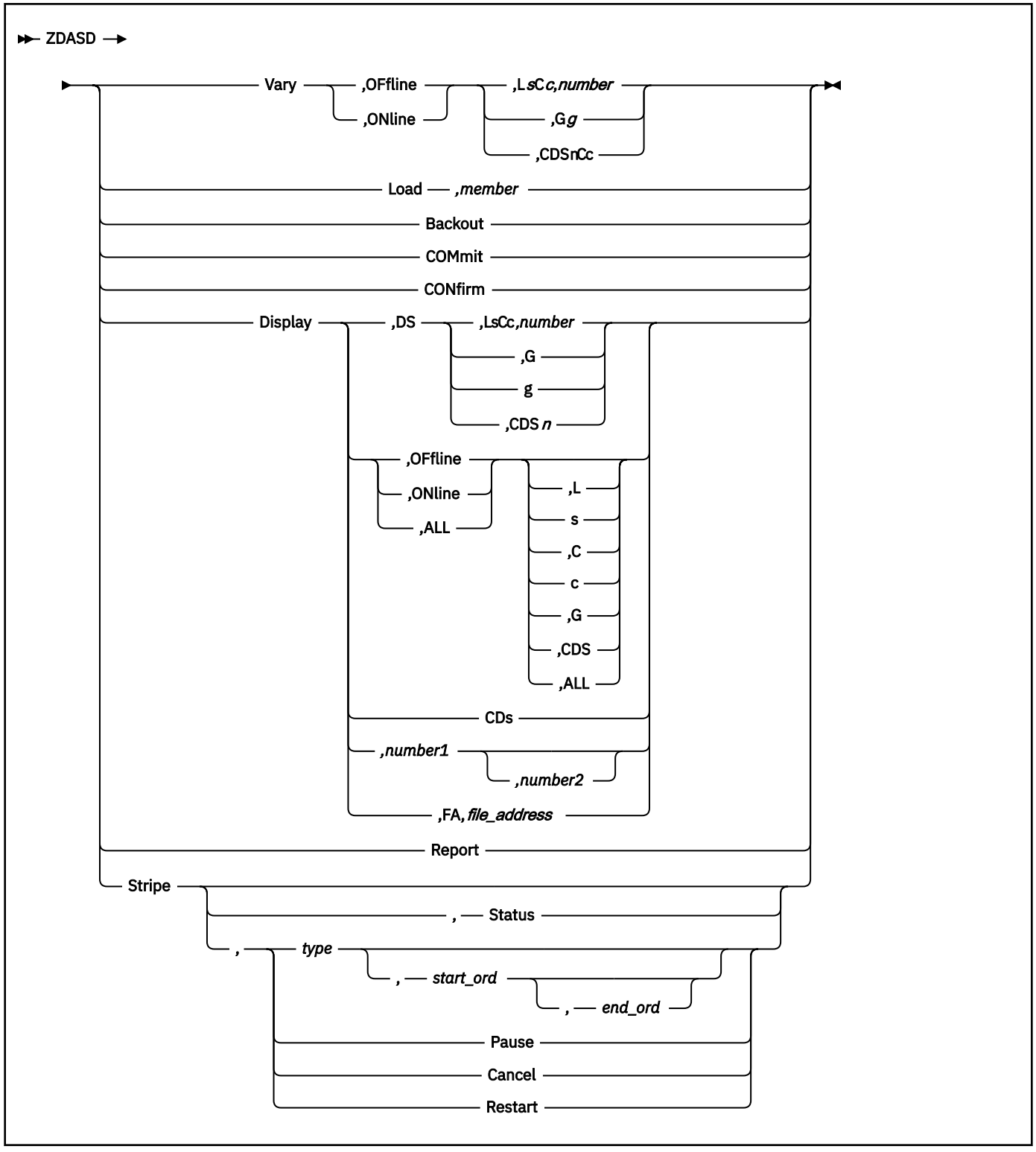
- Make a DASD data set available to ALCS.
- Make a DASD data set unavailable to ALCS.
- Load a DASD configuration table.
- Back out the current DASD configuration and revert to a previous one.
- Commit to the in-use DASD configuration table.
- Confirm changes to the DASD configuration.
- Distribute fixed-file or short-term pool records evenly across the available DASDs.

Use the ZDASD command -- ***from any CRAS authorized terminal*** -- to:

- Display information about one or more DASD data sets.

  The DASD data set can be a real-time database data set, a general file data set or a configuration data set.
- Display information about a DASD record.
- Produce reports on DASD configurations.

The format of the command is:



Where:

Vary,OFfline

Make a DASD data set unavailable to ALCS (deallocate the data set).

If the data set is a general file data set, ALCS accepts this command at any time, even if an application is using the data set. If an application is using the data set, this command can cause the application to fail. (This command does not directly affect offline programs that are using the data set.)

ALCS needs at least one copy of every real-time database data set and configuration data set, and does not accept the command if it makes the only copy of a data set unavailable. The command does not affect applications that are using the data set; they continue, using the other copy of the data set.

Vary,ONline
Make a DASD data set available to ALCS.

If it is a newly allocated data set and is not preformatted, ALCS formats the data set.

If the data set is a general file data set, it is available immediately after the formatting is complete.

If the data set is a real-time database data set, or a configuration data set, this command automatically initiates a copy. It copies all the records from the other (already online) copy of the data set. This ensures that the two copies of the data set contain identical information. The data set is not available until after the copy is complete.

Ls*Cc*,*number*
Real-time database data set, where Ls identifies the record size, C*c* identifies the data set copy (C1 for copy-1 or C2 for copy-2), and *number* identifies the data set number.

G*g*
General file data set, G0 for general file 0, and so on.

CDS*n*C*c*
Configuration data set. *n* is the configuration data set number (0, 1, or 2); *c* is the copy number (1 or 2).

Note: If you have two unequal sized data sets then you must ensure that the smaller one is varied online first.

Load,*member*
Load a DASD configuration table. *member* is the load module name of the DASD configuration table.

A ZDASD LOAD is permitted only if the previous ZDASD LOAD has been committed or backed out using the ZDASD COMMIT or ZDASD BACKOUT command.

Note: Following a system outage ALCS reverts to the previous load and the changes are lost.

Backout
Backout the currently in-use DASD configuration and revert to the previous one.

A ZDASD BACKOUT is only used after a ZDASD LOAD or a ZDASD CONFIRM command.

COMmit
Commit the currently in-use DASD configuration.

A ZDASD COMMIT is only used after a ZDASD CONFIRM.

Note: After you have used this command you cannot revert to a previous configuration with ZDASD BACKOUT.

CONfirm
Use this parameter to ensure that the changes you requested in a ZDASD LOAD command are secure over a system outage.

A ZDASD CONFIRM is only used after a ZDASD LOAD.

When the changes you requested in a ZDASD LOAD include a new DASD record size, use ZDASD VARY,ONLINE to vary online the new data sets before ZDASD CONFIRM.

Display,DS
Display the data set name, volume serial number, status, and other information about a data set. Specify the data set in the same way as for ZDASD VARY.

Display,{OFfline|ONline|ALL}
Display the data set name, volume serial number, and status information about one or more unavailable (OFfline), available (ONline), or unavailable and available (ALL) data sets. The default display includes all real-time database data sets.

L*s*
Restrict the display to real-time database data sets of this record size.

C*c*
Restrict the display to real-time database data sets of this copy.

G
Display information about general file data sets.

CDS
Display information about configuration data sets.

ALL
Display information about all data sets (real-time, configuration, and general file).

Display,*number1*[,*number2*]
Display the data set name, volume serial number, and status information about real-time database data sets with data set numbers in the range *number1* to *number2*. *number1* specifies the starting, or only, data set number; *number2* specifies the ending data set number in the range.

Display,CDs
Display the data set name, volume serial number and status information of the configuration data sets.

Display,FA,*file_address*
Display the data set name, relative record number within the data set, and other information about the DASD record that corresponds to the ALCS file address *file_address* (8 hexadecimal digits).

Use ZDFIL to display the file address that corresponds to a record type and record ordinal. Also use ZDFIL to display the contents of the record.

Report
Produce a report of the DASD configuration currently loaded.

Stripe,Status
Displays the status of any redistribution in progress.

Stripe,*type*
Starts to redistribute the fixed file or short-term pool file *type*.

Before you can use this command, all records in the file type must be table-based addressed. If this record type was part of your original generation it may still use algorithm-based addressing. See *ALCS Installation and Customization* for an explanation of how to use USRDTA ACTION=BUILD or DBHIST BUILD_DIRECTORIES to convert a file type to table-based addressing.

If you omit both *start_ord* and *end_ord* then ZDASD redistributes all records in the file.

If you specify *start_ord* (a decimal number), then ZDASD redistributes records starting with that ordinal number.

If you specify *end_ord* (a decimal number), then ZDASD redistributes records ending with that ordinal number.

Stripe,Pause
Pauses any redistribution in progress.

Stripe,Cancel
Cancels any redistribution in progress.

Stripe,Restart
Restarts redistribution that is paused by ZDASD STRIPE,PAUSE or by an ALCS outage.

Restrictions:

ZDASD VARY is not allowed if you are using a test data set. ZDASD LOAD, BACKOUT, CONFIRM or COMMIT are not allowed if other utilities such as Recoup are running. ZDASD STRIPE runs as a background activity even if other utilities are running. If ALCS terminates while ZDASD STRIPE is in progress it is automatically placed in pause status, and a warning message is output at restart time indicating that a ZDASD STRIPE,RESTART is required.

Normal responses

Normal responses to ZDASD VARY

```
DXC8765I CMD i hh.mm.ss DASD  Data set now off-line
```

The data set is no longer allocated to ALCS. The following message is sent to the RO CRAS:

```
DXC2756I CMD i hh.mm.ss DASD  Data set deallocated ZDASD VARY request,
               Volume VS-'volser',
               DSN-'dsname'
```

and message number DXC177I is sent to the MVS operator.

```
DXC8766I CMD i hh.mm.ss DASD  Data set VARY request accepted
```

The data set is allocated to ALCS. The following message is sent to the RO CRAS:

```
DXC2751I CMD i hh.mm.ss DASD  Data set allocated,
               Volume VS-'volser',
               DSN-'dsname'
```

and message number DXC167I is sent to the MVS operator. If the data set is a general file data set, it becomes available immediately.

If the data set is a real-time database data set it does not become available until the copy completes. When the copy completes ALCS also sends the following message to the RO CRAS:

```
DXC2752I CMD i hh.mm.ss DASD  Data set copy complete,
               Volume VS-'volser',
               DSN-'dsname'
```

It also sends message number DXC168I to the MVS operator.

See *ALCS Messages and Codes*, for details of messages DXC177I, DXC167I, and DXC168I.

Normal response to ZDASD DISPLAY,DS:

```
DXC8781I CMD i hh.mm.ss DASD
Data set name DSN--'dsname'
Volume.....volser
SIZE.............Ln
RECORDS.....records
ACTUAL.....actual
STATUS.....status
COPIED.....copied
```

Where:

dsname
Data set name of the data set. This is the data set name of the data component of the VSAM cluster.

volser
Volume serial number of the volume that contains the data set. The response does not include this line if ALCS cannot determine the volume serial; for example, if the data set is unavailable.

L*n*
Record size. The response does not include this line if ALCS cannot determine the record size; for example, if the data set is unavailable.

records
Number of records in the data set. The response does not include this line if ALCS cannot determine the number of records in the data set; for example, if the data set is unavailable.

actual
The precise number of records on a data set when this is more than the logical number of records shown in *records*.

This *actual* count appears in the repsonse only when two copies (copy-1 and copy-2) of the matching dataset are a different physical size. This *actual* count is the number of physical records in the larger copy.

status
Status of the data set, one of:

OFF
Applications cannot access the data set.

R/W
Applications can access the data set read/write.

RO
Applications can access the data set read-only.

WO
Applications can access the data set write-only.

NU
Applications cannot access the data set (the data set is not usable). This can be a temporary condition. It can occur when a data set has been varied offline during a formatting operation.

WP
Data set preformatting in progress.

WO status applies while ALCS is copying data to the real-time database data set for a ZDASD VARY,ONLINE command.

copied
The number of records copied to the data set so far. The response only includes this line while ALCS is copying data to a real-time database data set for a ZDASD VARY,ONLINE command.

Normal responses to

```
ZDASD DISPLAY,{OFfline|ONline|ALL}
ZDASD DISPLAY,number
ZDASD DISPLAY,number,number



DXC8782I CMD i hh.mm.ss DASD
Data-set-name                          Status  Volume
dsname                                 status  volser
⋮


DXC8782I CMD i hh.mm.ss DASD
                                       Status  Volume
NONE FOUND
```

Normal response to ZDASD DISPLAY,ALL:

```
DXC8782I CMD i hh.mm.ss DASD
Data-set-name                          Status  Volume
dsname                                 status  volser
dsname                                 status  volser
dsname                                 status  volser
```

Normal response to ZDASD DISPLAY,FA:

```
DXC8781I CMD i hh.mm.ss DASD
Data-set-name DSN--'dsname'
VOLUME.....volser
SIZE.............Ln
RRN.............rrn
RBA.............rba
```

Where:

dsname
Data set name of the data set that contains the record. If there are two copies of the data set, and if both copies are available, then this is the data set name of the copy-1 data set. Again, this the data set name of the data component of the VSAM cluster.

volser
Volume serial number of the volume that contains the data set. If there are two copies of the data set, and if both copies are available, then this is the volume serial number of the copy-1 data set.

L*n*
Record size.

rrn
Relative record number of the record within the data set.

rba
Relative byte address of the record within the data set.

Typical response to ZDASD REPORT

```
DXC8794I CMD i hh.mm.ss DASD  Member member Report
***COMMITTED***
comment
     Sequence......n  Event....event
comment
     Sequence......n  Event....event
comment
     Sequence......n  Event....event
***NOT YET CONFIRMED***
comment
     Sequence......n  Event....event
comment
     Sequence......n  Event....event
```

Where:

member
Member name of the DASD table in the library.

comment
User comment from the TEXT operand of the DBHIST macro.

Typical comments which your system programmer might include to indicate progress are:

- START NEW ST POOL
- INCREASE SIZE OF SHORT TERM

- START USING TYPE2 POOL

n
Sequence number of the DBHIST macro in the DASD generation.

event
Event type from the EVENT operand of the DBHIST macro.

*****COMMITTED*****
Changes associated with the DBHIST macroinstruction have been COMMITTED. It is not possible to BACKOUT these.

*****NOT YET CONFIRMED*****
Changes associated with the DBHIST macroinstruction have not been CONFIRMED. It is still possible to BACKOUT these changes.

*****NOT YET COMMITTED*** .**
Changes associated with the DBHIST macroinstruction have not been COMMITTED. In case of a planned or unplanned shutdown, these changes will not be lost. It is still possible to BACKOUT these changes.

*****BACKOUT IN PROGRESS*****
A BACKOUT is in progress for the changes associated with the DBHIST macroinstruction.

*****COMMIT IN PROGRESS *****
A COMMIT is in progress for the changes associated with the DBHIST macroinstruction.

For a full explanation of the DBHIST macro see *ALCS Installation and Customization*.

Normal response to ZDASD LOAD

```
DXC8790I CMD i hh.mm.ss DASD  Member member Loaded OK
comment
     Sequence......n  Event....event
comment
     Sequence......n  Event....event
```

Normal response to ZDASD BACKOUT

```
DXC8793I CMD i hh.mm.ss DASD  Member member Backout initiated
comment
     Sequence......n  Event....event
comment
     Sequence......n  Event....event

DXC2762I DAS i hh.mm.ss DASD  Backout complete
```

This response is sent to RO CRAS and does not appear on a display terminal.

Normal response to ZDASD CONFIRM

```
DXC8791I CMD i hh.mm.ss DASD  Member member Confirmed OK
comment
     Sequence......n  Event....event
comment
     Sequence......n  Event....event
```

Normal response to ZDASD COMMIT

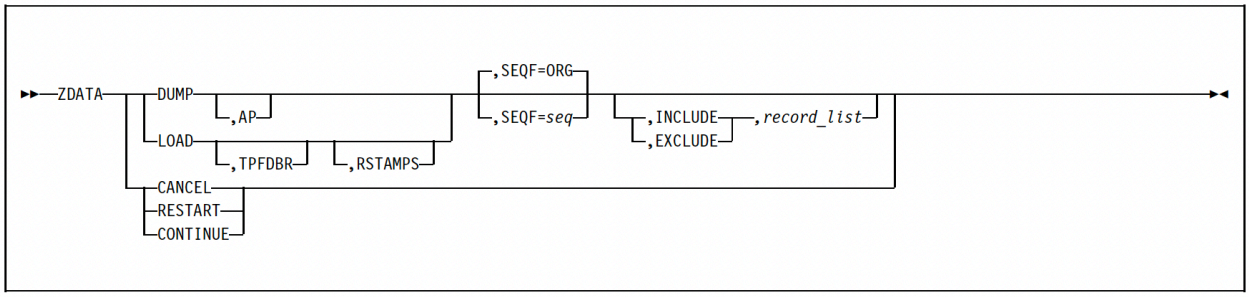
```
DXC8792I CMD i hh.mm.ss DASD  Member member Commit initiated
comment
     Sequence......n  Event....event
comment
     Sequence......n  Event....event


DXC2761I DAS i hh.mm.ss DASD  Commit complete
```

This response is sent to RO CRAS and does not appear on a display terminal.

ZDATA -- Load or dump DASD records

Use the ZDATA command -- ***from a Prime CRAS authorized terminal only*** -- to load records from a sequential file, such as a data file, to the real-time database or general files; or to dump records from the real-time database and general files to a sequential file.

The format of the command is:



Where:

LOAD
Load records from sequential file to the real-time database and general files. If any of these records are #GLOBL records, ALCS reloads the global area.

Note: ZDATA LOAD will not load records of type #KPTRI, or records from general file 0 (the Recoup general file).

For ZDATA LOAD, ALCS must be in IDLE state. No other utilities can be active when executing ZDATA LOAD or ZDATA DUMP.

DUMP
Dump records from the real-time database or general files to sequential file.

TPFDBR
Required only when migrating from TPF, this option controls the unblocking of the tape created by the TPF database reorganization facility.

RSTAMPS
When using ZDATA LOAD to read a tape produced by a pre-VFA ALCS/VSE ZRORG OUT command add the operand RSTAMPS to the end of the message.

The additional operand RSTAMPS is used to tell ALCS to reposition the time stamps in the long-term pool records. This is needed because the displacement of pre-VFA ALCS/VSE time stamps is slightly different from the displacement of ALCS V2 time stamps.

AP
Dump records from the real-time database with their allocatable-pool file addresses. Default if AP is not specified is to dump records with their fixed-file or pool-file addresses.

SEQF={ORG|*seq*}
Symbolic name of the sequential file.

INCLUDE|EXCLUDE

INCLUDE
Load or dump only the records in *record_list*.

EXCLUDE
Do not load or dump the records in *record_list*.

Omit this parameter to load every record from the sequential file to the real-time database and general files, or to dump all real-time database records. To dump all general files, specify each general file in the *record_list* for INCLUDE.

record_list
A list of one or more record selections. A comma or space must separate each selection. A selection can be any of the following:

file_address
File address; 8 hexadecimal digits.

type
Any of the following:

GF-*nnn*
nnn is the general file number; 1 to 3 digits.

#xxxxx
Fixed record type.

LsLTpool
Long-term pool record, where L*s* identifies the record size.

LsSTpool
Short-term pool record, where L*s* identifies the record size.

***type*(*n*)**
Where *n* is the record ordinal (decimal).

***type*(*n*--*n*)**
Where *n*--*n* is a range of record ordinals (decimal).

type*--*id
Where *id* is the record identifier; 2 alphanumeric characters or 4 hexadecimal digits.

FIXED
All fixed file records.

POOL
All pool file records.

SIZEL*n*
All records of size L*n*.

CANCEL
Cancel the ZDATA function. If your installation has utilities that use the CAP1 program then you may use ZDATA CANCEL to cancel the utility.

See *ALCS Installation and Customization* for information on the CAP1 program.

{RESTART|CONTINUE}
Restart ZDATA LOAD after an ALCS failure.

ZDCLR -- Control data collection

Use the ZDCLR command -- ***from a Prime CRAS authorized terminal only*** -- to control the ALCS statistical data collection function. You can also use the ZDCLR command -- ***from any CRAS authorized terminal*** -- to display which data collection options are active.

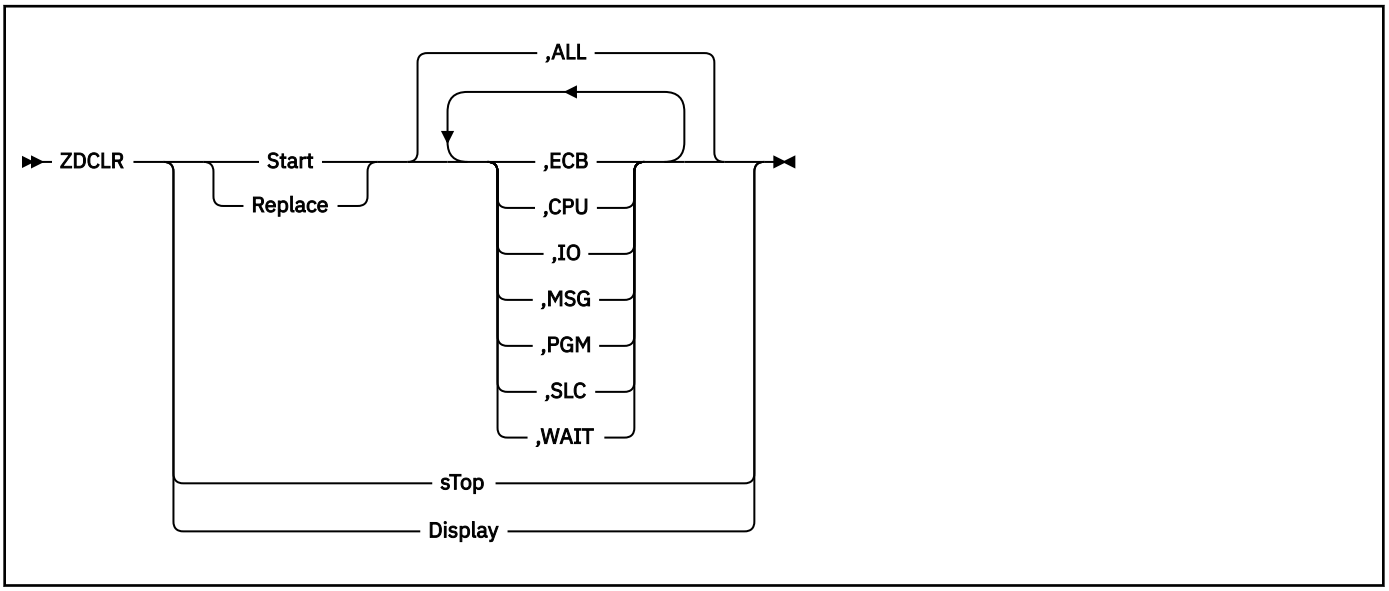
Your system programmer can specify options in the system generation macro to start data collection automatically on ALCS initialization. See the description of the SCTGEN macro in *ALCS Installation and Customization* for details.

Use ZDCLR to display the options specified in the system generation, or to alter them. Changes are not retained over an ALCS restart.

Attention

This command can produce a large quantity of output. Its use affects system performance, so use it with care.

The format of the command is:



Where:

Start
Start data collection.

Replace
Replace the current data collection options.

ALL
Equivalent to specifying all the collection options.

ECB
Collect statistics about each ECB. This option starts automatically when any of the other data collection options (except CPU) are specified. Specify ECB with no other options to restrict data collection to statistics on ECBs only.

CPU
Store the CPU utilization with the statistics about each ECB. The CPU utilization is the total elapsed CPU time while the entry has control in ECB-controlled program code, monitor-request macros, or C functions.

- Total elapsed CPU time does not include CPU time when the entry loses control. For example, time elasped during DASD I/O, sequential file I/O, or communication I/O is not included.

- This option does not start the ECB option automatically. If collection is needed the ECB option must be entered as well. If not entered, the CPU utilization information is only stored in the ECB descriptor.

IO
Collect statistics about input/output operations.

MSG
Collect statistics about input and output messages, excluding SLC traffic.

PGM
Collect statistics about application program use.

SLC
Collect statistics about SLC traffic.

WAIT
Collect statistics about ALCS task waits.

sTop
Stop data collection.

Display
Show which data collection options are currently active.

The data-collection output is sent to the ALCS data-collection file, which is defined by your system programmer during system generation. If no data-collection file has been defined, the output is sent to the ALCS diagnostic file.

The ALCS diagnostic file can be processed later by the statistical report generator (SRG), or by other products such as the Service Level Reporter (SLR). For further information see “Running the ALCS statistical report generator” on page 36.

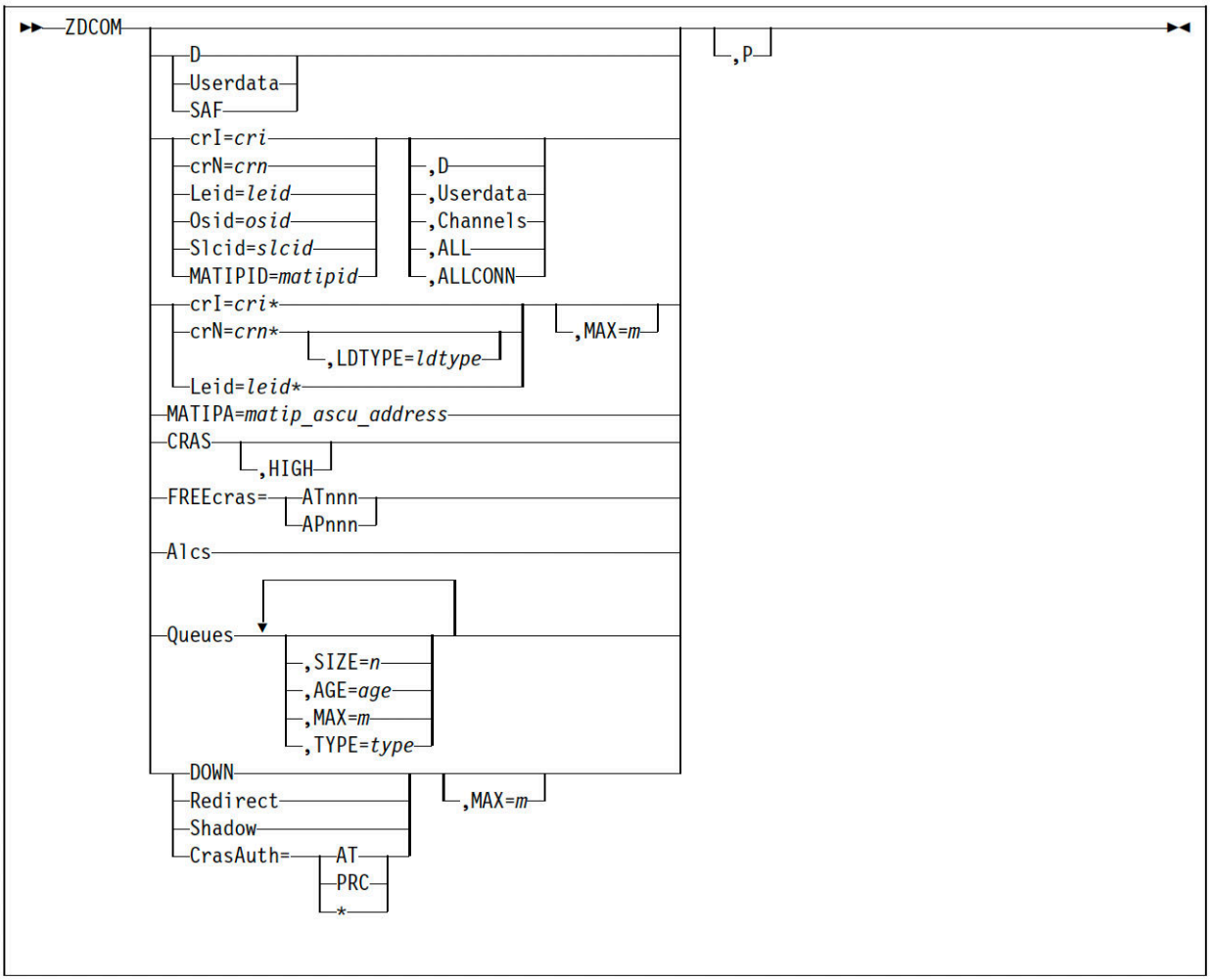
ZDCOM -- Display communication resource information

Use the ZDCOM command -- ***from any terminal*** -- to display:

- Information about a communication resource
- BATAP variables for AX.25 and MATIP Type B message processing
- Status of an ALCS printer
- Status of an LU 6.1 link
- Status of channels on an SLC link
- Information about message queues
- Information about unusable resources
- Information about redirected printers
- Information about shadow printing
- Terminals which have CRAS status or CRAS authority
- Active TCP/IP connections for a server resource
- ALCS communication system information

Use the ZDCOM command with no parameters -- ***from any terminal*** -- to display information about that terminal.

The format of the command is:



Where:

crI=*cri*
CRI of the resource (6 hexadecimal digits). CRIs starting 00 or 01 (the CRAS resources) are not valid; use the CRAS parameter instead.

crN=*crn*
CRN of the resource (1 to 8 alphanumeric characters).

Leid=*leid*
Logical end-point identifier (LEID) for terminals connected through an ALCI logical unit (LU). It is 1 to 6 hexadecimal digits. If you enter less than 6 hexadecimal digits, the value is padded on the left with hexadecimal zeros.

Osid=*osid*
Other system identifier used by another system to identify the resource. It is made up of the communication identifier (COMID; one alphabetic character) followed by the terminal identifier (CSID; 6 hexadecimal digits).

Slcid=*slcid*
A 16 hexadecimal digit SLC address for terminals connected through an SLC link.

The address is made up of the following concatenated data:

3-byte CRI of the owning SLC link
2-byte HLN exit address (HEX) of the terminal
1-byte HLN terminal circuit ID (TCID) of the terminal
1-byte interchange address (IA) of the terminal

1-byte terminal address (TA) of the terminal

MATIPID=*matipid*
A 10 hexadecimal digit MATIP address for terminals connected through a TCP/IP link using the MATIP Type A protocol.

The address is made up of the following concatenated data:

2-byte HLN exit address (HEX) of the terminal, or zeros
1-byte HLN terminal circuit ID (TCID) of the terminal
1-byte interchange address (IA) of the terminal
1-byte terminal address (TA) of the terminal

D
Display detailed information about a single resource. Enter D with no other parameters to display information about the originating terminal. Omit D to display only basic information.

Userdata
Display the communication user data for a resource. Enter `Userdata` with no other parameters to display user data about the originating terminal.

Channels
Display status information about the channels on an SLC link. This parameter is ignored for other types of resources.

ALL
Display all subordinate resources for an X25PVC, TCP/IP, MQ, or WAS resource.

ALLCONN
Display currently active connections for a TCP/IP server resource (`ZDCOM ALL` shows all active and inactive connections).

SAF
Display installation-defined data from the external security manager (ESM) connected GROUP profile for the end user that is currently logged on to the originating terminal.

crI=*cri**
A generic CRI of the resource; this is 2 through 5 hexadecimal digits with a trailing asterisk, for example 04* or 020A*. The command may require confirmation if many resources match the generic CRI. See “Confirmation of commands” on page 60.

crN=*crn**
A generic CRN of the resource; this is 1 through 8 characters which may include any number of wildcard characters (*, % or .). The wildcards may be placed in any position, for example ABC*, A%B* or *AB*. Use * to match any number of characters and use % or . to match a single character. The command may require confirmation if many resources match the generic CRN. See “Confirmation of commands” on page 60.

LDTYPE=*ldtype*
Use this parameter with a generic CRN. It specifies which type of resource to display in a generic search. *ldtype* is one of:

TERMINAL
A terminal resource

ALCSAPPL
An ALCS application resource

SLCLINK or WTTY
An SLC link or WTTY resource

ALCSLINK or APPC
An ALCS link or APPC resource

TCPIP
A TCP/IP resource

MQ
An MQ resource

WAS
A WAS resource

Omit LDTYPE to display all matching resources.

Leid=*leid
A generic LEID of the resource. This is 1 through 5 hexadecimal digits with a trailing asterisk, for example AA* or 0101*. The command may require confirmation if many resources match the generic LEID. See "Confirmation of commands" on page 60.

MATIPA=*matip_ascu_address*
An 8 hexadecimal digit MATIP ASCU address for terminals connected through a TCP/IP link using the MATIP Type A protocol.

The address is made up of the following concatenated data:

2-byte HLN exit address (HEX) of the terminals, or zeros
1-byte HLN terminal circuit ID (TCID) of the terminals
1-byte interchange address (IA) of the terminals

CRAS
Display basic information about all CRAS terminals. The CRASs are RO CRAS, Prime CRAS, the alternate CRAS displays AT1 to AT255, and the alternate CRAS printers AP1 to AP255.

CRAS,HIGH
Display basic information about RO CRAS, Prime CRAS, and the alternate CRAS displays AT1 to AT16.

FREEcras={AT*nnn*|AP*nnn*}
Display the number of the next alternate CRAS display (AT*nnn*) or printer (AP*nnn*) that is not already in use, starting from *nnn* and ending at 255.

Alcs
Display information about the current ALCS version and the communication system values.

Queues[,SIZE=*n*|,AGE=*age*|,MAX=*m*|,TYPE=*type*]
Display information about message queues for displays and printers.

Where:

SIZE=*n*
Display information about queues containing at least *n* messages. *n* can be any number up to 8 digits.

AGE={*mm*|M|*hh*|H|*dd*|D|*ww*W}
Display information about queues for which the oldest message has been in the queue for at least this length of time. You can specify the age in minutes (*mm*M), hours (*hh*H), days (*dd*D) or weeks (*ww*W). The number you specify in *mm*, *hh*,, *dd*, or *ww* can be any decimal number from 1 to 99.

TYPE=*type*
Display information about message queues for *type* resources where *type* is one of:

Printer
A printer device

Display
A display device

DOWN
Display information about all unusable resources.

Redirect
Display information about redirected printers.

Shadow
Display information about shadow printing.

CrasAuth={AT|PRC|*}
Display terminals which have CRAS authority, where:

AT
Display all terminals which have Alternate CRAS authority.

PRC
Display all terminals which have Prime CRAS authority.

*****
Display all terminals which have Alternate CRAS or Prime CRAS authority.

The command may require confirmation if many resources match the search argument. See “Confirmation of commands” on page 60.

MAX=*m*
This parameter is used to limit the display of a generic search, message queues, unusable resources, redirected printers, or CRAS authorities. Only the first *m* resources are displayed. It is a decimal value 1 through 5 000. Omit MAX to display the first 500 relevant resources.

P
Print the information on the printer that is associated with the originating terminal. If there is no associated printer, and the originating terminal is CRAS, then print the information on RO CRAS.

Normal responses (basic)

The normal response when D is not specified varies with resource type as follows:

Basic response for a display or printer:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS      Authority
type     crn      cri    ordinal  appl     status   cras  f   auth
```

Where:

type
Either the device type or the LDTYPE as specified by the ALCS communication generation. This is the basic display for the following resource types:

3270-DSP
3270 display.

3270-PRT
3270 printer.

ALCI-DSP
ALC display accessed through ALCI.

ALCI-PRT
ALC printer accessed through ALCI.

NETV-DSP
Netview display.

NETV-PRT
Netview printer.

OSYS-DSP
Display owned by another system.

OSYS-PRT
Printer owned by another system.

SLC-DSP
ALC display accessed through SLC.

SLC-PRT
ALC printer accessed through SLC.

MQ-DSP
ALC or 3270 display accessed through MQSeries.

MQ-PRT
ALC or 3270 printer accessed through MQSeries.

WAS-DSP
ALC or 3270 display accessed through WAS.

WAS-PRT
ALC or 3270 printer accessed through WAS.

TCP-DSP
ALC display accessed through TCP/IP.

TCP-PRT
ALC printer accessed through TCP/IP.

X25-DSP
ALC display accessed through X.25.

X25-PRT
ALC printer accessed through X.25.

****
Unrecognized device type. ALCS also displays this if the resource is a SNA device that has not been in session with ALCS.

crn
CRN of the resource.

cri
CRI of the resource.

ordinal
Communication ordinal number of the resource.

appl
CRN of the ALCS application that the resource is routed to. This is defined in the communication generation (see *ALCS Installation and Customization*). You can change this using the ZROUT or ZACOM command.

status
Status of the resource. One of:

ACTIVE
Available to send and receive messages.

INACTIVE
Unavailable.

STV
Test resource; it can only be used by the system test vehicle (STV).

cras
CRAS status, if any, of the resource. None, or one of:

PRC
Prime CRAS status.

ROC
RO CRAS status.

AT*nnn*
Alternate CRAS status, number *nnn*.

AP*nnn*
Alternate CRAS printer status, number *nnn*.

f
Indicates whether the resource is a CRAS fallback candidate.

auth
Additional CRAS authorities, may be one of:

PRC
Prime CRAS authority

AT*nnn*
Alternate CRAS authority, number *nnn*

PRC AT*nnn*
Prime and Alternate CRAS authority, number *nnn*

Basic response for an LU 6.1 link, ALCI LU, WTTY link, MQSeries, or WAS resource:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
type     crn      cri    ordinal  appl     status
```

Where:

type
Either the device type or the LDTYPE as specified by the ALCS communication generation. One of:

LU61BASE
LU 6.1 link.

ALCI-A
ALCI LU for which ALCS translates between ALC and EBCDIC.

ALCI-E
ALCI LU for which the ALCI feature translates between ALC and EBCDIC.

WTTY-FDX
WTTY duplex link.

WTTY-HDX
WTTY half-duplex link.

WTTY-SI
WTTY simplex input link.

WTTY-SO
WTTY simplex output link.

MQ
MQSeries resource.

WAS
WAS resource.

The other fields have the same meanings as previously described.

Basic response for an ALCS application:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
ALCSAPPL crn      cri    ordinal  prog     status
```

Where:

prog
The name of the input message editor program for this application.

The other fields have the same meanings as previously described.

Basic response for an X.25 permanent virtual circuit:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Type
X25PVC   crn      cri    ordinal  appl     status   t
```

Where:

t
Type of X.25 PVC as specified on the PRTCOL= parameter of the COMDEF generation macro.

The other fields have the same meanings as previously described.

Basic response for an SLC link:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Type  Hen
SLCLINK  crn      cri    ordinal  appl     status   ts    hen
```

Where:

ts
Type of SLC link as specified on the PRTCOL= parameter of the COMDEF generation macro.

hen
High level network address as specified on the HEN= parameter of the COMDEF generation macro.

The other fields have the same meanings as previously described.

Basic response for a virtual SLC link resource:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Real link
VSLCLINK crn      cri    ordinal  appl     status   crn2
```

Where:

crn2
CRN of the associated X.25 PVC or TCP/IP resource.

The other fields have the same meanings as previously described.

Basic response for an LU 6.1 parallel session:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Base link
LU61SESS crn      cri    ordinal  appl     status   crn2
```

Where:

crn2
CRN of the base LU 6.1 link.

The other fields have the same meanings as previously described.

Basic response for an APPC connection:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Type
APPC     crn      cri    ordinal  appl     status   tc
```

Where:

tc
Type of APPC connection as specified on the PRTCOL parameter of the COMDEF generation macro.

The other fields have the same meanings as previously described.

Basic response for a TCP/IP connection:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
TCPIP-C  crn      cri    ordinal  crn2     status


DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
TCPIP-S  crn      cri    ordinal  crn2     status


DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Base Server
TCPIP-D  crn      cri    ordinal  crn2     status   crn3
```

Where:

type
One of:

TCPIP-C
TCP/IP client connection.

TCPIP-S
TCP/IP server connection.

TCPIP-D
TCP/IP dynamic server connection.

crn3
For a TCP/IP dynamic server resource, this is the CRN of the owning TCP/IP server resource.

Status
One of:

ACTIVE
Available to send and receive messages.

INACTIVE
Unavailable.

STOPPING
In the process of becoming unavailable.

STARTING
In the process of becoming available.

The other fields have the same meanings as previously described.

Normal responses with parameters

To ZDCOM with a generic CRI or CRN parameter

The normal response when a generic CRI or CRN is specified is the same as multiple normal responses (basic).

Or, if no match is found:

```
DXC8901I CMD i  hh.mm.ss DCOM  No matching resources found
```

To ZDCOM with a generic LEID parameter

The normal response when a generic LEID is specified is similar to multiple normal responses (basic), but it includes the LEID for each resource.

Or, if no match is found:

```
DXC8901I CMD i  hh.mm.ss DCOM  No matching resources found
```

To ZDCOM with the CRAS parameter

The normal response is the same as multiple normal responses (basic).

To ZDCOM FREECRAS

The normal response when FREECRAS=*nnn*, or FREECRAS=AT*nnn*, is specified is one of:

```
DXC8908I CMD i  hh.mm.ss DCOM
Alternate CRAS number ATnnn is available


DXC8910W CMD i  hh.mm.ss DCOM
No available alternate CRAS number found
```

The normal response when FREECRAS=AP*nnn*, is specified is one of:

```
DXC8909I CMD i  hh.mm.ss DCOM
Alternate CRAS  number APnnn is available


DXC8911W CMD i  hh.mm.ss DCOM
No available alternate CRAS printer number found
```

To ZDCOM with the ALCS parameter:

```
DXC8907I CMD i  hh.mm.ss DCOM
ALCS product name ............. name
ALCS product number ........... nnnn-nnn
ALCS version level ............ n.n.n
ALCS VTAM application name .... acb_name
VTAM version level ............ v.r.m
ALCS communication ID ......... c
ALCS system ID ................ s
ALCS startup parameters:
  TCB count ................... t
  System table ................ sys
  Database table .............. das
  Communication list .......... com
  Sequential file table ....... seq
  Application program list .... pgm
  Parameter for USRINIT ....... usr
  Initial system state ........ i
```

Where:

name
The name of the product, for example ALCS V2.

nnnn-nnn
The IBM product number for this ALCS.

n.n.n
The version level of the ALCS you are using, for example 2.4.1.

acb_name
The VTAM ACB name for this ALCS. (This is sometimes called the VTAM application minor node name.)

v
The VTAM version number.

r
The VTAM release number.

m
The VTAM modification level.

c
Communication identifier (defined in the communication generation).

s
System identifier (defined in the ALCS generation).

t
Number of CPU loop TCBs, or TEST if test database is used. The format is one of:

nnn
nnn-mmm

where *nnn* is the number of active CPU loop TCBs and *mmm* is the maximum number of CPU loop TCBs.

sys
Name of the system configuration table.

das
Name of the database configuration table.

com
Name of the communication load list.

seq
Name of the sequential file configuration table.

pgm
Name of the program load list.

usr
User initialization parameter.

i
Initial system state (STANDBY, I or IDLE, C or CRAS, M or MESW, N or NORM).

To ZDCOM with the Userdata parameter:

```
DXC8905I CMD i  hh.mm.ss DCOM
User data for CRN-crn
ddddddddd ddddddddd ddddddddd ddddddddd  *ccccccccccccccc*
ddddddddd ........                       *cccc....        *
⋮
```

Where:

crn
CRN of the resource.

dd...dd
Storage contents (hexadecimal).

cc...cc
Storage contents (character).

To ZDCOM with the Channels parameter:

```
DXC8912I CMD i  hh.mm.ss DCOM  Information for SLC link CRN-crn
Channel       Primary S/C  Alt S/C  Opn Sta Proc Los Str Hld Tsx
1 - send      ppp u        aaa u    opn sta proc los str hld tsx
  - receive   ppp u        aaa u
2 - send      ppp u        aaa u    opn sta proc los str hld tsx
  - receive   ppp u        aaa u                                .
3 - send      ppp u        aaa u    opn sta proc los str hld tsx
  - receive   ppp u        aaa u
4 - send      ppp u        aaa u    opn sta proc los str hld tsx
  - receive   ppp u        aaa u
5 - send      ppp u        aaa u    opn sta proc los str hld tsx
  - receive   ppp u        aaa u
6 - send      ppp u        aaa u    opn sta proc los str hld tsx
  - receive   ppp u        aaa u
7 - send      ppp u        aaa u    opn sta proc los str hld tsx
  - receive   ppp u        aaa u
```

Where:

crn
CRN of the SLC link.

ppp
Primary subchannel address (blank if the channel is not defined for this SLC link).

aaa
Alternate subchannel address (blank if the channel is not defined for this SLC link, or if there is no alternate subchannel address).

u
Asterisk (*) if the primary or alternate subchannel address is currently in use (or was last used), otherwise blank.

opn
The channel is open (Yes or No).

sta
The channel is started (Yes or No).

proc
The procedure in progress for the channel. One of the following:

DAT
Sending data.

DWN
Channel down.

ENQ
Enquiry.

IDL
Idle.

NAK
Negative acknowledgement.

los
The channel is out of service (Yes or No).

str
The channel has received a STOP LCB (Yes or No).

hld
The channel output is held (Yes or No).

tsx
The channel has TSI exhaustion (Yes or No).

To ZDCOM with the Queues parameter:

```
DXC8913I CMD i  hh.mm.ss DCOM  Message Queues
Resource CRN      CRI    Q size   Q Age           Status Unusable
type     crn      cri    m        www ddd hhh mmm status su
⋮
```

Where:

m
The total number of messages on queue. (This is the total for all priorities.)

www ddd hhh mmm
The age of the oldest message on queue in weeks (*www*), days (*ddd*), hours (*hhh*), and minutes (*mmm*).

status
Status of the resource. One of:

ACTIVE
Available to send and receive messages.

INACTIVE
Unavailable.

STV
Test resource; it can only be used by the system test vehicle (STV).

su
The resource is unusable (Yes or No).

Messages that are more than 100 weeks old are displayed:

```
DXC8913I CMD i  hh.mm.ss DCOM  Message Queues
Resource CRN      CRI    Q size   Q Age           Status Unusable
type     crn      cri    m        **w **d **h **m status su
⋮
```

The other fields have the same meanings as previously described.

Or, if there are no resources with message queues that match:

```
DXC8913I CMD i  hh.mm.ss DCOM  Message Queues
No matching resources found
```

To ZDCOM with the Redirect parameter:

```
DXC8914I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Re-CRN   Re-status
type     crn      cri    re-crn   re-status
```

Where:

type
A printer resource.

re-crn
The CRN of the redirection printer.

re-status
Signifies whether redirection to the *re-crn* printer has been turned on (ACTIVE) or off (INACTIVE).

Or, if there are no printers that have been redirected:

```
DXC8901I CMD i  hh.mm.ss DCOM  No matching resources found
```

To ZDCOM with the Shadow parameter:

```
DXC8914I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    S-Status  S-CRN
type     crn      cri    s-status  s-crn
```

Where:

type
A printer resource.

s-crn
Up to 16 CRNs of the shadow printers.

s-status
Signifies whether shadow printing to the *s-crn* printer(s) has been turned on (ACTIVE) or off (INACTIVE).

Or, if there are no printers that have been shadowed:

```
DXC8901I CMD i  hh.mm.ss DCOM  No matching resources found
```

To ZDCOM with the CrasAuth parameter

The normal response is the same as multiple normal responses (basic).

To ZDCOM with the MATIPA parameter:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
type     crn      cri    ordinal  appl     status  matip_id
⋮
```

Where:

matip_id
MATIP address for the terminal, made up of:

2-byte HLN exit address (HEX)
1-byte HLN terminal circuit ID (TCID)
1-byte interchange address (IA)
1-byte terminal address (TA)

The other fields have the same meanings as previously described.

To ZDCOM with the SAF parameter

```
DXC5200I CME i  hh.mm.ss DCOM
SAF data for group name gname
ddddddddd ddddddddd ddddddddd ddddddddd  *ccccccccccccccc*
ddddddddd ........                       *cccc....        *
⋮
```

Where:

gname
Name of the ESM's connected GROUP profile.

dd...dd
Installation-defined data from the ESM (hexadecimal).

cc...cc
Installation-defined data from the ESM (character).

Normal response (detailed)

Normal responses when D is specified are as follows.

For a 3270 display terminal:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
3270-DSP crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Messages on queue       - n
Number of lines/rows    - l
Number of columns       - col
DBCS support            - dbcs
SNA between brackets    - sna
VTAM SEND queue         - v
Terminal/AAA hold       - aaa
Test modules            - tm
```

Where:

userid
The user ID of the user logged on to the resource or the default user ID defined in the communication generation for the resource. This line is omitted if either the resource is inactive or the resource has not logged on and has no default user ID defined.

crn1
crn1 of the associated resource. If there is no associated resource, the response is None.

n
The number of unsolicited messages on queue.

l
Screen size.

col
Screen size - number of columns.

dbcs
Double-byte character support (Yes or No) (defined in the communication generation).

sna
SNA between brackets status (Yes or No).

v
There are messages waiting for a previous VTAM SEND to complete (Yes or No).

aaa
Terminal hold is used (Yes or No).

tm
Test modules are used (Yes or No).

The other fields have the same meanings as previously described.

For a 3270 printer terminal:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
3270-PRT crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Messages on queue pri p - n
Printer buffer size     - pb
DBCS support            - dbcs
SNA between brackets    - sna
VTAM SEND queue         - v
Resource is unusable    - ru
System sends allowed    - ss
Message shadowing       - ms
Shadow CRNs             - crn2
Message re-direction    - mr
Re-direction CRN        - crn3
Waiting for ACK         - ack
Queue being processed   - q
Message completely sent - mc
Shared resource         - sr
```

Where:

n
The number of messages on queue with priority *p*. The highest priority is 0. The lowest priority is 15.

Priority
Use

0
SLMTC monitor-request macro

1-13
SENDC L monitor-request macro

14
ALCS printer shadowing facility

15
ALCS unsolicited message facility

pb
Printer buffer size (defined in the communication generation).

dbcs
DBCS support (Yes or No).

sna
SNA between brackets status (Yes or No).

v
There are messages waiting for a previous VTAM SEND to complete (Yes or No).

ru
The resource is unusable (Yes or No).

ss
System sends are allowed (Yes or No).

ms
Printer shadowing is in effect (Yes or No).

crn2
CRN of the shadow printer or printers. Up to 16 can be specified. If there are no shadow printers, the response is None.

mr
Messages are redirected to another printer (Yes or No).

crn3
CRN of the redirection printer. (Only one can be specified.)

ack
Waiting for acknowledgement (Yes or No).

q
Queue being processed (Yes or No).

mc
Message completely sent (Yes or No).

sr
The resource is shared (Yes or No).

The other fields have the same meanings as previously described.

For an ALC display accessed through ALCI:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
ALCI-DSP crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Messages on queue       - n
Number of lines/rows    - l
Terminal/AAA hold       - aaa
Test modules            - tm
CRN of owning LU        - crn4
LEID                    - leid
```

Where:

crn4
CRN of owning the ALCI LU.

leid
Logical end point identifier.

csid
Other system terminal identifier (if there is one).

The other fields have the same meanings as previously described.

For an ALC printer accessed through ALCI:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
ALCI-PRT crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Messages on queue pri p - n
Printer buffer size    - pb
Timeout 1 - seconds    - t1
Timeout 2 - seconds    - t2
Timeout repetitions    - tr
Resource is unusable   - ru
System sends allowed   - ss
Message shadowing      - ms
Shadow CRNs            - crn2
Message re-direction   - mr
Re-direction CRN       - crn3
Waiting for ACK        - ack
Queue being processed  - q
Message completely sent - mc
Shared resource        - sr
CRN of owning LU       - crn4
LEID                   - leid
```

Where:

t1
Timeout 1 (defined in the communication generation).

t2
Timeout 2 (defined in the communication generation).

tr
Timeout repetition (defined in the communication generation).

See *ALCS Installation and Customization* for a full description of these parameters.

The other fields have the same meanings as previously described.

For an ALCI LU:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
ALCI-A   crn      cri    ordinal  appl     status
VTAM SEND queue        - v
```

or:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
ALCI-E   crn      cri    ordinal  appl     status
VTAM SEND queue        - v
```

The fields have the same meanings as previously described.

For an ALC display accessed through MQSeries:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
MQ-DSP   crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Device type            - ALC
Messages on queue      - n
Number of lines/rows   - l
Terminal/AAA hold      - aaa
Test modules           - tm
CRN of owning MQ queue  - crn8
```

For an ALC printer accessed through MQSeries:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
MQ-PRT   crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Device type            - ALC
Messages on queue pri p - n
Printer buffer size    - pb
Timeout 1 - seconds    - t1
Timeout 2 - seconds    - t2
Timeout repetitions    - tr
Resource is unusable   - u
System sends allowed   - ss
Message shadowing      - ms
Shadow CRNs            - crn2
Message re-direction   - mr
Re-direction CRN       - crn3
Waiting for ACK        - ack
Queue being processed  - q
Message completely sent - mc
Shared resource        - sr
CRN of owning MQ queue  - crn8
```

For a 3270 display accessed through MQSeries:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
MQ-DSP   crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Device type            - 3270
Messages on queue      - n
Number of lines/rows   - l
Number of columns      - col
DBCS support           - dbcs
Terminal/AAA hold      - aaa
Test modules           - tm
CRN of owning MQ queue  - crn8
```

For a 3270 printer accessed through MQSeries:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
MQ-PRT   crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Device type            - 3270
Messages on queue pri p - n
Printer buffer size    - pb
DBCS support           - dbcs
Timeout 1 - seconds    - t1
Timeout 2 - seconds    - t2
Timeout repetitions    - tr
Resource is unusable   - u
System sends allowed   - ss
Message shadowing      - ms
Shadow CRNs            - crn2
Message re-direction   - mr
Re-direction CRN       - crn3
Waiting for ACK        - ack
Queue being processed  - q
Message completely sent - mc
Shared resource        - sr
CRN of owning MQ queue  - crn8
```

For an MQSeries resource:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
MQ       crn      cri    ordinal  appl     status
Request queue name      - q1
Response queue name     - q2
Service queue name      - q3
Request queue trigger control - tc
Request queue trigger type - tt
```

Where:

crn8
Name of the owning MQ resource.

q1
Name of the MQSeries request queue.

q2
Name of the MQSeries response queue.

q3
Name of the MQSeries service queue.

tc
MQSeries trigger control for the request queue when this MQ resource was started, one of:

OFF
ON

tt
MQSeries trigger type for the request queue when this MQ resource was started, one of:

FIRST
EVERY
DEPTH
NONE

The other fields have the same meanings as previously described.

For an ALC display accessed through WAS:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
WAS-DSP  crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Device type            - ALC
Messages on queue      - n
Number of lines/rows   - l
Terminal/AAA hold      - aaa
Test modules           - tm
Scrolling Inhibited    - si
CRN of owning resource - crn8
```

si
Scrolling inhibited (YES or NO).

The other fields have the same meanings as previously described.

For an ALC printer accessed through WAS:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
WAS-PRT  crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Device type            - ALC
Messages on queue pri p - n
Printer buffer size    - pb
Timeout 1 - seconds    - t1
Timeout 2 - seconds    - t2
Timeout repetitions    - tr
Resource is unusable   - u
System sends allowed   - ss
Message shadowing      - ms
Shadow CRNs            - crn2
Message re-direction   - mr
Re-direction CRN       - crn3
Waiting for ACK        - ack
Queue being processed  - q
Message completely sent - mc
Shared resource        - sr
CRN of owning resource - crn8
```

For a 3270 display accessed through WAS:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
WAS-DSP  crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Device type            - 3270
Messages on queue      - n
Number of lines/rows   - l
Number of columns      - col
DBCS support           - dbcs
Terminal/AAA hold      - aaa
Test modules           - tm
Scrolling Inhibited    - si
CRN of owning resource - crn8
```

si
Scrolling inhibited (YES or NO).

The other fields have the same meanings as previously described.

For a 3270 printer accessed through WAS:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
WAS-PRT  crn      cri    ordinal  appl     status   cras   f   auth
User ID               - userid
Associated resource CRN - crn1
Device type           - 3270
Messages on queue pri p - n
Printer buffer size   - pb
DBCS support          - dbcs
Timeout 1 - seconds   - t1
Timeout 2 - seconds   - t2
Timeout repetitions   - tr
Resource is unusable  - u
System sends allowed  - ss
Message shadowing     - ms
Shadow CRNs           - crn2
Message re-direction  - mr
Re-direction CRN      - crn3
Waiting for ACK       - ack
Queue being processed - q
Message completely sent - mc
Shared resource       - sr
CRN of owning resource  - crn8
```

For a WAS resource:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
WAS      crn      cri    ordinal  appl     status
Daemon group name       - w1
Node name               - w2
Server name             - w3
Register name           - w4
Service name 1          - w5
Service name 2          - w6
WAS protocol type       - w7
Connections             - w8/w9
```

Where:

w1
Name of the WebSphere Application Server for z/OS Daemon group

w2
Name of the WebSphere Application Server for z/OS node

w3
Name of the WebSphere Application Server for z/OS server

w4
Register name

w5
Name of the inbound (WAS to ALCS) service

w6
Name of the outbound (ALCS to WAS) service

w7
WAS protocol type, either TYPE1 (default) or TYPE2

w8
The number of concurrent receive/response connections allowed for this WAS resource

w9
The number of concurrent send connections allowed for this WAS resource

The other fields have the same meanings as previously described.

For a Netview display terminal:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
NETV-DSP crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Messages on queue       - n
Number of lines/rows    - l
Terminal/AAA hold       - aaa
Test modules            - tm
```

Where the fields have the same meanings as previously described.

For a NetView printer terminal:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
NETV-PRT crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Messages on queue pri p - n
Printer buffer size     - pb
Timeout 1 - seconds     - t1
Timeout 2 - seconds     - t2
Timeout repetitions     - tr
Resource is unusable    - ru
Shared resource         - sr
System sends allowed    - ss
Message shadowing       - ms
Shadow CRNs             - crn2
Message re-direction    - mr
Re-direction CRN        - crn3
Waiting for ACK         - ack
Queue being processed   - q
Message completely sent - mc
```

Where the fields have the same meanings as previously described.

For a display owned by another system:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
OSYS-DSP crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Messages on queue       - n
Number of lines/rows    - l
Terminal/AAA hold       - aaa
Test modules            - tm
Terminal ID - CSID      - csid
Remote system COMID     - c
```

Where:

c
Remote system communication identifier.

The other fields have the same meanings as previously described.

For a printer owned by another system:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
OSYS-PRT crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Messages on queue pri p - n
Printer buffer size     - pb
Timeout 1 - seconds     - t1
Timeout 2 - seconds     - t2
Timeout repetitions     - tr
Resource is unusable    - u
System sends allowed    - ss
Message shadowing       - ms
Shadow CRNs             - crn2
Message re-direction    - mr
Re-direction CRN        - crn3
Waiting for ACK         - ack
Queue being processed   - q
Message completely sent - mc
Shared resource         - sr
Terminal ID - CSID      - csid
Remote system COMID     - c
```

The fields have the same meanings as previously described.

For an ALC display accessed through SLC:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
SLC-DSP  crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Messages on queue       - n
Number of lines/rows    - l
Terminal/AAA hold       - aaa
Test modules            - tm
CRN of owning SLC link  - crn4
HEX                     - hex
TCID                    - tcid
IA                      - ia
TA                      - ta
Translate code          - code
```

Where:

crn4
CRN of the owning SLC link.

hex
High level network exit address.

tcid
Terminal circuit identity.

ia
Terminal interchange address.

ta
Terminal address.

code
Translate code for messages.

The other fields have the same meanings as previously described.

For an ALC printer accessed through SLC:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
SLC-PRT  crn      cri    ordinal  appl     status   cras   f   auth
User ID               - userid
Associated resource CRN - crn1
Messages on queue pri p - n
Printer buffer size     - pb
Timeout 1 - seconds     - t1
Timeout 2 - seconds     - t2
Timeout repetitions     - tr
Resource is unusable    - u
System sends allowed    - ss
Message shadowing       - ms
Shadow CRNs             - crn2
Message re-direction    - mr
Re-direction CRN        - crn3
Waiting for ACK         - ack
Queue being processed   - q
Message completely sent - mc
Shared resource         - sr
CRN of owning SLC link  - crn4
HEX                     - hex
TCID                    - tcid
IA                      - ia
TA                      - ta
Translate code          - code
```

The fields have the same meanings as previously described.

For an ALC display accessed through TCP/IP:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
TCP-DSP  crn      cri    ordinal  appl     status   cras   f   auth
User ID               - userid
Associated resource CRN - crn1
Messages on queue       - n
Number of lines/rows    - l
Terminal/AAA hold       - aaa
Test modules            - tm
Scrolling Inhibited     - si
CRN of owning resource  - crn4
CRN of current TCP/IP connection - crn5
HEX                     - hex
TCID                    - tcid
IA                      - ia
TA                      - ta
Translate code          - code
TCP/IP remote host address - rh
TCP/IP remote host addr 2 - rh2
TCP/IP remote host addr 3 - rh3
TCP/IP remote host addr 4 - rh4
```

Where:

crn4
CRN of the owning TCP/IP server.

crn5
CRN of the current TCP/IP connection.

hex
High level network exit address.

tcid
Terminal circuit identity.

ia
Terminal interchange address.

ta
Terminal address.

code
Translate code for messages.

rh
Remote host IP address (rhost). The primary or sole host defined.

rh2,rh3,rh4
Alternate remote TCP/IP host addresses, if defined.

si
Scrolling inhibited (YES or NO).

The other fields have the same meanings as previously described.

For an ALC printer accessed through TCP/IP:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
TCP-PRT  crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Messages on queue pri p - n
Printer buffer size     - pb
Timeout 1 - seconds     - t1
Timeout 2 - seconds     - t2
Timeout repetitions     - tr
Resource is unusable    - u
System sends allowed    - ss
Message shadowing       - ms
Shadow CRNs             - crn2
Message re-direction    - mr
Re-direction CRN        - crn3
Waiting for ACK         - ack
Queue being processed   - q
Message completely sent - mc
Shared resource         - sr
CRN of owning resource  - crn4
CRN of current TCP/IP connection - crn5
HEX                     - hex
TCID                    - tcid
IA                      - ia
Translate code          - code
TCP/IP remote host address - rhost
```

The fields have the same meanings as previously described.

For an ALC display accessed through an X.25 PVC:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
X25-DSP  crn      cri    ordinal  appl     status   cras   f   auth
User ID                - userid
Associated resource CRN - crn1
Messages on queue       - n
Number of lines/rows    - l
Terminal/AAA hold       - aaa
Test modules            - tm
CRN of owning X.25 PVC  - crn5
TCID                    - tcid
IA                      - ia
TA                      - ta
```

Where:

crn5
CRN of the owning X.25 PVC.

The other fields have the same meanings as previously described.

For an ALC printer accessed through an X.25 PVC:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   CRAS       Authority
X25-PRT  crn      cri    ordinal  appl     status   cras   f   auth
User ID               - userid
Associated resource CRN - crn1
Messages on queue pri p - n
Printer buffer size     - pb
Timeout -  seconds      - t1
Timeout -  seconds      - t2
Timeout repetitions     - tr
Resource is unusable    - u
Shared resource         - sr
System sends allowed    - ss
Message shadowing       - ms
Shadow CRNs             - crn2
Message re-direction    - mr
Re-direction CRN        - crn3
Waiting for ACK         - ack
Queue being processed   - q
Message completely sent - mc
CRN of owning X.25 PVC  - crn5
IA                      - ia
TA                      - ta
```

The fields have the same meanings as previously described.

For a Type 1, 3, 4, 5, 6 X.25 PVC:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Type
X25PVC   crn      cri    ordinal  appl     status   t
VTAM SEND queue         - v
```

The fields have the same meanings as previously described.

For a Type 2 X.25 PVC:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Type
X25PVC   crn      cri    ordinal  appl     status   2
CRN of virtual SLC link - crn6
VTAM SEND queue         - v
BATAP Exit program 1    - prog
BATAP Exit program 2    - prog
BATAP Exit program 3    - prog
BATAP Exit program 4    - prog
BATAP Exit program 5    - prog
BATAP Input window size - n
BATAP Output window size- n
BATAP Timeout (seconds) - n
BATAP Retry count       - n
```

Where:

crn6
CRN of the virtual SLC link.

prog
Name of the BATAP installation-wide exit program.

n
Value of the BATAP variable.

The other fields have the same meanings as previously described.

See “Set or clear BATAP variables for AX.25 and MATIP Type B message processing” on page 77 for a description of the fields.

For a WTTY link:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
WTTY-HDX crn      cri    ordinal  appl     status
User ID                - userid
```

The fields have the same meanings as previously described.

For an SLC LINK:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Type Hen
SLCLINK  crn      cri    ordinal  appl     status   ts   hen
User ID                - userid
Remote system COMID    - c
Loop bit setting       - lbs
Loop bit test          - lbt
CML exchange           - cml
Type A single block CML - cmla
Type B single block CML - cmlb
STOP-ALL exchange      - st
Response to ENQUIRY    - resp
Message retransmission  - mr
SLC timer values - seconds:
  i1   i2   i3   i4   i5   i6   i7   i8   i9  i10  i11  i12
  t    t    t    t    t    t    t    t    t   t    t    t
SLC counter values:
  c1   c2   c3   c4   c5   c6   c7   c8
  v    v    v    v    v    v    v    v
RCR message queue:  Item ... In use ... File address
                    1        u          address
                    2        u          address
                    3        u          address
                    4        u          address
                    5        u          address
                    6        u          address
                    7        u          address
                    Window: w
  ...
Messages on link output queues:
Type A                 - n
Type B                 - n
NCBs                   - n
```

Where:

c
Remote system communication identifier (for Type 2 SLC links only)

lbs
Loop bit setting (1 or 0).

lbt
Loop bit test option (Yes or No).

cml
Clear message labels are exchanged (Yes or No).

cmla
Type A single block CMLs are exhanged (Yes or No).

cmlb
Type B single block CMLs are exchanged (Yes or No).

st
STOP-ALL LCBs are exchanged (Yes or No).

resp
Valid response to an enquiry LCB does (ACK) or does not (RSM) include an acknowledgement.

mr
Retransmit a partial (PART) or complete (ALL) message after an I/O error.

t...t
SLC timer values in seconds.

v...v
SLC counter values.

u
ALCS maintains a queue of output Type B messages for an SLC link if required. This queue item contains an output message (Yes or No).

address
File address of the output Type B message on queue.

w
The maximum number of output Type B messages has (Closed) or has not (Open) been reached.

n
The number of messages (Type A and Type B) and the number of network control blocks (NCBs) queued for transmission over the SLC link.

See *ALCS Installation and Customization* for a full description of these values.

The other fields have the same meanings as previously described.

For an LU 6.1 link:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
LU61BASE crn      cri    ordinal  appl     status
User ID                - userid
Remote system COMID    - c
Messages on queue      - n
```

Where:

c
Remote system communication identifier.

n
The total number of messages on queue. This is the total for the base link and all parallel sessions in this LU 6.1 link.

The other fields have the same meanings as previously described.

For an LU 6.1 parallel session:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Base Link
LU61SESS crn      cri    ordinal  appl     status   crn7
User ID                - userid
Remote system COMID    - c
Remote half-session name- rhsid
Session type           - stype
Resource is unusable   - u
Queue being processed  - q
Messages on queue      - n
```

Where:

crn7
CRN of the base LU 6.1 link.

rhsid
The name of the remote half-session. If there is no name the response is None. This value is defined in the communication generation. *ALCS Installation and Customization* describes the ALCS generation COMDEF macro parameters for an LU 6.1 parallel session.

stype
Type of session (Receive or Send). This value is defined in the communication generation. *ALCS Installation and Customization* describes the ALCS generation COMDEF macro parameters for an LU 6.1 parallel session.

n
The total number of messages on the queue for this parallel session.

The other fields have the same meanings as previously described.

For an ALCS local application:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
ALCSAPPL crn      cri    ordinal  prog     status
Initial status permanent - ip
Process commands        - pc
Special processing      - sp
Input message format    - msg
Minimum system state    - sys
Suppress FID conversion - nf
Mixed case              - mc
```

Where:

ip
The ALCS operator is prevented from using the ZACOM command to change the status of the application. (Yes or No).

pc
The application can process ALCS commands (Yes or No).

sp
For Type A (conversational) messages on an SLC link or X.25 PVC, ALCS formats the message (removes backspace and invalid characters) before presenting it to the application (Yes or No).

msg
Input message format expected by the application (see *ALCS Application Programming Guide* for details of message formats). This is one of:

IMSG
Input message format.

OMSG
Output message format.

AMSG
Application message format.

XMSG
IPARS message switching format.

sys
The lowest system state in which the application program accepts input messages. This is one of:

IDLE
CRAS
MESW
NORM

nf
For messages from ALC terminals, ALCS does (No) or does not (Yes) convert any field identifier (FID) characters in the text. The field identifier keys cause an alphabetic character followed by a colon, ':', to be placed in the input message. ALCS replaces these two characters by the appropriate command.

mc
ALCS does (No) or does not (Yes) translate the input message text to upper case before presenting it to the application.

The values are defined in the communication generation -- see *ALCS Installation and Customization* for details of *ip, pc, sp, sys, nf*, and *mc*, and *ALCS Application Programming Guide* for an explanation of *msg*.

For an ALCS remote application:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
ALCSAPPL crn      cri    ordinal           status
Initial status permanent - ip
Remote system COMID      - c
```

Where:

ip
The ALCS operator is prevented from using the ZACOM command to change the status of the application (Yes or No).

c
For an application owned by another host system, the communication identifier of that system.

For a virtual SLC link (VSLCLINK)

There is no detailed response for this resource type. Specifying D produces the basic response shown in “Normal responses (basic)” on page 140.

For an APPC connection:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Type
APPC     crn      cri    ordinal  appl     status   tc
User ID                - userid
Local LU name          - name1
Partner LU name        - name2
Mode name              - name3
Symbolic dest name     - name4
Partner TP name        - name5
Conversation ID - rcv  - id1
Conversation ID - send - id2
```

Where:

tc
Type of APPC connection as specified on the PRTCOL parameter of the COMDEF generation macro.

name1
APPC local LU name (this line is omitted if there is no local LU name).

name2
APPC partner LU name (or blank if there is none).

name3
APPC mode name (or blank if there is none).

name4
APPC symbolic destination name (or blank if there is none).

name5
APPC partner TP name (or blank if there is none).

id1
APPC conversation ID of the conversation used to send data (this line is omitted if no conversation is allocated).

id2
APPC conversation ID of the conversation used to receive data (this line is omitted if no conversation is allocated).

The other fields have the same meanings as previously described.

For a TCP/IP connection - client:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
TCPIP-C crn       cri    ordinal  crn2     status
User ID                   - userid
TCP/IP local port number  - lp
TCP/IP remote port number - rp
Remote port match         - rpm
TCP/IP server port number - sp
Server port match         - spm
TCP/IP remote host addr   - rh
TCP/IP remote host addr2  - rh2
TCP/IP remote host addr3  - rh3
TCP/IP remote host addr4  - rh4
TCP/IP trace              - tr
Max message size in KB    - mmgsz
Idle connection timeout   - it
Blocked send timeout      - bt
Application type          - at
```

For a TCP/IP connection - server:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
TCPIP-S crn       cri    ordinal  crn2     status
User ID                   - userid
TCP/IP local port number  - lp
TCP/IP remote port number - rp
TCP/IP remote host addr   - rh
TCP/IP trace              - tr
Max message size in KB    - mmgsz
Idle connection timeout   - it
Blocked send timeout      - bt
Current connections       - cc
Maximum connections       - mc
Application type          - at
```

For a TCP/IP connection - dynamic server:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status   Base Server
TCPIP-P crn       cri    ordinal  crn2     status  crn3
User ID                   - userid
TCP/IP local port number  - lp
TCP/IP remote port number - rp
TCP/IP remote host addr   - rh
TCP/IP trace              - tr
Max message size in KB    - mmgsz
Idle connection timeout   - it
Blocked send timeout      - bt
Application type          - at
```

For a TCP/IP connection - MATIP Type A server:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
TCPIP-S crn       cri    ordinal  crn2     status
User ID                   - userid
TCP/IP local port number  - lp
TCP/IP remote port number - rp
TCP/IP remote host addr   - rh
TCP/IP trace              - tr
Max message size in KB    - mmgsz
Idle connection timeout   - it
Blocked send timeout      - bt
Current connections       - cc
Maximum connections       - mc
Application type          - MATIP-A
Translate code            - code
Data translation          - dt
MATIP session status      - mss
MATIP session type        - mat
```

For a TCP/IP connection - MATIP Type B server:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
TCPIP-S crn       cri    ordinal  crn2     status
User ID                   - userid
TCP/IP local port number  - lp
TCP/IP remote port number - rp
TCP/IP remote host addr   - rh
TCP/IP trace              - tr
Idle connection timeout   - it
Blocked send timeout      - bt
Current connections       - cc
Maximum connections       - mc
Application type          - MATIP-B
Translate code            - code
Data translation          - dt
MATIP session status      - mss
MATIP session type        - mat
```

For a TCP/IP connection - MATIP Type A host-to-host client:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
TCPIP-C crn       cri    ordinal  crn2     status
User ID                   - userid
TCP/IP local port number  - lp
TCP/IP remote port number - rp
Remote port match         - rpm
TCP/IP server port number - sp
Server port match         - spm
TCP/IP remote host addr   - rh
TCP/IP remote host addr2  - rh2
TCP/IP remote host addr3  - rh3
TCP/IP remote host addr4  - rh4
TCP/IP trace              - tr
Max message size in KB    - mmgsz
Idle connection timeout   - it
Blocked send timeout      - bt
Application type          - MATIP-A
HEX                       - hex
Translate code            - code
Data translation          - dt
MATIP session status      - mss
MATIP session type        - mat
MATIP MPX                 - mpx
MATIP HDR                 - hdr
MATIP flow ID             - flw
```

For a TCP/IP connection - MATIP Type B client:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
TCPIP-C crn       cri    ordinal  crn2     status
User ID                   - userid
TCP/IP local port number  - lp
TCP/IP remote port number - rp
Remote port match         - rpm
TCP/IP server port number - sp
Server port match         - spm
TCP/IP remote host addr   - rh
TCP/IP remote host addr2  - rh2
TCP/IP remote host addr3  - rh3
TCP/IP remote host addr4  - rh4
TCP/IP trace              - tr
Idle connection timeout   - it
Blocked send timeout      - bt
Application type          - MATIP-B
HEN                       - hen
HEX                       - hex
Translate code            - code
Data translation          - dt
MATIP session status      - mss
MATIP session type        - mat
CRN of virtual SLC link   - crn6
BATAP Exit program 1      - prog
BATAP Exit program 2      - prog
BATAP Exit program 3      - prog
BATAP Exit program 4      - prog
BATAP Exit program 5      - prog
BATAP Input window size   - n
BATAP Output window size  - n
BATAP Timeout (seconds)   - n
BATAP Retry count         - n
BATAP Batched IMAs        - n
BATAP Flush   - seconds   - n
```

Where:

lp
Local TCP/IP port number.

rp
Remote TCP/IP port number.

rpm
Remote port match is on (Yes) or off (No).

sp
Server TCP/IP port number.

spm
Server port match is on (Yes) or off (No).

rh
Remote TCP/IP host address (primary or only)

rh2, rh3, rh4
Alternate remote TCP/IP host addresses, if defined. One of the *rh*, *rh2*, *rh3*, or *rh4* values can be followed by an asterisk to show which of the primary or alternate addresses was last used.

hen
High-level designator as specified on the HEN parameter of the COMDEF generation macro.

hex
High-level designator as specified on the HEX parameter of COMDEF generation macro.

tr
TCP/IP trace facility is on (Yes) or off (No) for this TCP/IP connection.

mmgsz
Maximum message size in KB, if defined

it
Idle connection timeout value in seconds, or 0 if there is no timeout.

bt
Blocked send timeout value in seconds, or 0 if there is no timeout.

cc
Current number of concurrent connections for this server.

mc
Maximum possible number of concurrent connections for this server, or 0 if there is no limit.

at
Application protocol (or blank if there is none).

code
For MATIP connections, the translate code used when ALCS establishes the connection.

dt
For MATIP connections, ALCS data translation is on (Yes) or off (No).

mss
For MATIP connections, Session Status Query (SSQ) packets will (Yes) or will not (No) be sent.

mat
For a TCP/IP connection that uses the MATIP protocol, this is the current MATIP session status. One of:

None
No session

A
Type A terminal-to-host session

A IATA H-TO-H
Type A IATA host-to-host session

A SITA H-TO-H
Type A SITA host-to-host session

B
Type B session

mpx
MATIP multiplex type for the session.

hdr
MATIP header type for the session.

flw
MATIP flow ID for the session.

crn6
CRN of the virtual SLC link.

prog
Name of the BATAP installation-wide exit program.

n
Value of the BATAP variable.

The other fields have the same meanings as previously described.

See "Set or clear BATAP variables for AX.25 and MATIP Type B message processing" on page 77 for a description of the BATAP fields.

For an unrecognized device type:

```
DXC8900I CMD i  hh.mm.ss DCOM
Resource CRN      CRI    Ordinal  Routing  Status
******** crn      cri    ordinal  appl     status
Associated resource CRN - crn
Messages on queue       - n
```

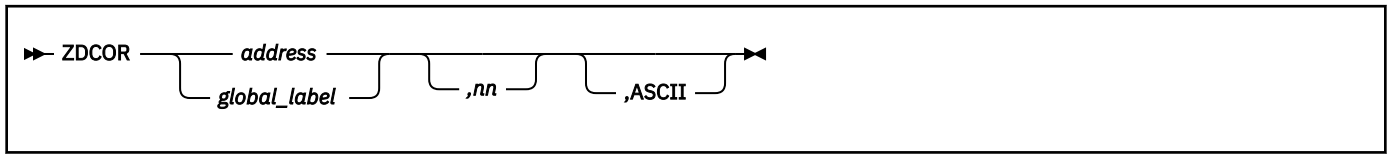
The fields have the same meanings as previously described.

ZDCOR -- Display storage area

Use the ZDCOR command -- ***from any CRAS authorized terminal*** -- to display an area of storage or the contents of a global field.

This command uses the ALCS scrolling function, see "Scrolling displays" on page 58 for details.

The format of the command is:



Where:

address
Storage address. You can omit leading zeros.

global_label
A 1 through 8 character label of a field in the application global area. (The system programmer sets up these labels in the programs AGT*n*, as described in *ALCS Installation and Customization*.)

nn
Number of fullwords to display. The scrolling function is not invoked if *nn* is less than or equal to 16 fullwords.

ASCII
Interpret the binary data as ASCII characters. Omit this parameter to interpret the data as EBCDIC characters.

Normal response

```
DXC8201I CMD i  hh.mm.ss DCOR  aaaaaaaa
oooo ** dddddddd dddddddd dddddddd dddddddd  *cccccccccccccccc*
oooo    dddddddd dddddddd dddddddd dddddddd  *cccccccccccccccc*
oooo ** dddddddd ........                    *cccc....        *
⋮
```

Where:

aaaaaaaa
Starting storage address of the storage displayed.

oooo
Low four hexadecimal digits of the storage address of the storage displayed. (On the first line this is the address of the fullword containing the first displayed byte).

****
Indicates that the line is repeated on the display.

dd...dd
Storage contents (hexadecimal).

cc...cc
Storage contents (character).

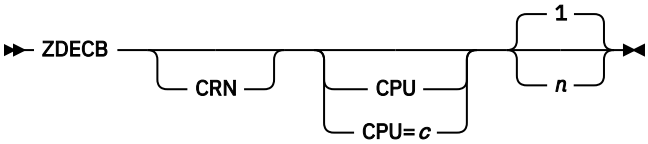
If the requested display extends into a storage area to which application programs do not have read access, then ALCS only displays the storage that they can access.

ZDECB -- Display information about active ECBs

Use the ZDECB command -- ***from any terminal*** -- to display a list of active ECBs.

This command invokes the ALCS scrolling function, see "Scrolling displays" on page 58 for details.

The format of the command is:



Where:

n
Show the number of entries which are older than *n* seconds. *n* can be any number from 0 (display all entries) to 9999. The default value is 1 second.

CRN
For each entry, show the CRN of the originating terminal instead of its CRI.

CPU
Show the CPU utilization for each entry when the data collection CPU collector is running.

CPU=*c*
Show the CPU utilization for each entry when the data collection CPU collector is running and omit entries which have been used less than c seconds of CPU time. *c* can be any number from 0 (display all entries) to 999.

Normal response

The normal response is either:

```
DXC8290I CMD i  hh.mm.ss DECB  Entries older than n second(s) -- None
```

if there are no entries longer than *n* seconds.

Or

```
DXC8291I CMD i  hh.mm.ss DECB  Entries older than n second(s) -- s
ECB      Origin   --Last-macro--- -ECB-age- -Create-- Hold-counts Waiting  CPU
address  CRI      Prog Macro Addr hhh.mm.ss Type Prog Rec Res Seq for-ECB  sss.iii
xxxxxxxx xxxxxx   xxxx xxxxx xxxx xxxxxxxx  xxxx xxxx xxx xxx xxx xxxxxxxx xxx.xxx
xxxxxxxx xxxxxx   xxxx xxxxx xxxx xxxxxxxx  xxxx xxxx xxx xxx xxx xxxxxxxx xxx.xxx
⋮
```

Or when CRN is specified:

```
DXC8291I CMD i  hh.mm.ss DECB  Entries older than n second(s) -- s
ECB      Origin   --Last-macro--- -ECB-age- -Create-- Hold-counts Waiting  CPU
Address  CRN      Prog Macro Addr hhh.mm.ss Type Prog Rec Res Seq for-ECB  sss.iii
xxxxxxxx xxxxxxxx xxxx xxxxx xxxx xxxxxxxx  xxxx xxxx xxx xxx xxx xxxxxxxx xxx.xxx
xxxxxxxx xxxxxxxx xxxx xxxxx xxxx xxxxxxxx  xxxx xxxx xxx xxx xxx xxxxxxxx xxx.xxx
⋮
```

Or when CPU is specified:

```
DXC8289I CMD i  hh.mm.ss DECB  Entries older than n second(s) -- s
ECB      Origin   --Last-macro--- -ECB-age- -Create-- Hold-counts Waiting  CPU
Address  CRI      Prog Macro Addr hhh.mm.ss Type Prog Rec Res Seq for-ECB  sss.iii
xxxxxxxx xxxxxx   xxxx xxxxx xxxx xxxxxxxx  xxxx xxxx xxx xxx xxx xxxxxxxx xxx.xxx
xxxxxxxx xxxxxx   xxxx xxxxx xxxx xxxxxxxx  xxxx xxxx xxx xxx xxx xxxxxxxx xxx.xxx
⋮
```

Or when CRN and CPU are specified:

```
DXC8289I CMD i  hh.mm.ss DECB  Entries older than n second(s)-- s
ECB      Origin   --Last-macro--- -ECB-age- -Create-- Hold-counts Waiting  CPU
Address  CRN      Prog Macro Addr hhh.mm.ss Type Prog Rec Res Seq for-ECB  sss.iii
xxxxxxxx xxxxxxxx xxxx xxxxx xxxx xxxxxxxx  xxxx xxxx xxx xxx xxx xxxxxxxx xxx.xxx
xxxxxxxx xxxxxxxx xxxx xxxxx xxxx xxxxxxxx  xxxx xxxx xxx xxx xxx xxxxxxxx xxx.xxx
⋮
```

Where:

s
The number of entries that are older than *n* seconds.

ECB address
ECB address.

Origin CRI
CRI of originating terminal.

Origin CRN
CRN of originating terminal. Entries created by the ALCS monitor are shown as *SYSTEM*. Entries with an unknown originator are shown as ********.

Last-macro
Details of last macro:

Prog
Name of program.

Macro
Name of macro.

Addr
Listing address in the program.

ECB-age
Entry age in hours, minutes, and seconds with leading zeros suppressed. Entries younger that *n* seconds are not shown. The oldest entry is shown first. Entries older than 999 hours 59 minutes and 59 seconds (approximately 5 weeks) are shown as ***.**.**.

Create
If the entry creates another entry, these columns show how the entry was created.

Type
One of:

SNA
SNA created entry.

TAS
TAS created entry.

H/S
High speed (or type A) input message entry.

L/S
Low speed (or type B) input message entry.

CRE
Entry created by CREMC, CREDC, or CREXC macro (or the equivalent C language function).

CRET
Entry created by CRETC macro (or the equivalent C language function).

MON
Entry created by ALCS monitor.

ROUT
Entry created by ALCS router.

Prog
For entries created by CREMC, CREDC, CREXC, or CRETC macro, (or the equivalent C language function). This column contains the creating program name. For other entries, the column is blank (spaces).

Hold-counts
Hold counts are shown in decimal with leading zeros suppressed. Zero counts are shown as blanks. Where:

Rec
Number of records held by this entry.

Res
Number of resources held by this entry.

Seq
Number of sequential files held by this entry.

Waiting for ECB
This column displays an address if, and only if, this entry is queued by record hold, resource hold, or sequential file assign. The column shows the address of the ECB immediately in front of this one in the queue.

CPU
CPU time used by the entry in seconds and milliseconds with leading zeroes suppressed. Entries that have used more than 999 seconds and 999 millliseconds of CPU time are shown as ***.***. Non-zero CPU times are only shown when the data collection CPU collector is running (see "ZDCLR -- Control data collection" on page 134 for information about using the data collection ZDCLR command).

If the entry is being traced conversationally, the token `trace` is added to the end of the display line. If the CPU parameter has been specified the token will be displayed in the CPU column.

If the entry has been purged by ZPURG FORCE, the token `purged` is added to the end of the display line. If the CPU parameter has been specified the token will be displayed in the CPU column.

If your ALCS system is using the Common Entry Point (CEP) feature of TPFDF, additional information is provided for those entries that have issued a TPFDF macro call. This information is provided in an additional display line that includes the name of the application program that issued the TPFDF macro, the listing address of the TPFDF macro within the application program, and the name of the TPFDF macro call.

Normal response when CPU=c is specified

The normal response is either:

```
DXC8293I CMD i  hh.mm.ss DECB
Entries older than n second(s) with at least c seconds CPU -- None
```

if there are no entries which have used at least *c* seconds of CPU time.

Or

```
DXC8294I CMD i  hh.mm.ss DECB
Entries older than n second(s) with at least c seconds CPU -- s
ECB      Origin   --Last-macro--- -ECB-age- -Create-- Hold-counts Waiting  CPU
address  CRI      Prog Macro Addr hhh.mm.ss Type Prog Rec Res Seq for-ECB  sss.iii
xxxxxxxx xxxxxx   xxxx xxxxx xxxx xxxxxxxx xxxx  xxxx xxx xxx xxx xxxxxxxx xxx.xxx
xxxxxxxx xxxxxx   xxxx xxxxx xxxx xxxxxxxx xxxx  xxxx xxx xxx xxx xxxxxxxx xxx.xxx
⋮
```

Or when CRN is specified:

```
DXC8294I CMD i  hh.mm.ss DECB
Entries older than n second(s) with at least c seconds CPU -- s
ECB      Origin   --Last-macro--- -ECB-age- -Create-- Hold-counts Waiting  CPU
Address  CRN      Prog Macro Addr hhh.mm.ss Type Prog Rec Res Seq for-ECB  sss.iii
xxxxxxxx xxxxxxxx xxxx xxxxx xxxx xxxxxxxx  xxxx xxxx xxx xxx xxx xxxxxxxx xxx.xxx
xxxxxxxx xxxxxxxx xxxx xxxxx xxxx xxxxxxxx  xxxx xxxx xxx xxx xxx xxxxxxxx xxx.xxx
⋮
```

Where:

s
: The number of entries that are older than *n* seconds and have used at least *c* seconds of CPU time.

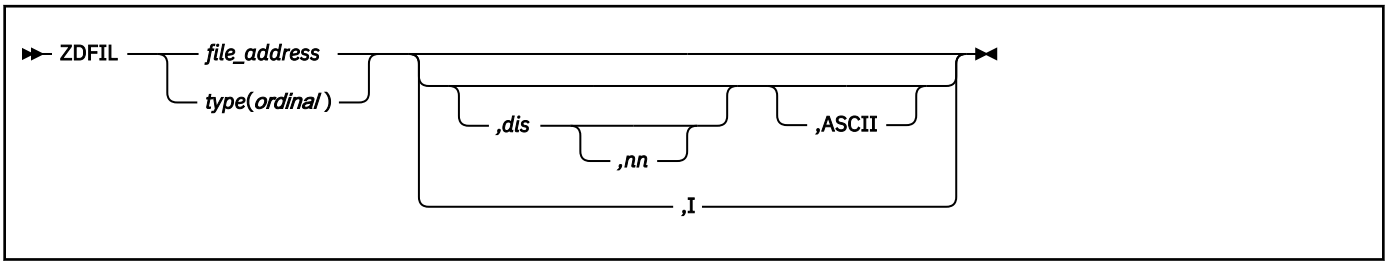
The other fields have the same meanings as previously described.

ZDFIL -- Display contents of a DASD record

Use the ZDFIL command -- ***from any CRAS authorized terminal*** -- to display the contents of a DASD record.

This command invokes the ALCS scrolling function, see "Scrolling displays" on page 58 for details.

The format of the command is:



Where:

file_address
File address (8 or 16 hexadecimal digits).

type
Any of the following:

GF-*nnn*
nnn is the general file number (3 decimal digits).

#xxxxx
Fixed record type.

LsLTpool
Long-term pool record, where L*s* identifies the record size.

LsSTpool
Short-term pool record, where L*s* identifies the record size.

Lspool
Allocatable pool record, where L*s* identifies the record size.

LsDIR
Allocatable pool directory, where L*s* identifies the record size.

LsSTDIR
Short-term pool directory, where L*s* identifies the record size.

FFINDEX
Fixed-file index.

STINDEX
Short-term index.

CDS*n*
Configuration data set. *n* is the configuration data set number (0, 1, or 2).

ordinal
Record ordinal number; a decimal number.

dis
Starting displacement within the record, up to 4 hexadecimal digits. The default is displacement 0.

nn
Number of fullwords to be displayed. The scrolling function is not invoked if *nn* is less than or equal to 17 fullwords.

I

Display information about the record, such as the record type, ID, RCC (record code check), ordinal number, and the name of the application or TPFDF program that filed the record.

For ***long-term pool records*** the history of the record is also displayed, providing information about:

- When the record was dispensed.
- The last time the record was filed.
- The last time the record was chain-chased by Recoup.
- When the record was released.

This information can be useful for analyzing errors in long-term pool handling.

For ***type 2 short-term pool records***, the most recent events recorded by short-term pool event logging are also displayed, providing information about:

- When the record was dispensed.
- The last time the record was read.
- The last time the record was filed.
- When the record was released.

This information can be useful for analyzing errors in short-term pool handling.

Note: Either the name of the application program which requested TPFDF to file the record or the name of the TPFDF program that filed the record will be displayed. (See the description of the SCTGEN TPFDF parameter in *ALCS Installation and Customization* for more details.)

ASCII

Interpret the binary data as ASCII characters. Omit this parameter to interpret the data as EBCDIC characters.

Normal responses

The normal response to ZDFIL without the I parameter is a hexadecimal and character display of the whole record, as follows:

```
DXC8219I CMD i  hh.mm.ss DFIL  file_address
 oooo ** dddddddd dddddddd dddddddd dddddddd  *ccccccccccc*
 oooo    dddddddd dddddddd dddddddd dddddddd  *ccccccccccc*
 oooo ** dddddddd ........                    *cccc....    *

  ⋮
```

Where:

file_address

File address of the record displayed.

oooo

Displacement within the record of the data displayed. (On the first line this is the displacement of the fullword containing the first displayed byte).

dd...dd

Record contents (hexadecimal).

cc...cc

Record contents (character).

The normal response to ZDFIL I varies according to the type of record as follows:

For ***fixed file*** or ***type 1 short-term pool*** records, the response is:

```
DXC8222I CMD i  hh.mm.ss DFIL  file_address
Record Class ........class
Record Type ........ type-hex   type-id
Ordinal ............ nnn

Record Size ........ rsze-hex/rsze-id     Block Size ......... size-hex/size-dec
VFA Option ......... vfa-opt              CI Size ............ size-hex/size-dec
Record ID/Code check id/rcc               Program stamp....... prog

Number of Records .. rec
Max Ordinal ........ max

Allocatable pool
   Type.............rsze-id
   Ordinal..........nnn
   File Address..... file-address

                     hh.mm.ss   dd.mm.yy   yyddd
Last file .......... hh.mm.ss   dd.mm.yy   yyddd
Last write ......... hh.mm.ss   dd.mm.yy   yyddd
```

Where:

class
Record class.

type-hex
Record type in hex.

type-id
Record type id.

nnn
Record ordinal number.

The indicator *(Deleted)* following the ordinal number indicates the file address will soon be purged and is therefore invalid.

rsze-hex
Record size type in hex.

rsze-id
Record size type id.

size-hex
Size in hex.

size-dec
Size in decimal.

vfa-opt
Record VFA options. For further explanation please refer to USRDTA macro for the DASD generation in the *Installation and Customization guide*.

id/rcc
Record ID and RCC (record code check).

prog
Name of the application program that filed the record.

rec
Number of record for that record type.

max
Max ordinal for that record type.

file address
Record file address.

`Last File`
The time and date when the record was last filed.

Last Write
The time and date when the record was last written. This line only appears when the write time is different from the file time.

hh.mm.ss
Hours, minutes, and seconds.

dd.mm.yy
Day, month, and year.

yyddd
Year and day number in year.

The other fields have the same meanings as previously described.

For ***type 2 short-term pool*** records, the response is:

```
DXC8222I CMD i  hh.mm.ss DFIL  file_address
Record Class ........class

Record Type ........ type-hex   type-id
Ordinal ............ nnn

Record Size ........ rsze-hex/rsze-id     Block Size ......... size-hex/size-dec
VFA Option ......... vfa-opt              CI Size ............ size-hex/size-dec
Record ID/Code check id/rcc               Program stamp....... prog

Number of Records .. rec
Max Ordinal ........ max

ST Dir Byte Value... byte                  ST Dir Byte Disp ... disp
ST Dir Number ...... dir

Allocatable pool
   Type............. rsze-id

   Ordinal.......... nnn

   File Address..... file-address

                     hh.mm.ss   dd.mm.yy   yyddd
Last file .......... hh.mm.ss   dd.mm.yy   yyddd
Last write ......... hh.mm.ss   dd.mm.yy   yyddd

Event log            hh.mm.ss   dd.mm.yy   yyddd   Prog
   Dispense ........ hh.mm.ss   dd.mm.yy   yyddd   prog
   File ............ hh.mm.ss   dd.mm.yy   yyddd   prog
   Find ............ hh.mm.ss   dd.mm.yy   yyddd   prog
   Release ......... hh.mm.ss   dd.mm.yy   yyddd   prog
```

Where:

byte
Pool directory byte contents in hex.

disp
Pool directory byte displacement within directory.

dir
Pool directory record ordinal.

Event log
The most recent events (dispenses, finds, files, and releases) for this record. The event log only appears when short-term pool event logging is enabled and event details have been recorded for the record.

The other fields have the same meanings as previously described.

For ***long-term pool*** records, the response is:

```
DXC8222I CMD i  hh.mm.ss DFIL  file_address
Record Class ........class

Record Type ........ type-hex   type-id
Ordinal ............ nnn

Record Size ........ rsze-hex/rsze-id     Block Size ......... size-hex/size-dec
VFA Option ......... vfa-opt              CI Size ............ size-hex/size-dec
Record ID/Code check id/rcc               Program stamp....... prog

Allocatable pool
   Type..............rsze-id
   Ordinal...........nnn
   File Address..... file-address

History              hh.mm.ss   dd.mm.yy   yyddd   Prog
Dispense ........... hh.mm.ss   dd.mm yy   yyddd   prog
Last file .......... hh.mm.ss   dd.mm.yy   yyddd   prog
Last Chain Chase ... hh.mm.ss   dd.mm.yy   yyddd   prog
Release ............ hh.mm.ss   dd.mm.yy   yyddd   prog
Last write ......... hh.mm.ss   dd.mm.yy   yyddd
```

Where:

Dispense
The time and date the record was dispensed (and the program name).

Last Chain Chase
The time and date that the record was last chain-chased using Recoup (and the descriptor program name).

Release
The time and date the record was released (and the program name).

The other fields have the same meanings as previously described.

For ***type 1 long-term pool*** records, the history information and the allocated status information are not set until the record has been updated on the database. Therefore the normal display will be slightly different during the first few days of a new ALCS installation.

For ***long-term pool*** records allocated from the **allocatable-pool space** when you specify the allocatable pool file address, the response is:

```
DXC8222I CMD i  hh.mm.ss DFIL  file_address
Record Class ........class

Record Type ........ type-hex   type-id
Ordinal ............ nnn

Record Size ........ rsze-hex/rsze-id     Block Size ......... size-hex/size-dec
VFA Option ......... vfa-opt              CI Size ............ size-hex/size-dec
Record ID/Code check id/rcc               Program stamp....... prog

Allocated status ... status
Record Type ........ type
Ordinal ............ nnn
File Address........ file-address

History              hh.mm.ss   dd.mm.yy   yyddd   Prog
Dispense ........... hh.mm.ss   dd.mm yy   yyddd   prog
Last File .......... hh.mm.ss   dd.mm.yy   yyddd   prog
Last Chain Chase ... hh.mm.ss   dd.mm.yy   yyddd   prog
Release ............ hh.mm.ss   dd.mm.yy   yyddd   prog
Last write ......... hh.mm.ss   dd.mm.yy   yyddd
```

Where:

status
The name of the allocated class, for example fixed-file or short-term pool or long-term pool.

The other fields have the same meanings as previously described.

For ***fixed-file*** or ***short-term pool*** records allocated from the **allocatable-pool space** when you specify the allocatable pool file address, the response is:

```
DXC8222I CMD i  hh.mm.ss DFIL  file_address
Record Class ........class

Record Type ........ type-hex   type-id
Ordinal ............ nnn

Record Size ........ rsze-hex/rsze-id     Block Size ......... size-hex/size-dec
VFA Option ......... vfa-opt              CI Size ............ size-hex/size-dec
Record ID/Code check id/rcc               Program stamp....... prog

Allocated status ... status
Record Type ........ type
Ordinal ............ nnn
File Address........ file-address

                     hh.mm.ss   dd.mm.yy   yyddd
Last File .......... hh.mm.ss   dd.mm.yy   yyddd
Last write ......... hh.mm.ss   dd.mm.yy   yyddd
```

The fields have the same meanings as previously described.

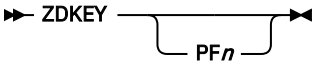
Message numbers DXC8219I and DXC8222I are used when the file address has 8 hexadecimal digits (4 bytes); message numbers DXC8232I and DXC8233I are used when the file address has 16 hexadecimal digits (8 bytes).

ZDKEY -- Display PF key settings of terminals

Use the ZDKEY command with no parameters -- ***from any terminal*** - to display the current PF key settings. See “ZAKEY -- Alter terminal program function key settings” on page 87 for how to set the PF key values.

This command invokes the ALCS scrolling function, see “Scrolling displays” on page 58 for details.

The format of the command is:



Where:

PF*n*
Program function key number *n*. Specify PF1 or PF01 for program function key one, and so on, up to PF24.

Normal response

The normal response depends on the way the PF keys have been set up in each installation. An example response to ZDKEY with no parameters is shown below:

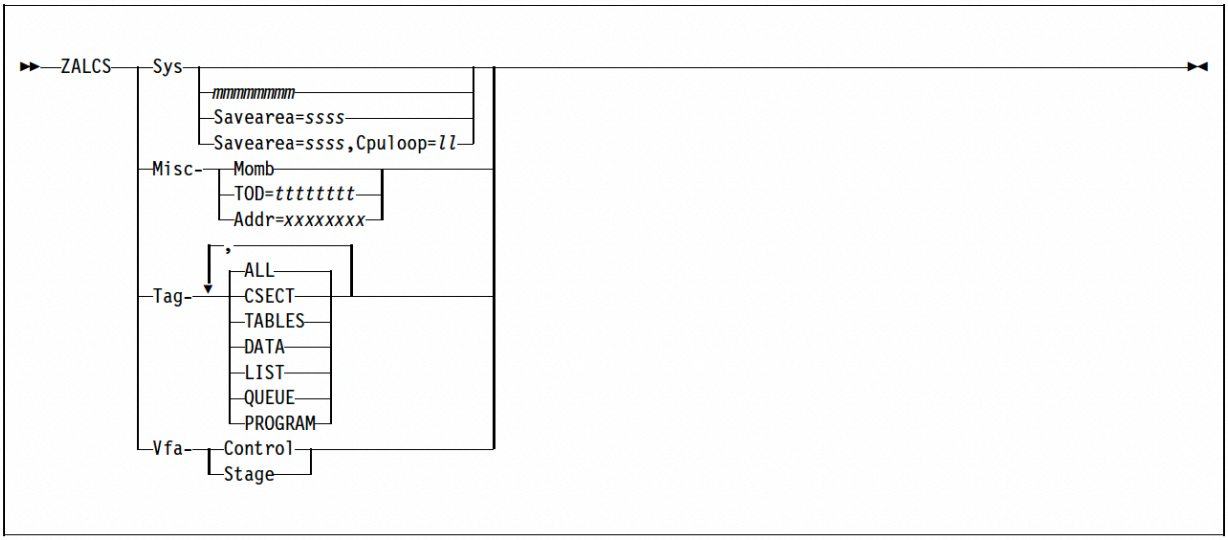
```
DXC8667I CMD i  hh.mm.ss DKEY
* indicates installation default
I indicates ZAKEY inhibited

* PF01 % ZSCRL COLS %1
* PF02 % ZSCRL COLS RESET
* PF03   0A
* PF04   R
* PF05 % ZSCRL FIND %1
* PF06   XI
I PF07 % ZSCRL UP %1
I PF08 % ZSCRL DOWN %1
* PF09
I PF10 % ZSCRL LEFT %1
I PF11 % ZSCRL RIGHT %1
* PF12 % ZSCRL PREV %1
* PF13   E
* PF14   I
* PF15   0A
```

In the example response, all the PF keys are at the installation default settings. Some, indicated by the I prefix, (for example PF7) cannot be altered from the installation default.

ZALCS --Display general ALCS information

Use the ZALCS command - from any CRAS authorized terminal - to display general ALCS information.

The format of the command is:



Where:

Sys
Display ALCS system information.

mmmmmmmm
Monitor storage area name. Default is DXCNUC.

Savearea=*ssss*
Display the respective monitor savearea. Variable *ssss* is the id of the monitor savearea. The default is NUC.

Cpuloop-*ll*
Specifies the the CPUloop number for the savearea display. Default is 00 (initializer task).

Misc
Display ALCS miscellaneous information.

Momb
Display the ALCS MOMB control area.

Addr=*xxxxxxxx*
Display information about the ALCS monitor CSECT that contains the address *xxxxxxxx*.

TOD=*tttttttt*
Display the conversion of the TOD clock specified by *tttttttt*. The maximum size for *tttttttt* is 16 heaxadecimal digits.

Tag
Display ALCS monitor storage tags. Use ALL to display all tags or any combination of CSECT, TABLES, DATA, LIST, QUEUE, and PROGRAM.

CSECT
Display all ALCS monitor csect tags.

TABLES
Display table tags.

DATA
Display general data tags.

LIST
Display list tags.

QUEUE
Display queue tags.

PROGRAM
display program table tags.

Vfa
display VFA control information.

Stage
display VFA staging buffer information.

Control
display VFA control area.

Normal response

Sample miscellaneous output for a specified address.

```
User:   zalcs m a=20000

System: DXC5245I CME M 18.52.55 ALCS
        Address-00020000 CSECT-DXCCOMF Base Addr-0001FC40 Offset-000003C0
```

where in
Address-aaaaaaaa CSECT-mmmmmmmm Base Addr-bbbbbbbb Offset-oooooooo

aaaaaaaa
CSECT address to be displayed.

mmmmmmmm
Monitor CSECT that contains the displayed address.

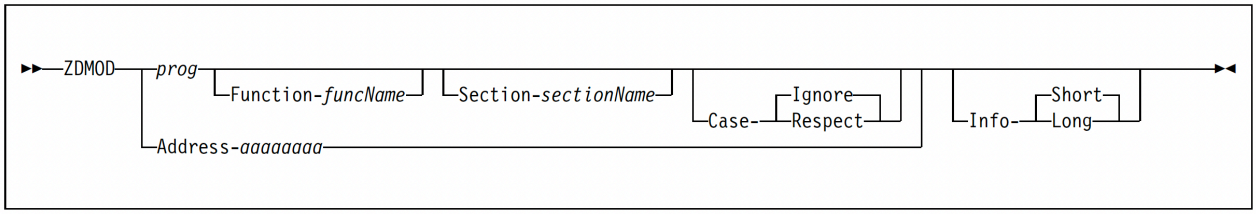
bbbbbbbb
Base address of the monitor CSECT.

oooooooo
Offset of the displayed address within the monitor CSECT.

ZDMOD --Display DASD information

Use the ZDMOD command with no parameters -- ***from any terminal*** - to display ALCS modules.

The format of the command is:



Where:

Address-*aaaaaaaa*
Hexadecimal address (1 to 8 hexadecimal characters) within the program to be displayed.

prog
Program name to be displayed.

Function-*funcName*
Name of the function to be displayed

Section-*sectionName*
Name of the program section to be displayed.

Case-Ignore|Respect
Specifies whether text case (upper or lower) is to be respected or ignored for FUNCTION and SECTION parameters. Case-Ignore is the default.

Info-Short|Long
Controls the amount of information to be displayed.

Notes:

1. Omitting the Function or Section parameters results in a display of all functions or sections in a program.
2. Wildcards can be used:

   An asterisk ("*") is used to match any number of characters.
   A period (".") is used to match a single character.
3. Module mapping data can come from three sources:
   a. Binder module mapping records loaded with the module.
   b. Binder module mapping records stored with the load module.
   c. Records that make up the load module.
4. The output displayed by ZDMOD will vary somewhat depending on the source of the mapping data.
5. The DMOD*xxxx*I message in the display identifies the source of the mapping data.

Normal response

An example response to ZDMOD with only a program parameter:

```
User: zdmod jbzdmod

System:
                   Lines      1 to     24 of     24  Columns    1 to  79 of 79
                   Active:                 *1*
DXC0001I    02:27:43
DXC5500I Mapping Data obtained from BMMP records loaded with module
                   JBZDMOD
                   Program Address                                    7D0EE000
                   Program Size                                       000467C8
cdmain.c - Section Size: 1D60 Offset: 4350 Address: 7D0F2350
  Compiled on: 2011.080
cdmdsp.c - Section Size: 15E8 Offset: 60B0 Address: 7D0F40B0
  Compiled on: 2011.080
cdmer1.c - Section Size: C90 Offset: 7698 Address: 7D0F5698
  Compiled on: 2011.080
cdmprc.c - Section Size: 1E60 Offset: 8328 Address: 7D0F6328
  Compiled on: 2011.080
cdmprs.c - Section Size: AE8 Offset: A188 Address: 7D0F8188
  Compiled on: 2011.080
czdmod.c - Section Size: 10790 Offset: AC70 Address: 7D0F8C70
  Compiled on: 2011.080
cdmisc.c - Section Size: 1578 Offset: 1B400 Address: 7D109400
  Compiled on: 2011.080
CLEZ.c - Section Size: F30 Offset: 1C978 Address: 7D10A978
  Compiled on: 2011.026
dxcbhllf.c - Section Size: 2FB0 Offset: 1D8A8 Address: 7D10B8A8
  Compiled on: 2011.076
END OF DISPLAY
```

Normal response

Sample output when the Function operand is used:

```
User: zdmod jbzdmod f-*sort*

System:
                     Lines      1 to     36 of    144  Columns   1 to  79 of 79

                     Active:              2  *1*      More:           down
 DXC0001I    02:25:29
DXC5500I Mapping Data obtained from BMMP records loaded with module
                    JBZDMOD
                    Program Address                                      7D0EE000
                    Program Size                                         000467C8


 cdmain.c - Section Size: 1D60 Offset: 4350 Address: 7D0F2350
   Compiled on: 2011.080
   Offset    Address   Function Name
   00000C80  7D0F2FD0  qsortModuleList

 czdmod.c - Section Size: 10790 Offset: AC70 Address: 7D0F8C70
   Compiled on: 2011.080
   Offset    Address   Function Name
   000012A0  7D0F9F10  sortFunctions__12ZdmodSectionFv
   00001358  7D0F9FC8  sort__3stdHQ2_3std6_PtritXTP13ZdmodFunctionTiTPP1
                       unctionTRP13ZdmodFunctionTPP13ZdmodFunctionTRP13Z
                       ction_18ZdmodFunctionsSort_Q2_3std6_PtritXTP13Zdm
                       ionTiTPP13ZdmodFunctionTRP13ZdmodFunctionTPP13Zdm
                       ionTRP13ZdmodFunction_T118ZdmodFunctionsSort_v
   00002670  7D0FB2E0  sort__3stdHQ2_3std6_PtritXTP12ZdmodSectionTiTPP12
                       ctionTRP12ZdmodSectionTPP12ZdmodSectionTRP12Zdmod
                       _17ZdmodSectionsSort_Q2_3std6_PtritXTP12ZdmodSect
                       P12ZdmodSectionTRP12ZdmodSectionTPP12ZdmodSection
                       modSection_T117ZdmodSectionsSort_v

   00007068  7D0FFCD8  _Push_heap__3stdHQ2_3std6_PtritXTP12ZdmodSectionT
                       dmodSectionTRP12ZdmodSectionTPP12ZdmodSectionTRP1
                       ection_iP12ZdmodSection17ZdmodSectionsSort_Q2_3st
                       PtritXTP12ZdmodSectionTiTPP12ZdmodSectionTRP12Zdm
                       onTPP12ZdmodSectionTRP12ZdmodSection_iT2P12ZdmodS
                       7ZdmodSectionsSort_v
   00007118  7D0FFD88  make_heap__3stdHQ2_3std6_PtritXTP13ZdmodFunctionT
                       dmodFunctionTRP13ZdmodFunctionTPP13ZdmodFunctionT
                       odFunction_18ZdmodFunctionsSort_Q2_3std6_PtritXTP
                       FunctionTiTPP13ZdmodFunctionTRP13ZdmodFunctionTPP
                       FunctionTRP13ZdmodFunction_T118ZdmodFunctionsSort
   00007220  7D0FFE90  _Unguarded_insert__3stdHQ2_3std6_PtritXTP13ZdmodF
 More...
```

Normal response

Sample output when INFO-LONG is specified:

```
User: zdmod jbzdmod info-l

System:
                     Lines      1 to     36 of    269  Columns   1 to  79 of 79

                     Active:      4   3   2  *1*      More:           down
 DXC0001I    02:27:43
 DXC5500I Mapping Data obtained from BMMP records loaded with module
                    JBZDMOD
                    Program Address                                    7D0EE000
                    Program Size                                       000467C8


 $IS$CMOD - Section Size: 58 Offset: 0 Address: 7D0EE000
   Compiled on: 2011.080
   Original File Name: SYS11080.T211043.RA000.IALJBLNK.HDROUT.H01
 DXCBBMMP - Section Size: 30 Offset: 58 Address: 7D0EE058
   Compiled on: 2011.026
   Original File Name: IALJB.PM22459.DXCOBJ3(DXCBBMMP)
 $IS$HLLE - Section Size: 58 Offset: 88 Address: 7D0EE088
   Compiled on: 2010.270
  Original File Name: DXC.V2R4M1.DXCOBJ1(DXCBHLLE)
 DXCBEXIT - Section Size: 30 Offset: 7F8 Address: 7D0EE7F8
   Compiled on: 2010.270
   Original File Name: DXC.V2R4M1.DXCOBJ1(DXCBHLLE)
 DXCBABRT - Section Size: 28 Offset: 828 Address: 7D0EE828
   Compiled on: 2010.270
   Original File Name: DXC.V2R4M1.DXCOBJ1(DXCBHLLE)
 DXCBENTD - Section Size: 30 Offset: 850 Address: 7D0EE850
   Compiled on: 2010.270
   Original File Name: DXC.V2R4M1.DXCOBJ1(DXCBHLLE)
 DXCBBCAL - Section Size: 548 Offset: 880 Address: 7D0EE880
   Compiled on: 2011.080
   Original File Name: IALJB.PM22459.DXCOBJ3(DXCBBCAL)
 DXCBSERE - Section Size: 440 Offset: DC8 Address: 7D0EEDC8
   Compiled on: 2008.128
   Original File Name: DXC.V2R4M1.DXCOBJ1(DXCBSERE)
 __errno - Section Size: 10 Offset: 1208 Address: 7D0EF208
   Compiled on: 2008.079
   Original File Name: PP.ADLE370.ZOS110.SCEELKEX(EDC4@156)
 strerror - Section Size: 10 Offset: 1218 Address: 7D0EF218
 More...
```

Normal response

Sample output when the ADDRESS operand is used:

```
User: zdmod a-7d0fa000

System:
                    Lines      1 to     17 of     17  Columns   1 to  79 of 79
                    Active:  5   4   3   2  *1*
 DXC0001I    02:30:16
 DXC5500I Mapping Data obtained from BMMP records loaded with module
                    JBZDMOD
                    Program Address                                    7D0EE000
                    Program Size                                       000467C8

 czdmod.c - Section Size: 10790 Offset: AC70 Address: 7D0F8C70
   Compiled on: 2011.080
   Offset    Address   Function Name
   00001358  7D0F9FC8  sort__3stdHQ2_3std6_PtritXTP13ZdmodFunctionTiTPP13ZdmodF
                       unctionTRP13ZdmodFunctionTPP13ZdmodFunctionTRP13ZdmodFun
                       ction_18ZdmodFunctionsSort_Q2_3std6_PtritXTP13ZdmodFunct
                       ionTiTPP13ZdmodFunctionTRP13ZdmodFunctionTPP13ZdmodFunct
                       ionTRP13ZdmodFunction_T118ZdmodFunctionsSort_v

 7D0FA000 is at offset 00001390 into section czdmod.c

 END OF DISPLAY
```

ZDPDU -- Display status of emergency pool recovery facility

Use the ZDPDU command -- ***from any terminal*** -- to display the emergency pool recovery status for each long-term pool size. The format of the command is:

Normal response

The normal response to the ZDPDU command is to display the emergency pool recovery status for each long-term pool size:

```
DXC8831I  Size  Status  Log stream name  Release count  Dispense count  FA count
          Ls    status  name             c1             c2              c3
          .
          .
          .
```

There is one line of status information for each long-term pool size, where:

s
Long-term pool size, 1 to 8.

status
Emergency pool recovery status, one of:

Collecting
ALCS is recording released file addresses in the MVS log stream.

Dispensing
ALCS is redispensing file addresses from the MVS log stream.

Disabled
ALCS cannot use the MVS log stream, due to a previous error condition.

name
Name of the MVS log stream for this pool size.

c1
Count of released file addresses that have been recorded in the MVS log stream since ALCS was started, or since the last Recoup.

c2
Count of released file addresses that have been redispensed from the MVS log stream since ALCS was started, or since the last Recoup.

c3
Estimated count of released file addresses currently in the MVS log stream. After an ALCS restart, if this count is zero, then ALCS is inserting released file addresses in the initial 4KB block that will be written to the MVS log stream (approximately 1000 file addresses).

The normal response from ZDPDU when emergency pool recovery is not supported by this ALCS is:

```
DXC8830I  PDU not supported
```

ZDPRG -- Display program

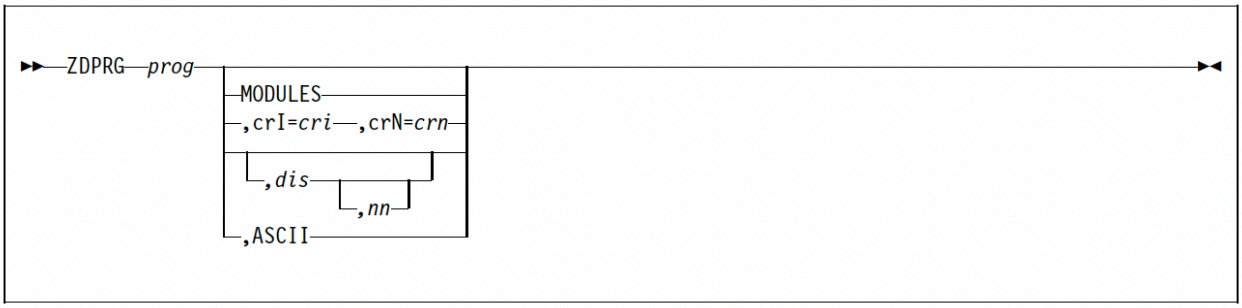
Use the ZDPRG command to display the contents of an ALCS application program. ZDPRG can display either the system-wide version or a test version of the program.

Use the ZDPRG command -- ***from any terminal*** -- to display the version of the program for that terminal.

Use the ZDPRG command ***from any CRAS authorized terminal*** -- to display the version of the program for another terminal.

This command invokes the ALCS scrolling function, see "Scrolling displays" on page 58 for details.

The format of the command is:



Where:

prog
The 4-character name of the ALCS application program.

crI=*cri*|(crN=*crn*
Use this parameter to specify the terminal whose version of the program is to be displayed (the test version, if there is one; otherwise the system-wide version). Omit this parameter to display the version for the input terminal.

Where:

cri
CRI of the terminal (6 hexadecimal digits).

crn
CRN of the terminal (8 alphanumeric characters).

Specify **CRI**=000000 to display the system-wide version.

dis
Listing address (displacement in program). It prints as LOC in assembly listings. You can omit leading zeros. Omit *dis* to display the program starting at displacement 0.

nn
Number of fullwords to be displayed. This must be a value from 1 to 16. Omit this parameter to display the whole program.

ASCII
Interpret the binary data as ASCII characters. Omit this parameter to interpret the data as EBCDIC characters.

MODULES
List all active load modules containing the specified program.

Normal responses

Normal response

A hexadecimal and character display of the application program formatted as fullwords in four columns with up to four rows.

```
DXC8221I CMD i  hh.mm.ss DPRG
Vector   Program   Version   Module     CRN
vector   prog      vv        module     crn
```

Where:

vector
Transfer vector name

prog
Program name

vv
Version number

module
Name of the application load module that contains the program

crn
Communication resource name of the terminal that loaded the module with test status, or *SYSTEM* if the module is system wide

Normal response

A list of all active modules containing program *prog*:

```
DXC5322I CME i  hh.mm.ss DPRG
Program   Version     Module      CRN      Load Date/Time
prog      vv          module      crn
      yyyy.ddd hh.mm.ss.lll
```

Where:

prog
Program name.

vv
Version number.

module
Name of the application load module that contains the program.

crn
Communication resource name of the terminal that loaded the module with test status, or *SYSTEM* if the module is system wide.

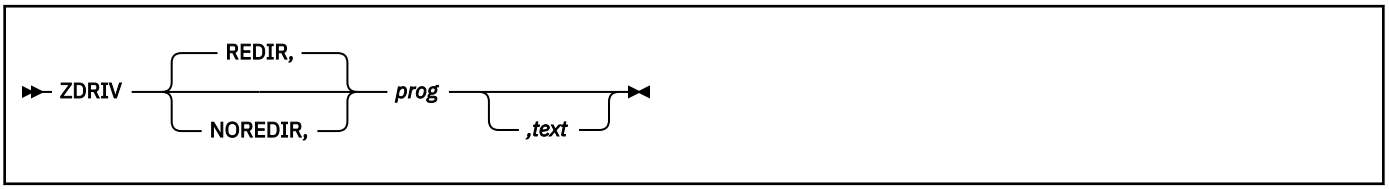
yyyy.ddd
Year, and day number of the year.

hh.mm.ss.lll
Time in hours, minutes, seconds, and milliseconds

ZDRIV -- Activate application program

Use the ZDRIV command -- ***from any CRAS authorized terminal*** -- to activate an ALCS application program.

The format of the command is:



Where:

prog
The 4-character name of the ALCS application program or transfer vector.

text
Text (up to 350 characters) to pass to the program as an input message on level 0 of the ECB in IMSG format.

REDIR|NOREDIR
By default, the ZDRIV command redirects response messages to a printer. This is either the associated printer for the originating terminal, or RO CRAS.

For some application programs which are activated by ZDRIV, it is better to direct response messages to the originating terminal. For such applications, specify NOREDIR to suppress redirection of response messages.

REDIR is the default.

"Program driver" on page 382 describes this command in more detail.

ZDSEQ -- Display sequential file status

Use the ZDSEQ command -- ***from any CRAS authorized terminal*** -- to display sequential file status.

This command invokes the ALCS scrolling function. See “Scrolling displays” on page 58 for details.

The format of the command is:

Where:

seq
Symbolic name of the sequential file. Omit *seq* to display the status of all sequential files.

ALLOC
Display only files that are not closed.

Normal responses

```
DXC8233I CMD i  hh.mm.ss DSEQ
File    Device     Status    Type     Volume    Data-set-name
seq     device     status    type     volser    dsname
DISP=disp                       b  Blocks read/written
⋮
```

Where:

seq
Symbolic name of the sequential file.

device
One of:

SYSOUT=*c*
Sysout file, *c* is the output class.

DUMMY
Dummy file.

USES *alt*
seq is a synonym for *alt*.

unit
Unit specification from file definition.

status
One of:

OPEN
Open and in use.

RESERVE
Open but not in use.

STANDBY
Standby data set.

CLOSED
Closed.

type
One of:

DIAG
ALCS diagnostic file.

LOG
ALCS update log file.

DCR
Data collection sequential file.

USER
User sequential file.

RT
Real-time sequential file.

IN
Input general sequential file.

OUT
Output general sequential file.

volser
Volume serial of first volume in data set.

dsname
Data set name.

disp
Data set disposition. (This information is only included when you specify *seq*.)

b
Block count. (This information is only included when you specify *seq*.)

The following warning is attached when *seq* is LOG and full logging of record updates is active.

```
WARNING - LOGALL is active, all database updates are being logged
```

Note: If ALCS updates the sequential file configuration table while ZDSEQ is building this response, an inconsistent response message will be produced. If this happens, repeat the ZDSEQ command.

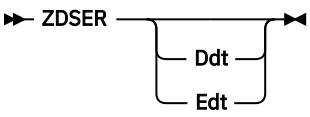
ZDSER -- Display system error options

Use the ZDSER command -- ***from any CRAS authorized terminal*** -- to display the ALCS system error options for both control (CTL) and operator (OPR) dumps.

The system error options determine the action taken in case of error. The initial settings for the options are specified in the ALCS generation. This is described in detail in *ALCS Installation and Customization*.

You can modify the system error options using the ZASER command. This is described in "ZASER -- Alter system error options" on page 94.

The format of the command is:



ZDSER with no parameters displays the system error options.

Where:

Ddt
Display Duplicate Dump table.

Edt
Display Exception Dump table.

Normal response

The response to ZDSER without any options:

```
DXC8583I CMD i  hh.mm.ss DSER  System error dump option
Option                CTL-Dumps      OPR-Dumps
Duplicate dumps.......aaa            aaa
Monitor tables dump...aaa            aaa
Entry storage dump....aaa            aaa
Global area...........aaa            aaa
Dump message..........aaa            aaa
Nodump message........aaa            aaa
VFA buffers dump......aaa            aaa
```

Where *aaa* are the current options for each type of dump.

Refer to "ZASER -- Alter system error options" on page 94 for a description of these options.

The response to ZDSER `Ddt`:

```
DXC8275I CMD i hh.mm.ss DSER  Duplicate Dump Table
Nbr Prog Ver Error  Off  Sup       Nbr Ver Prog Error  Off  Sup
nnn pppp vv  eeeeee oooo ssss      ccc vv  pppp eeeeee oooo ssss
⋮
```

Where:

nnn
Entry number in Duplicate Dump Table

pppp
Program name

vv
Program version

eeeeee
System error code

oooo
Offset in the program (or **** if the offset cannot be determined)

ssss
Number of duplicate system error messages suppressed since the last one was sent (or **** if the number of suppressed messages is more than 9999).

The response to ZDSER Edt:

```
DXC8279I CMD i  hh.mm.ss DSER  Exception Dump Table
Nbr  Prog      Error  Cur Freq   Nbr  Prog      Error  Cur Freq
nnn  pppp  ttt-eeeeee ccc fff    nnn  pppp  ttt-eeeeee ccc fff
⋮
```

Where:

nnn
Entry number in Exception Dump Table

pppp
Program name (generic names include trailing * characters)

ttt
Type of dump (CTL or OPR), or blank for both types

eeeeee
System error code (or ****** for all system error codes)

ccc
Current number of nodumps for this program and system error combination

fff
Frequency of dumps for this program and system error combination

ZDSYS -- Display current system state

Use the ZDSYS command -- ***from any terminal*** -- to display the ALCS system operating state.

The format of the command is:

Normal responses

```
DXC8238I CMD i  hh.mm.ss DSYS  ALCS in xxxx state


DXC8239I CMD i  hh.mm.ss DSYS
ALCS state change from xxxx to xxxx in progress
```

Where *xxxx* is one of the following:

HALT
Halt state

IDLE
Idle state

CRAS
CRAS state

MESW
Message switching state

NORM
Normal state

These states are described in “Operational states” on page 2.

ZDTIM -- Display current time and date

Use the ZDTIM command -- ***from any terminal*** -- to display time and date information.

The format of the command is:

Normal response

```
DXC8649I CMD i  hh.mm.ss DTIM
            yyddd   yyyy.mm.dd   hh.mm.ss
ALCS local  yyddd   yyyy.mm.dd   hh.mm.ss   date
MVS  local  yyddd   yyyy.mm.dd   hh.mm.ss   date
ALCS GMT    yyddd   yyyy.mm.dd   hh.mm.ss   date
MVS  GMT    yyddd   yyyy.mm.dd   hh.mm.ss   date
```

Where:

ALCS local
ALCS local time

MVS local
MVS local time

ALCS GMT
ALCS GMT

MVS GMT
MVS GMT

hh.mm.ss
Hours, minutes, and seconds (24-hour clock format)

dd.mm.yy
Day, month, and year

yyddd
Year, and day number in the year (the Julian date)

yyyy.mm.dd
Year, month and day

hh.mm.ss
Hours, minutes, and seconds (24-hour clock format)

date
Date in the installation default format

See *ALCS Installation and Customization* for further details on your installation default date format.

ZDUMP -- Request a system error dump

Use the ZDUMP command -- ***from any CRAS authorized terminal*** -- to take an ALCS system error dump to the ALCS diagnostic file. This command does not terminate ALCS. However, it does affect system performance, so use it with care.

The format of the command is:

Where:

ALL
Override current system error options and dump as if all options are selected. Also include the online program tables.

Partial
Dump according to the current CTL system error option settings (enter ZDSER to display the current settings) and include entry storage for all entries.

text
Any text explaining reason for requesting the dump. It appears on the RO CRAS and diagnostic file output, and is part of the response to the originating display.

Normal response

The response, to the RO CRAS and originating terminal is:

```
DXC8350I CMD i  hh.mm.ss DUMP
SE-nnnnnn CTL-000000 PSW-pppppppp pppppppp
VOLUME=volser DSNAME=dsname
MSG='ZDUMP ALL text'
```

Where:

nnnnnn
System error number.

CTL-000000
System error number (see *ALCS Messages and Codes*).

pp....pp
Corrected program status word (see *ALCS Messages and Codes*).

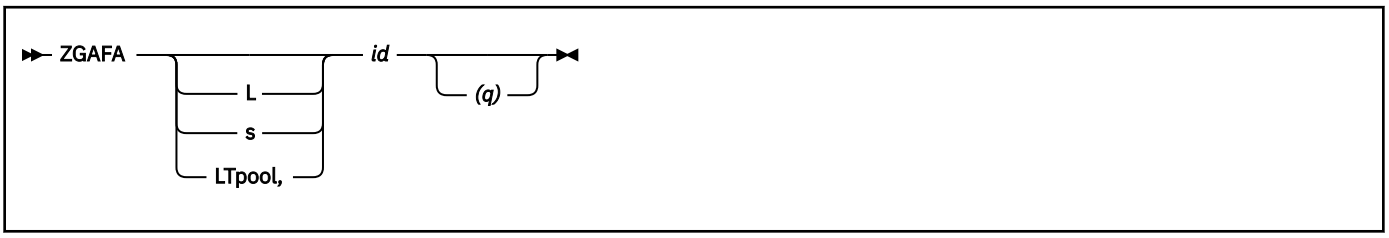
volser
Volume serial number of the diagnostic file that contains the dump.

dsname
Data set name of the diagnostic file that contains the dump.

ZGAFA -- Get an available pool file address

Use the ZGAFA command -- ***from any CRAS authorized terminal*** -- to obtain a long-term pool file address and write an initialized record at the address. The record ID that you specify is written into the first two bytes and the record is filed.

The format of the command is:



Where:

L*s*LTpool
Pool file identifier, where L*s* identifies the record size. You must specify this parameter when the record ID is defined in more than one pool, but if the record ID is unambiguous you can omit the pool file identifier.

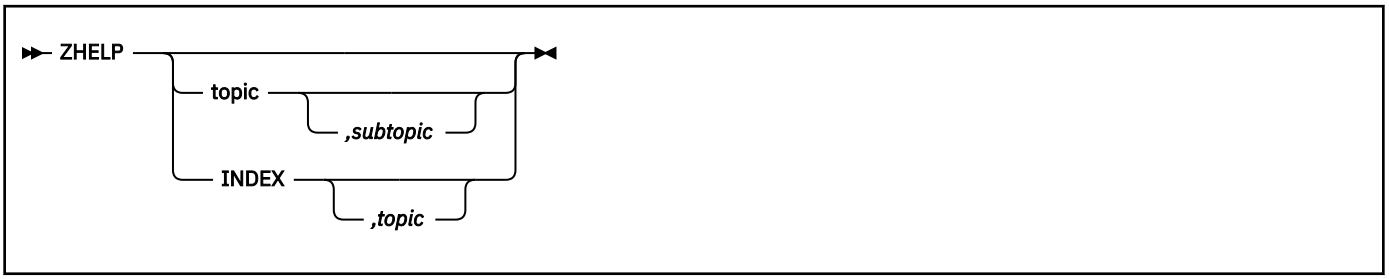
id
The record ID, either as 2 alphanumeric characters or as 4 hexadecimal digits.

q
Optional record ID qualifier. One digit from 0 to 9.

ZHELP -- Command help facility

Use the ZHELP command -- from ***any terminal*** -- to obtain help information.

The format of the command is:



Where:

topic
The primary help topic, 1 through 8 characters.

Valid help topics include help topics included with the ALCS shipment and can also include topics provided specifically for your installation.

For ALCS commands, you can specify the command action code (for example ZASYS) as the primary topic.

If you omit the primary help topic, ZHELP displays help information appropriate for the current context if there is one. If not, ZHELP with no parameters displays help information about the ZHELP command.

subtopic
Secondary help topic, one through eight characters.

For some help topics, you can get additional or related help information by providing a secondary help topic. This applies to some help topics provided with ALCS and can also apply to topics specifically for your installation.

INDEX
An index of all the available topics of help text that have been installed at your installation. If you include *topic*, displays an index of subtopics for *topic.*

Normal response

Use ZHELP INDEX to display a menu of help information.

```
DXC8310I CMD i  hh.mm.ss HELP INDEX

ZHELP COMMS             -- Communication related commands
ZHELP CONV     ADSTOP   -- Conversational trace -- ADSTOP
ZHELP CONV     ANYSTOP  -- Conversational trace -- ANYSTOP
ZHELP CONV     BRANCH   -- Conversational trace -- BRANCH
ZHELP CONV     DETAIL   -- Conversational trace -- DETAIL
ZHELP CONV     DISPLAY  -- Conversational trace -- DISPLAY
ZHELP CONV     EXIT     -- Conversational trace -- EXIT
ZHELP CONV     FLIP     -- Conversational trace -- FLIP
ZHELP CONV     FLUSH    -- Conversational trace -- FLUSH
ZHELP CONV     GET      -- Conversational trace -- GET
ZHELP CONV     HELP     -- Conversational trace -- HELP
ZHELP CONV     PROCESS  -- Conversational trace -- PROCESS
ZHELP CONV     REFSTOP  -- Conversational trace -- REFSTOP
ZHELP CONV     REGSTOP  -- Conversational trace -- REGSTOP
ZHELP CONV     REL      -- Conversational trace -- REL
ZHELP CONV     RUNSTOP  -- Conversational trace -- RUNSTOP
ZHELP CONV     SET      -- Conversational trace -- SET
ZHELP CONV     SKIP     -- Conversational trace -- SKIP
ZHELP CONV     STEP     -- Conversational trace -- STEP
ZHELP CONV     SUBSTEP  -- Conversational trace -- SUBSTEP
ZHELP CONV     SWAP     -- Conversational trace -- SWAP
ZHELP FILE              -- Pool file related commands
ZHELP HFS      CHDIR    -- HFS -- Change the current directory
ZHELP HFS      CLEAR    -- HFS -- Clear Screen
ZHELP HFS      COPY     -- HFS -- Copy files and programs
ZHELP HFS      DIR      -- HFS -- List directories, files, and programs
ZHELP HFS      ERASE    -- HFS -- Delete files, and programs
ZHELP HFS      MKDIR    -- HFS -- Create directories
ZHELP HFS      MKPRG    -- HFS -- Create and change programs
ZHELP HFS      MOVE     -- HFS -- Move directories, files, and programs
ZHELP HFS      RECEIVE  -- HFS -- Down-load files from ALCS HFS to PC
ZHELP HFS      RENAME   -- HFS -- Rename directories, files, and programs
ZHELP HFS      RMDIR    -- HFS -- Delete directories
ZHELP HFS      SEND     -- HFS -- Up-load files from PC to ALCS HFS
ZHELP PERF              -- Performance related commands
ZHELP PLMT              -- Purge LMT queue to PLM sequential file (CRAS only)
ZHELP RLMT              -- Repeat last LMT message (CRAS only)
ZHELP SEQF              -- Sequential file related commands
ZHELP SINE              -- IPARS Agent Sine codes
ZHELP SLC               -- SLC related commands
⋮
```

Otherwise, you receive a screen showing help information for the requested command or topic.

Using help commands

Notes:

1. ZHELP xxx INDEX is exactly equivalent to ZHELP INDEX xxx. ZHELP treats *xxx* as the primary help topic.
2. If you specify INDEX the help topic can include a trailing *. This * is treated as a wild card and allows you to display a subset of the available topics. For example you can display an index that shows the ALCS commands with:

   ```
   ZHELP Z* INDEX
   ```

3. If you enter a ZHELP command with a valid primary topic but an invalid secondary topic then ZHELP ignores the secondary help topic.
4. If you omit the secondary help topic or specify an invalid secondary topic (which ZHELP ignores) then the resulting display depends on what help information exists.

   If help information exists for the primary topic on its own then ZHELP will display that help information.

   If help information exists for the primary help topic with secondary help topics then ZHELP will display the index for the primary help topic.

For example if the help is available for the following topics:

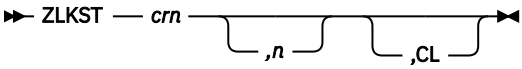
```
RES     AVAIL
RES     SCHEDULE
RES     BOOKING
RES     CLASSES
MESW
MESW    ADDRESS
MESW    QUEUES
```

then ZHELP RES or ZHELP RES ZZZ will display an index showing the RES secondary help topics. ZHELP MESW or ZHELP MESW ZZZ will display the MESW help information.

ZLKST -- Display statistics for an SLC link or channel

Use the ZLKST command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1--16 authorized terminal only*** -- to display statistics for an SLC channel or link, and optionally to reset the counts to zero.

The format of the command is:



Where:

crn
CRN of the link.

n
Number of the channel within the link; a number from 1 to 7. Omit *n* to display statistics for all channels on the link.

CL
Clear the counts (reset them to zero).

Normal responses

```
DXC8985I CMD i hh.mm.ss LKST  Statistics for SLC link CRN--crn
channel n    All LCB counts are zero
All message counts are zero
```

or:

```
DXC8985I CMD i hh.mm.ss LKST  Statistics for SLC link CRN--crn
Channel n     INPUT   OUTPUT
lcb            aaaaaa bbbbbb
lcb            aaaaaa bbbbbb
⋮

NCB            nnnnnn oooooo

Single Multi  Chars  Blocks
TYPE A I/P  cccccc dddddd eeeeee ffffff
TYPE A O/P  cccccc dddddd eeeeee ffffff
TYPE B I/P  cccccc dddddd eeeeee ffffff
TYPE B I/P  cccccc dddddd eeeeee ffffff
⋮
```

Repeated for each channel of the link.

Where:

crn
CRN of the link.

n
Number of the channel within the link.

lcb
Type of link control block (LCB). (See Table 17 on page 206 for a description of the LCB types.)

aaaaaa
Count of input LCBs.

bbbbbb
Count of output LCBs.

nnnnnn
Count of input NCBs.

oooooo
Count of output NCBs.

cccccc
Count of single block messages.

dddddd
Count of multiblock messages.

eeeeee
Count of characters.

ffffff
Count of blocks.

Table 17. Types of link control block (LCB)

lcb	**Description**
ACK	Positive acknowledgement
AML	Acknowledge (Clear) message label
ENQ	Enquiry
ERR	Unrecognized LCB type
ILB	Idle line block
NKP	Negative acknowledgment -- Invalid block
NKS	Negative acknowledgment -- Out of sequence block
RSA	Resume sending on all channels
RS1	Resume sending on channel 1
RS2	Resume sending on channel 2
RS3	Resume sending on channel 3
RS4	Resume sending on channel 4
RS5	Resume sending on channel 5
RS6	Resume sending on channel 6
RS7	Resume sending on channel 7
STA	Stop sending on all channels
ST1	Stop sending on channel 1
ST2	Stop sending on channel 2
ST3	Stop sending on channel 3
ST4	Stop sending on channel 4
ST5	Stop sending on channel 5
ST6	Stop sending on channel 6

Table 17. Types of link control block (LCB) (continued)	
lcb	**Description**
ST7	Stop sending on channel 7

ZLKTR -- Control SLC link trace

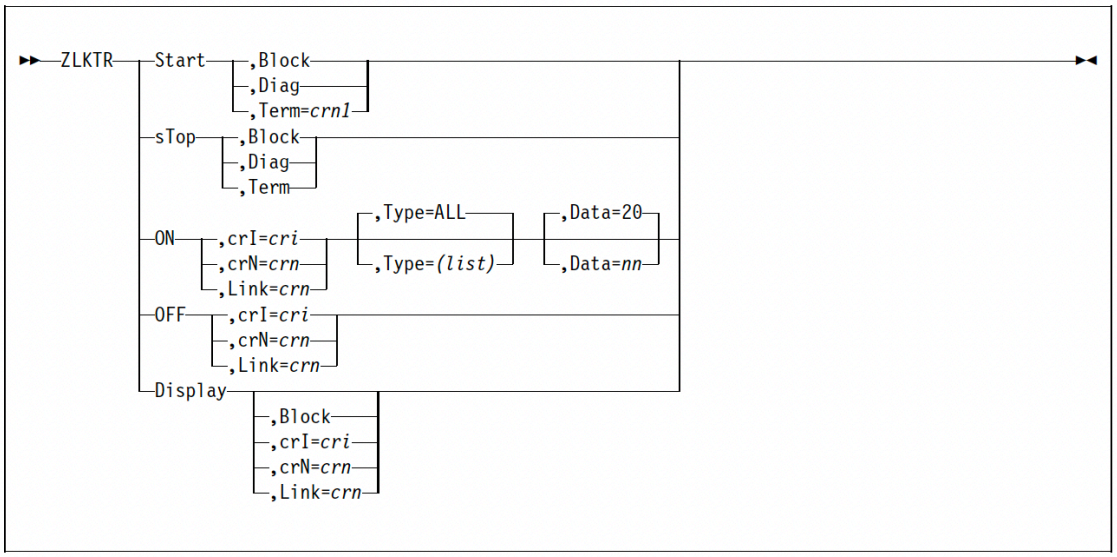
Use the ZLKTR command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1--16 authorized terminal only*** -- to trace activity on one or more SLC links.

You can trace to:

- The SLC link trace block
- The ALCS diagnostic file
- A terminal

You can trace to all these destinations simultaneously.

The format of the command is:



The `Start` and `sTop` keywords control whether or not ALCS traces SLC link activity. The `ON` and `OFF` keywords select which links (and traffic types) to trace. ALCS does not generate any trace output unless at least one `ZLKTR ON` command has been issued. A possible sequence of commands might be:

```
ZLKTR ON CRN=LINK1 TYPE=(A,B)
ZLKTR ON CRN=LINK2 TYPE=L
ZLKTR ON CRN=LINK3
ZLKTR START DIAG
ZLKTR OFF CRN=LINK1
ZLKTR OFF CRN=LINK3
ZLKTR ON CRN=LINK4 TYPE=N
ZLKTR START BLOCK
ZLKTR STOP BLOCK
ZLKTR STOP DIAG
ZLKTR OFF CRN=LINK2
ZLKTR OFF CRN=LINK4
```

Where:

Start,{Block|Diag|Term=*crn1*}
Begin tracing selected links and traffic types.

`Block`
Trace to the SLC link trace block

Diag
Trace to the diagnostic file.

Term=*crn1*
Trace to a terminal, where *crn1* is the CRN of the terminal. *crn1* must be a CRAS printer, or a CRAS display with an associated printer.

sTop,{Block|Diag|Term}
Stop tracing to:

Block
The SLC link trace block.

Diag
The diagnostic file.

Term
The terminal.

ON,{crI=*cri*|crN=*crn*|Link=*crn*}[,Type=(*list*)|ALL][,Data=*nn*]
Select an SLC (and traffic type) for tracing, where:

cri
CRI of the link.

crn
CRN of the link.

Type=(*list*)
Trace selected types of traffic; one or more of:

A
Trace Type A traffic.

B
Trace Type B traffic.

N
Trace network control blocks (NCBs).

L
Trace link control blocks (LCBs).

The parentheses can be omitted when only one traffic type is specified.

A comma or space must separate each traffic type.

Type=ALL
Trace all types of traffic (this is the default).

Data=*nn*
Amount of data to be traced, in bytes. A decimal number between 4 and 63; default is 20.

OFF,{crI=*cri*|(crN=*crn*|(Link=*crn*}
Deselect an SLC link, where:

cri
CRI of the link.

crn
CRN of the link.

Display[,Block|crI=*cri*|(crN=*crn*|(Link=*crn*]
Display trace status, where:

Display
`Display` with no other parameters shows the current status of trace to the SLC link trace block, to the diagnostic file, and to a terminal.

Block
Display current contents of the SLC link trace block.

cri
Display current trace options for a link, where *cri* is the CRI of the link.

crn
Display current trace options for a link, where *crn* is the CRN of the link.

The ZLKTR DISPLAY,BLOCK command invokes the ALCS scrolling function, see "Scrolling displays" on page 58 for details. "SLC link trace facility" on page 383 describes this command in more detail.

Normal responses

```
DXC8995I CMD i hh.mm.ss LKTR  SLC link trace status for CRN--crn
OFF


DXC8994I CMD i hh.mm.ss LKTR  SLC link trace status for CRN--crn
ON
Type ttt...ttt
Length dd
```

Where:

crn
CRN of the link.

ttt...ttt
ALL or any combination of A, B, N, L.

dd
Length of data to be traced, in bytes

Normal response

To ZLKTR DISPLAY with no other parameters specified is either:

```
DXC8993I CMD i hh.mm.ss LKTR SLC link trace status
status  to DIAG
status  to block
Started to terminal CRN-crn
```

or:

```
DXC8992I CMD i hh.mm.ss LKTR SLC link trace status
status  to DIAG
status  to block
Stopped to terminal
```

Where:

status
Indicates current trace activity; it is either Started or Stopped.

crn
CRN of the terminal

Normal responses

To ZLKTR DISPLAY,BLOCK

```
DXC8990I CMD i hh.mm.ss LKTR  SLC trace block entries
hh.mm.ss.t LKT crn a b cc ddd ee MBI=fg-h LEN=iii DAT=jj...
:
```

Where:

hh.mm.ss.t
Time in hours, minutes, seconds and tenths of a second.

crn
CRN of the link.

a
Link channel number.

b
R (read) or W (write).

cc
Error index byte (read only).

ddd
Type of LCB, NCB, or MSG (message).

ee
Transmission sequence number (TSN) or acknowledge transmission sequence number (ATSN).

f
A (Type A message) or B (Type B message).

g
Message label.

h
Message block number.

iii
Block size (bytes).

jj...
First *nn* bytes of a message in hexadecimal. The number *nn* is specified on the ZLKTR data= parameter.

mmmmm
Number of trace entries lost because of link trace shutdown.

See "Trace to terminal" on page 211 for a description of the fields.

```
DXC8991I CMD i hh.mm.ss LKTR  No SLC link trace block entries in use
```

Trace to terminal

The following messages appear on the printer (CRAS printer, or printer associated with a CRAS display):

For incoming and outgoing message blocks and LCBs:

```
DXC2556I COM i hh.mm.ss.t LKT crn a b cc dddee MBI=fg-h LEN=iii DAT=jj...
```

For incoming data that has been discarded with no processing:

```
DXC2556I COM i hh.mm.ss.t LKT  crn a b cc Discarded data=jj...
```

Following a temporary shutdown to prevent system overload:

```
DXC2556I COM i hh.mm.ss.t LKT  Trace resuming after shutdown -- mmmmm items lost
```

See *ALCS Installation and Customization* for further information on communications management.

Trace to the ALCS diagnostic file

See the description of the SLC link trace facility in “SLC link trace facility” on page 383 for information about link trace output on the diagnostic file.

ZL0GF -- Log off a VTAM controlled terminal

Use the ZL0GF command to log the terminal off from ALCS. The format of the command is:

Normal response

The response depends on the type of terminal equipment at your site.

- For 3270 displays ZLOGF terminates a VTAM session and VTAM displays the USSTAB message.
- For ALC displays and for displays connected via the MQ Bridge or WAS Bridge ALCS displays a 'log-off OK' message.

ZLOGN -- Log on to ALCS with a different user ID

Use the ZLOGN command -- ***from a VTAM controlled resource*** -- to logon to ALCS with a different user ID.

The format of the command is:

Normal response

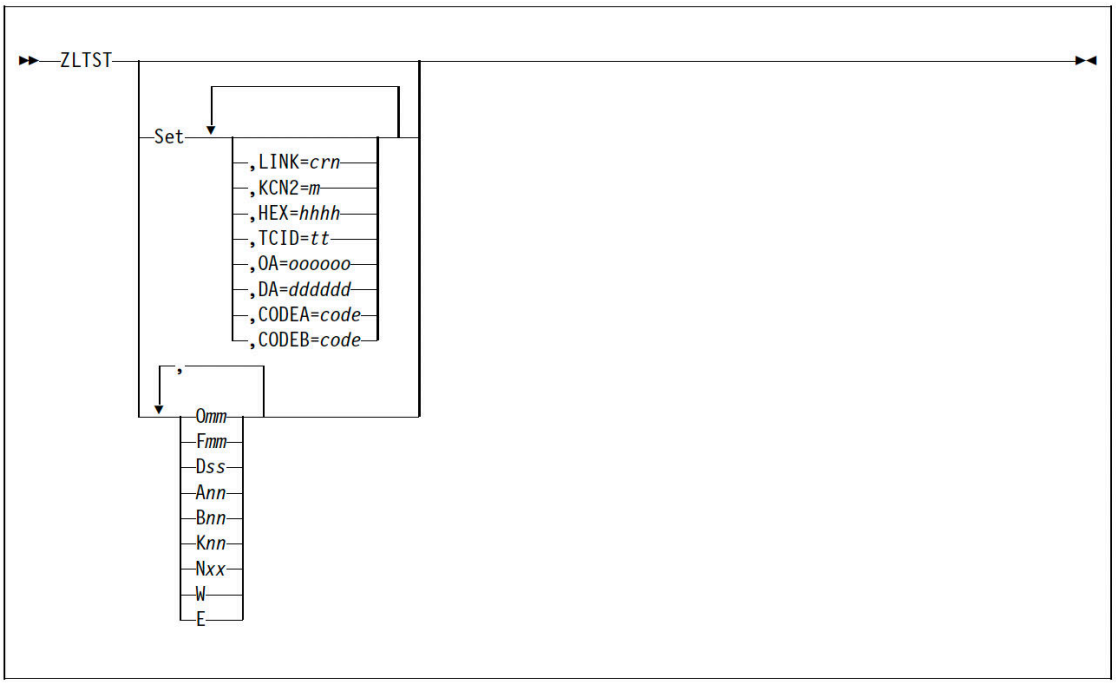
The normal response is to initiate the log-on dialog. This will vary between installations. Typically you will be asked for a user ID and a password but the precise format of the log-on dialog depends on your local procedures.

ZLTST -- Invoke an SLC link test

Use the ZLTST command -- ***from a Prime CRAS authorized terminal only*** -- to initiate an SLC test. These tests were designed to correspond to the tests that *P.1024 Test Guide*, SITA Document PZ.1885.3 describes.[2]

SITA no longer uses these tests, but you can use the ZLTST command for loop tests. For details see *ALCS Installation and Customization*.

The format of this command is:



Where:

ZLTST with no parameters and ZLTST SET with no further parameters display the current test values.

Set,*value*[,*value...*]

Set new values for one or more of the following test parameters. Any number of parameters can be specified on one ZLTST entry; separate each with a comma or space:

LINK=*crn*

Test link CRN. The default is the first SLC link defined in ALCS communications generation.

KCN2=*m*

Link channel number for test commands that require a second channel; see *P.1024 Test Guide*, SITA Document PZ.1885.3. Specify a decimal number in the range 2 through 7. The default is 2. The default link channel number for all test commands is 1.

HEX=*hhhh*

Test high level network (HLN) exit address; 4 hexadecimal digits. The default is 0000.

TCID=*tt*

Test terminal circuit identifier (TCID); 2 hexadecimal digits. The default is 00.

OA=*oooooo*

Origin address for test Type B messages (7 alphanumeric characters). The default is all blanks.

[2] Test commands 1, 3, and 15 are not implemented.

DA=*ddddddd*
Destination address for test Type B messages; 7 alphanumeric characters. The default is all blanks.

CODEA=*code*
Test translate code for Type A messages. Specify one of the following:

NONE
ALCS does not translate output messages.

ALC
Padded 6-bit ALC transmission code.

IATA5
Padded ATA/IATA 5-bit transmission code (default for CODEB).

IATA7
ATA/IATA 7-bit transmission code (default for CODEA).

IATA7-E
Extended ATA/IATA 7-bit transmission code.

ALC-E
Extended padded 6-bit ALC transmission code.

CODEB=*code*
Test translate code for Type B messages. The possible test translate codes are the same as for Type A messages. See description of CODEA.

Specify a maximum of 62 actions on one ZLTST entry; any of the following:

O*mm*
Turn on command number *mm*. *mm* is a 1- or 2-digit decimal number. Valid command numbers are 0 through 15, and 31. See Table 18 on page 216. Refer to *P.1024 Test Guide*, SITA Document PZ.1885.3 for a description of test commands 0 through 15 and the expected test results. While command 31 is turned on, the loop-bit for incoming link control blocks is not tested.

F*mm*
Turn off command number *mm*. *mm* is a 1- or 2-digit decimal number. Valid command numbers are 0 through 15, and 31. See Table 17 on page 206 for a list of acronyms and types of link control blocks. See Table 18 on page 216.

Table 18. Use of O and F actions in ZLTST

mm	**O*mm***	**F*mm***
0	Suppress ETB in output LDBs	Go back to normal
1	(Not implemented)	(Not implemented)
2	Send one illogical ENQUIRY LCB on channel 1	--
3	(Not implemented)	(Not implemented)
4	Treat input LDBs as having parity error	Go back to normal
5	Suppress output LDB with TSN 3, but consider it transmitted	Go back to normal
6	Send one illogical NAK-PARITY	--
7	Send illogical STOP LCBs on channel 1	Stop sending illogical STOP LCBs
8	Send illogical RESUME LCBs on channel 1	Stop sending illogical RESUME LCBs
9	Do not set 'last block' on last block of multiblock message	Go back to normal

Table 18. Use of O and F actions in ZLTST (continued)		
mm	**O*mm***	**F*mm***
10	Send STOP LCBs on channel 1	Send RESUME LCBs on channel 1
11	Send STOP LCBs on channel 2	Send RESUME LCBs on channel 2
12	Send one STOP-ALL LCB	Send one RESUME-ALL LCB
13	Suppress DLE in output LCBs/LDBs	Go back to normal
14	Send STOP-CHANNEL-2 LCB on channel 1	Send RESUME-CHANNEL-2 LCB on channel 1
15	Do not send block number 3 of multiblock message	Go back to normal
31	Do not test loop-bit for input LCBs	Test loop-bit for input LCBs

D*ss*

ss is the time interval, in seconds, between successive actions on one ZLTST command. It is a 1- or 2-digit decimal number in the range 1 to 99.

A*nn*

Generate Type A message series *nn*. *nn* is a 1- or 2-digit decimal number in the range 1 to 17. Messages are sent directly to the SLC link. Refer to *P.1024 Test Guide*, SITA Document PZ.1885.3 for a description of these test messages and the expected test results.

B*nn*

Generate Type B message series *nn*. *nn* is a 1- or 2-digit decimal number in the range 1 to 17. Messages are sent from an application for transmission on the SLC link. Refer to *P.1024 Test Guide*, SITA Document PZ.1885.3 for a description of these test messages and the expected test results.

K*nn*

Generate Type B message series *nn*. *nn* is a 1- or 2-digit decimal number in the range 1 to 17. Messages are sent directly to the SLC link. Refer to *P.1024 Test Guide*, SITA Document PZ.1885.3 for a description of these test messages and the expected test results.

Table 19. Message series for Type A and Type B messages. See Table 20 on page 218 for the characteristics of the message series sent in response to *nn*.	
nn	**Message series**
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	1, 2, 3, 4, 5, 6, 7
10	1, 5, 2, 6, 3, 7, 4
11	10 to 29
12	10, 12, 14, 16, 18, 20, ... 44, 46, 48

Table 19. Message series for Type A and Type B messages. See Table 20 on page 218 for the characteristics of the message series sent in response to *nn*. *(continued)*	
nn	**Message series**
13	11, 13, 15, 17, 19, 21, ... 45, 47, 49
14	10 to 49
15	50 times message 2
16	1 to 49
17	Twice message 8

Table 20. Characteristics of Type A and Type B message series

Message number	**Type A**		**Type B**	
	Number of characters	**Number of LDBs**	**Number of characters**	**Number of LDBs**
1	1	1	33	1
2	50	1	50	1
3	150	1	150	1
4	240	1	240	1
5	241	2	241	2
6	480	2	480	2
7	500	3	500	3
8	3700	16	3700	16
9	200	1	200	1
10	100	1	102	1
11	300	2	302	2
12	105	1	107	1
13	305	2	307	2
14	110	1	112	1
15	310	2	312	2
16	115	1	117	1
17	315	2	317	2
18	120	1	122	1
19	320	2	322	2
20	125	1	127	1
21	325	2	327	2
22	130	1	132	1
23	330	2	332	2

Table 20. Characteristics of Type A and Type B message series (continued)

Message number	Type A		Type B	
	Number of characters	Number of LDBs	Number of characters	Number of LDBs
24	135	1	137	1
25	335	2	337	2
26	140	1	142	1
27	340	2	342	2
28	145	1	147	1
29	345	2	347	2
30	155	1	152	1
31	355	2	352	2
32	160	1	157	1
33	360	2	357	2
34	165	1	162	1
35	365	2	362	2
36	170	1	167	1
37	370	2	367	2
38	175	1	172	1
39	375	2	372	2
40	180	1	177	1
41	380	2	377	2
42	185	1	182	1
43	385	2	382	2
44	190	1	187	1
45	390	2	387	2
46	195	1	192	1
47	395	2	392	2
48	205	1	197	1
49	400	2	397	2

N*xx*
Generate a network control block (NCB). *xx* is the NCB command identifier. See *Synchronous Link Control Procedure*, SITA Document P.1124 for a description of the command identifier.

W
Wait until a good data block has been received.

E
Terminate a wait.

Normal responses

ALCS sends the following response to the RO CRAS when the test has completed.

```
DXC8316I CMD i hh.mm.ss LTST  End


DXC8322I CMD i hh.mm.ss LTST  Link test constants are set to
LINK  crn
KCN2  m
HEX   hhhh
TCID  tt
OA    ooooooo
DA    ddddddd
CODEA code
CODEB code
```

ALCS sends the following response to the originating display terminal.

```
DXC8039I CMD i hh.mm.ss LTST  Started
Program messages on printer - CRN--crn CRI--cri
```

ZMAIL -- Using ALCS e-mail facility

You can use the ZMAIL command to:

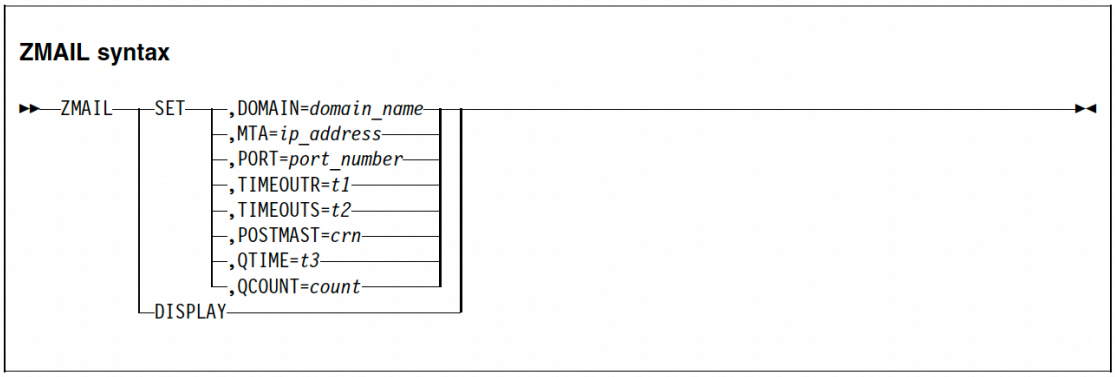
- Set and display e-mail operating values.
- Send an e-mail message from a 3270 terminal.
- Send an e-mail message from an ALC terminal.
- Control the outbound e-mail queue handler.

These operations are described in the following sections.

ZMAIL -- Set and display e-mail operating values

Use the ZMAIL command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal*** -- to set or display the e-mail operating values.

The format of the command is:



Where:

SET
Set or change the operating values for ALCS e-mail operation. Any parameters that you set or change with this command are not retained over an ALCS restart.

DOMAIN=*domain_name*
ALCS mail domain name, 1 to 64 characters. This overrides any value specified in the ALCS generation. See the description of the SCTGEN EMAILDOMAIN parameter in *ALCS Installation and Customization*.

MTA=*ip_address*
IP address of the local message transfer agent (MTA) for outbound e-mail. The IP address must be provided in dotted decimal notation. This overrides any value specified in the ALCS generation. See the description of the SCTGEN EMAILMTA parameter in *ALCS Installation and Customization*.

PORT=*port_number*
Port number of the local MTA for outbound e-mail (normally 25). This overrides any value specified in the ALCS generation. See the description of the SCTGEN EMAILPORT parameter in *ALCS Installation and Customization*.

TIMEOUTR=*t1*
Connection timeout for inbound e-mail, 1 to 256 seconds. This overrides any value specified in the ALCS generation. See the description of the SCTGEN EMAILTIMEOUTR parameter in *ALCS Installation and Customization*.

TIMEOUTS=*t2*
Connection timeout for outbound e-mail, 1 to 256 seconds. This overrides any value specified in the ALCS generation. See the description of the SCTGEN EMAILTIMEOUTS parameter in *ALCS Installation and Customization*.

POSTMAST=*crn*
Destination for inbound e-mail messages addressed to 'Postmaster@*domain_name*', where *domain_name* is your ALCS mail domain name. This overrides any value specified in the ALCS generation. See the description of the SCTGEN EMAILPOSTMASTER parameter in *ALCS Installation and Customization*.

QTIME=*t3*
Timeout in minutes for the e-mail queue handler. After adding a message to the outbound e-mail queue when the queue was previously empty, ALCS waits for the timeout interval to expire before sending messages from the queue. This overrides any value specified in the ALCS generation. See the description of the SCTGEN EMAILQTIME parameter in *ALCS Installation and Customization*.

QCOUNT=*count*
Threshold for the e-mail queue handler. ALCS waits for the number of messages on the outbound e-mail queue to reach the threshold value before sending messages from the queue. This overrides any value specified in the ALCS generation. See the description of the SCTGEN EMAILQCOUNT parameter in *ALCS Installation and Customization*.

DISPLAY
Display current operating values for the ALCS e-mail facility.

Normal responses

The normal response to ZMAIL DISPLAY:

```
DXC5010I CMD i  hh.mm.ss MAIL  E-mail operating values
ALCS mail domain - domain_name
MTA IP address   - ip_address
MTA port number  - port_number
Inbound timeout  - t1
Outbound timeout - t2
Postmaster name  - crn
```

Where:

domain_name
The ALCS mail domain name.

ip_address
The IP address of the local message transfer agent (MTA) for outbound e-mail.

port_number
The port number of the local MTA for outbound e-mail.

t1
The connection timeout for inbound e-mail.

t2
The connection timeout for outbound e-mail.

crn
The destination for inbound e-mail messages addressed to 'Postmaster@*domain_name*', where *domain_name* is your ALCS mail domain name.

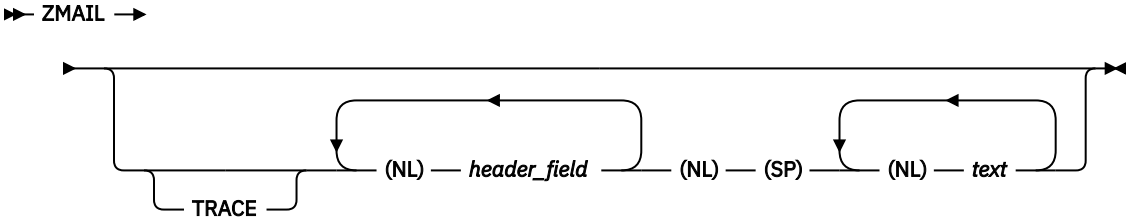
ZMAIL -- Send an e-mail from a 3270 terminal

Use the ZMAIL command -- ***from any IBM 3270 display terminal or compatible terminal*** -- to send an e-mail message. Ensure that your ALCS system is correctly configured for sending e-mail messages before sending a message with the ZMAIL command.

The e-mail message will be added to a queue before transmission. If the queue handler cannot send the message, it remains on the queue until the queue handler retries the transmission later.

The format of the command is:

ZMAIL syntax



Where:

The ZMAIL command with no header fields or message text displays a "template" message. You can update this template (inserting header field information and adding the message text) and then press enter to send the e-mail message.

TRACE
Send SMTP trace messages to the RO CRAS when the e-mail message is transmitted. These trace messages can help diagnose problems with connections to your local SMTP server (called the local message transfer agent (MTA)).

(NL)
New line.

(SP)
One or more space (blank) characters.

header_field
Each header field is comprised of a field name and field text. The Internet Request For Comments (RFC) *Standard for the Format of APRA Internet Text Messages*, RFC 822, specifies the basic standard for Internet e-mail content. This content includes standard header fields for routing information (such as the mail addresses of the originator and the intended recipients) and the text of the message. The header fields that can be used in the ZMAIL command are described below, together with guidance on the format of the corresponding field text.

text
The e-mail message should contain one or more lines of text. Each line can contain text or be left blank. If you want to include a blank line in your message, enter a line containing at least one space (blank) character.

Do not use characters which are not part of the standard US ASCII set (for example do not use the not symbol (¬) or split vertical bar (¦) characters). Also be aware that currency symbols may display differently on different equipment.

Note:

Like other ALCS commands, you can enter the command name 'ZMAIL' and the optional parameter 'TRACE' without worrying about case. For example, you can enter 'ZMAIL', 'zmail', or even 'ZmAiL' - they are all equivalent.

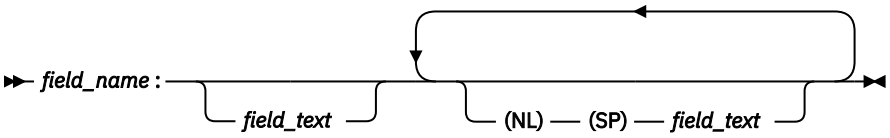
But unlike other ALCS commands, you enter the header fields and message text using the appropriate case. ZMAIL sends your message just as you enter it. For example, you might want to include a person's name with an initial upper case character followed by the rest in lower case, like this: 'Brian'.

E-mail header fields in ZMAIL command -- 3270 terminal

This section includes recommendations on the use of header fields in the e-mail message. It also includes information on conventions and restrictions that apply to header fields in the ZMAIL command.

The format of each header field is:

ZMAIL syntax



Where:

field_name
An e-mail header field name. This field name must start at the first character of the line, it must not contain space (blank) characters, and you must follow it immediately with a colon (:) character. A list of the recommended header fields is described below.

field_text
Text appropriate for the header field name which it follows. The field text for a particular header field can continue over several screen lines. If you do this, each continuation line must start with at least one space (blank) character.

Attention:

IBM 3270 and compatible terminals and emulators support a variety of different "code pages". These code pages associate the characters and symbols which appear on your keyboard and display with their internal binary representations.

Unfortunately, different code pages can use different binary representations for the same character.

This means that ALCS may not correctly recognize some characters which you enter from your keyboard - even though your display shows the character correctly.

ALCS interprets the binary representations according to the Latin 1 EBCDIC code page (01047). If your terminal or emulator is configured to use a different code page then you may need to use different characters when you enter e-mail. For example, if you are using a Brazilian EBCDIC code page (00273 or 00275) you must use the Ã character in place of @.

Recommended header fields

We recommend that you only use one or more of the following header fields in the ZMAIL command:

Name
Contents of header text

To:
E-mail address of the person you are sending the message to, or list of addresses if you are sending the message to more than one person.

Cc:
E-mail address of a person you are sending a copy of the message to, or list of addresses if you are sending copies to more than one person.

Bcc:
E-mail address of a person you are sending a "blind copy" of the message to, or list of addresses if you are sending blind copies to more than one person.

ALCS removes `Bcc:` header fields before it sends your message. This means that only the person who receives a blind copy knows that he or she has a copy.

From:
Your e-mail address. This is the address where you prefer correspondents to send your e-mail, it is not necessarily the e-mail address of your ALCS terminal. It could, for example, be the e-mail address you normally use on your corporate e-mail system.

Subject:
A few words to describe the subject of this message.

ALCS checks that header field names do not contain space (blank) characters and that a colon (:) character follows the field name. If you do not follow these rules, ALCS generates an error response and does not send your message.

You can enter header field names without worrying about case. For example, you can enter the field name 'From' as 'FROM', 'from', or 'FrOm' but you are recommended to follow the case conventions given above.

In some cases, you might want the recipient to send the reply to a different e-mail address - for example, to an e-mail address specifically reserved for customer communication. The normal way to do this is to include a 'Reply-to:' header line.

Required header fields

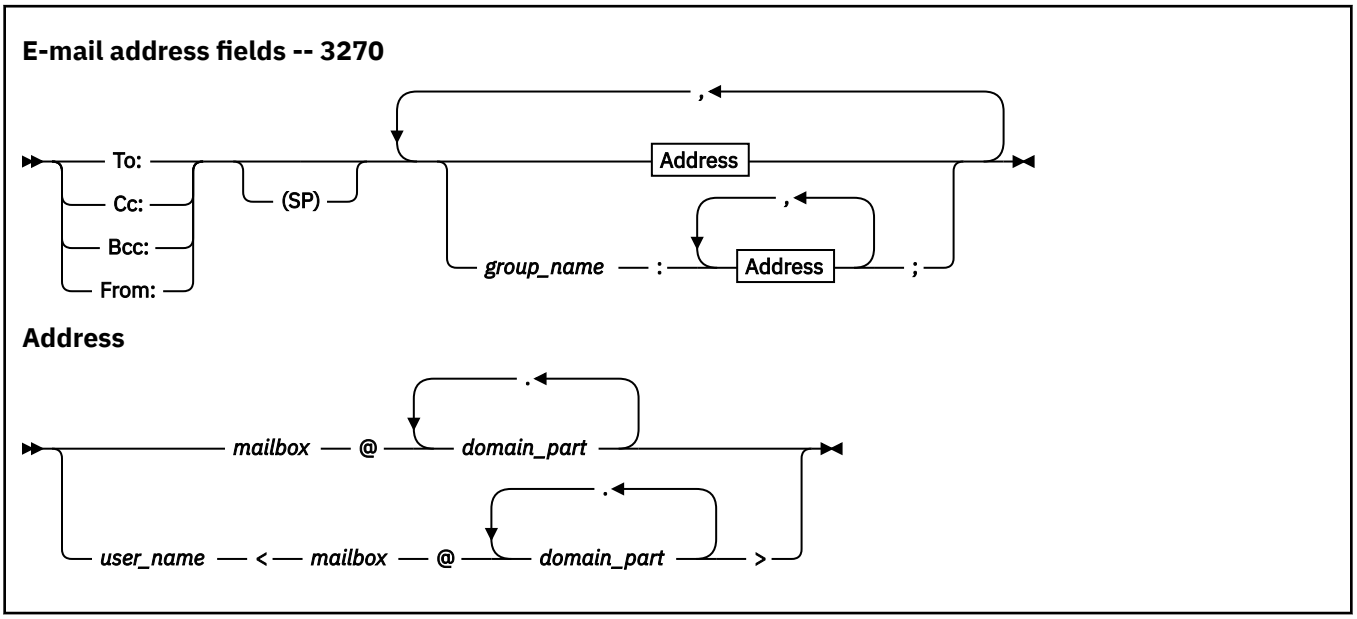
Specify at least one recipient for your message. You can specify recipients in `To:`, `Cc:`, or `Bcc:` header fields.

Empty header fields

ALCS checks for and removes any header fields with no field text (or field text that comprises only space characters) before it sends your message. This means that when you use the `ZMAIL` command to display a template message, you do not need to enter anything into header lines which you do not want to use.

Header fields containing e-mail addresses

ALCS supports the following formats for header fields which contain e-mail addresses. (ALCS does not actually check the format of the `From:` field, but we recommend you use the syntax shown.)



Where:

(SP)
One or more space (blank) characters.

group_name
Name of a group of people. This name is not used for electronic routing of the message but the recipient's e-mail software may display this name when it lists messages within their mailbox.

group_name can include space (blank) characters, but if it includes punctuation such as comma (,), period (.), semicolon (;), colon (:), and so on, then you must enclose it in quotes (").

user_name
Person's usual name. This name is not used for electronic routing of the message but the recipient's e-mail software may display this name when it lists messages within their mailbox.

user_name can include space (blank) characters, but if it includes punctuation such as comma (,), period (.), semicolon (;), colon (:), and so on, then you must enclose it in quotes (''). For example:

```
To: Gerry Fisher <gfisher@uk.ibm.com>
To: "Gerry Fisher" <gfisher@uk.ibm.com>
To: "Fisher, Gerry" <gfisher@uk.ibm.com>
To: "Gerry :-)" <gfisher@uk.ibm.com>
```

are valid, but:

```
To: "Gerry Fisher <gfisher@uk.ibm.com>
To: Fisher, Gerry <gfisher@uk.ibm.com>
To: Gerry :-) <gfisher@uk.ibm.com>
```

are not.

mailbox* @ *domain
Person's e-mail address. This address is used for electronic routing of the message. It comprises the following parts:

mailbox
Identifies the specific mail box within an organization. Be aware that some organizations use case-sensitive mail box names (so that 'Gerry' and 'gerry' are not the same mail box).

In the unlikely event that a mail box name contains spaces (blanks) or punctuation marks, you must enclose *mailbox* in quotes (").

domain
Identifies the specific organization where the mail box is. The domain part of an e-mail address is not case-sensitive. It typically comprises several parts (*domain_part*) joined by period (.) characters, for example: 'uk.ibm.com'.

You can include spaces (blanks) around and between the various parts of an e-mail address if you wish. For example, the following are all equivalent:

```
<gfisher@uk.ibm.com>
<gfisher @ uk.ibm.com>
< gfisher @ uk . ibm . com >
```

Comments in e-mail addresses

If you wish, you can include comments in e-mail addresses. A comment is any text enclosed in parentheses; it can be included anywhere that a space is allowed. For example, the following are all equivalent:

```
<gfisher@uk.ibm.com>
<gfisher(Gerry Fisher)@uk.ibm.com>
<gfisher @ uk (actually England) . ibm . com >
```

ALCS generated header fields

ALCS automatically generates and adds `Date:`,, `Message-ID:`, and `Sender:` header fields to your message. If you include these header fields in the `ZMAIL` command, ALCS replaces the ones you include.

ALCS also automatically generates and adds MIME version and MIME content type header fields to your message.

Other header fields

ALCS accepts `Resent-to:`, `Resent-cc:`, and `Resent-bcc:`. It processes these headers in the same way as `To:`,, `Cc:`, and `Bcc:`. Most e-mail messages do not need these "resent" fields.

You can include header fields that are not listed in this section. For example, you can include `In-reply-to:` or `Reference:`. ALCS does not check header fields that it does not recognize (except for removing unused headers).

If you include extra header fields, we recommend that you follow Internet standards, and in particular, if you invent your own header fields, we recommend that you prefix the names with 'X-'.

Using `ZMAIL` to send an e-mail message from a 3270 terminal

The following example shows a variety of e-mail address formats, all of which are allowed in the ALCS `ZMAIL` command:

```
zmail
To: gfisher @ uk.ibm.com
Cc: "Ms. Smith" (Ann) <arsmith @ uk.ibm.com>,
    Someone Else <"A. N. Other" @ someplace.com>
Bcc: <person@another.place.org>
From: tjones@uk.ibm.com
Subject: Successful sale

Hi,
I wanted to tell you about my successful hardware sale.
When can I phone you?

Brgds, Tom
```

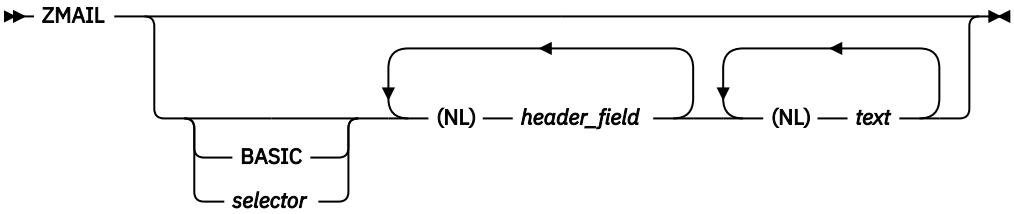
ZMAIL -- Send an e-mail from an ALC terminal

Use the ZMAIL command -- ***from any ALC display terminal*** -- to send an e-mail message. Ensure that your ALCS system is correctly configured for sending e-mail messages before sending a message with the ZMAIL command.

The e-mail message will be added to a queue before transmission. If the queue handler cannot send the message, it remains on the queue until the queue handler retries the transmission later.

The format of the command is:

ZMAIL syntax



Where:

The ZMAIL command with no header fields or message text displays a "template" message. You can update this template (inserting header field information and adding the message text) and then press enter to send the e-mail message.

BASIC
Use the default process for converting ALC e-mail message text. Use this parameter if you are not sure what custom formatting is implemented by your ALCS system, or if you want to override any custom formatting and use the formatting rules described in this book.

selector
Use the specified custom process for converting ALC e-mail message text. The allowed values for *selector* depend on what custom formatting is implemented by your ALCS system; refer to your organization's own documentation. For details see the description of the ALC e-mail custom conversion exit program ASM5 in *ALCS Installation and Customization*.

header_field
Each header field is comprised of a field name and field text. The Internet Request For Comments (RFC) *Standard for the Format of APRA Internet Text Messages*, RFC 822, specifies the basic standard for Internet e-mail content. This content includes standard header fields for routing information (such as the mail addresses of the originator and the intended recipients) and the text of the message. The header fields that can be used in the ZMAIL command are described below, together with guidance on the format of the corresponding field text.

text
The e-mail message should contain one or more lines of text. The first line of text must *not* start with a slash or space character.

See “Mixed case and punctuation in ZMAIL input from an ALC terminal” on page 232 for details of how you indicate where you want upper-case characters (capital letters).

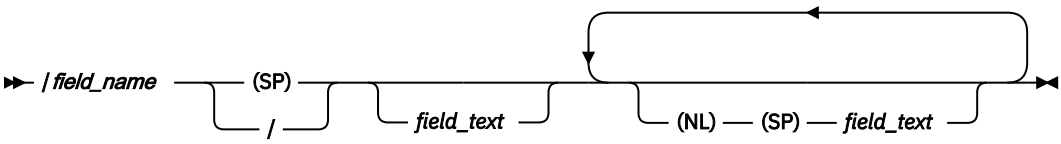
Do not use characters which are not part of the standard US ASCII set (for example do not use the lozenge (&lozenge.) or end-item (&bx1212.) characters). Also be aware that currency symbols may display differently on different equipment.

E-mail header fields in ZMAIL command -- ALC terminal

This section includes recommendations on the use of header fields in the e-mail message. It also includes information on conventions and restrictions that apply to header fields in the ZMAIL command.

The format of each header field is:

ZMAIL syntax



Where:

field_name
An e-mail header field name. You must prefix each header field name with a slash (/) character, which must be the first character of the line. The field name must not contain space (blank) characters, and you must follow it immediately with a space or slash character. A list of the recommended header fields is described below.

field_text
Text appropriate for the header field name which it follows. The field text for a particular header field can continue over several screen lines. If you do this, each continuation line must start with at least one space (blank) character.

Recommended header fields

We recommend that you only use one or more of the following header fields in the `ZMAIL` command:

Name
Contents of header text

/TO
E-mail address of the person you are sending the message to, or list of addresses if you are sending the message to more than one person.

/CC
E-mail address of a person you are sending a copy of the message to, or list of addresses if you are sending copies to more than one person.

/BCC
E-mail address of a person you are sending a "blind copy" of the message to, or list of addresses if you are sending blind copies to more than one person.

ALCS removes `/BCC` header fields before it sends your message. This means that only the person who receives a blind copy knows that he or she has a copy.

/FROM
Your e-mail address. This is the address where you prefer correspondents to send your e-mail, it is not necessarily the e-mail address of your ALCS terminal. It could, for example, be the e-mail address you normally use on your corporate e-mail system.

/SUBJECT
A few words to describe the subject of this message.

In some cases, you might want the recipient to send the reply to a different e-mail address - for example, to an e-mail address specifically reserved for customer communication. The normal way to do this is to include a '/REPLY-TO' header line.

Required header fields

Specify at least one recipient for your message. You can specify recipients in `/TO`, `/CC`, or `/BCC` header fields.

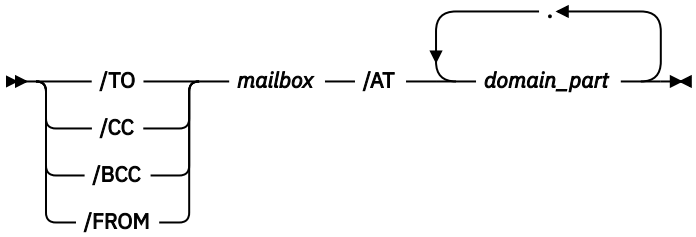
Empty header fields

ALCS checks for and removes any header fields with no field text (or field text that comprises only space characters) before it sends your message. This means that when you use the ZMAIL command to display a template message, you do not need to enter anything into header lines which you do not want to use.

Header fields containing e-mail addresses

ALCS supports the following formats for header fields which contain e-mail addresses. (ALCS does not actually check the format of the /FROM field, but we recommend you use the syntax shown.)

E-mail address fields -- ALC



Where:

mailbox* /AT *domain
Person's e-mail address. This address is used for electronic routing of the message. It comprises the following parts:

mailbox
Identifies the specific mail box within an organization. Be aware that some organizations use case-sensitive mail box names (so that 'Gerry' and 'gerry' are not the same mail box).

In the unlikely event that a mail box name contains spaces (blanks) or punctuation marks, you must enclose *mailbox* in quotes.

See "Mixed case and punctuation in ZMAIL input from an ALC terminal" on page 232 for information about entering upper-case characters and punctuation in e-mail addresses.

domain
Identifies the specific organization where the mail box is. The domain part of an e-mail address is not case-sensitive. It typically comprises several parts (*domain_part*) joined by period (.) characters, for example: 'UK.IBM.COM'.

You can include spaces (blanks) around and between the various parts of an e-mail address if you wish. For example, the following are equivalent:

```
GFISHER /AT UK.IBM.COM
GFISHER /AT UK . IBM . COM
```

ALCS generated header fields

ALCS automatically generates and adds /DATE,, /MESSAGE-ID, and /SENDER header fields to your message. If you include these header fields in the ZMAIL command, ALCS replaces the ones you include.

ALCS also automatically generates and adds MIME version and MIME content type header fields to your message.

Other header fields

ALCS accepts /RESENT-TO, /RESENT-CC, and /RESENT-BCC. It processes these headers in the same way as /TO, /CC, and /BCC. Most e-mail messages do not need these "resent" fields.

You can include header fields that are not listed in this section. For example, you can include /IN-REPLY-TO or /REFERENCE. ALCS does not check header fields that it does not recognize (except for removing unused headers).

If you include extra header fields, we recommend that you follow Internet standards, and in particular, if you invent your own header fields, we recommend that you prefix the names with 'X-'.

Mixed case and punctuation in ZMAIL input from an ALC terminal

Internet etiquette requires that e-mail messages are entirely in lower case, except where upper case (capital letters) are required by the rules of your language. For example, in English a person's name usually is written with an initial capital, followed by the rest in lower case, like this: 'Tom'.

E-mail messages also require some punctuation marks, both in the text of your message, and in header fields (for example, you need the '@' in e-mail addresses).

Because many ALC terminals do not allow entry of mixed-case text, and many do not allow entry of the punctuation marks you may need, the ZMAIL command provides special facilities to help you.

This section describes those special facilities.

Sentences - lower case with an initial capital

The ZMAIL command converts the whole of your message, including any field text in header fields, to lower case.

It then converts the first character of each "sentence", and the first character of each header field name, to a capital. (It also adds a colon character to the end of each header field name.)

A new sentence, in this context, means:

- The field text of a /SUBJECT header field.
- The first line of your message body.
- Any text in your message body which follows a "full stop".

A full stop is a period (.), question mark (?), or exclamation mark (!) which is followed immediately by a space (blank) or new-line.

For example:

If you input this...	**...ZMAIL sends**
/TO SOMEONE /AT SOMEPLACE. COM	To: someone @ someplace. com
/SUBJECT YOUR ENQUIRY	Subject: Your enquiry
HI.	Hi.
FIRST SENTENCE. SECOND SENTENCE.	First sentence. Second sentence.
MY DOMAIN IS UK.IBM.COM.	My domain is uk.ibm.com.
BEST REGARDS. TOM ANYBODY	Best regards. Tom Anybody

Note that:

- ZMAIL translates all the field text in your /TO field to lower case. The only field text it interprets as a sentence is in your /SUBJECT field.
- ZMAIL replaces '/AT' with '@'. (We say more about this in “Punctuation” on page 233.)
- ZMAIL does not interpret 'ibm' or 'com' as new sentences in 'uk.ibm.com' because there is no space or new line immediately following the periods.
- ZMAIL does not use a capital 'h' for 'anybody' because it does not know that 'anybody' is a person's name. (We explain how you fix this in “Other capitals” on page 233.)

To get the best result when you enter e-mail messages, we recommend you follow these two rules:

- **Always end a sentence with a full stop.**

  Notice that in the example above, we end the conventional greeting 'Hi' with a full stop. It is more usual to use a comma for this. But a full stop ensures that the next sentence (which really *is* a new sentence) correctly starts with a capital. And we do not have a comma on our ALC keyboard.

- **Never use a full stop in the middle of a sentence.**

  In English, abbreviations are often written with a following period, for example 'Mr.', 'Co.', 'etc.', and so on. ZMAIL interprets these periods as full stops.

  We recommend that you either avoid this type of abbreviation, or omit the period.

Other capitals

"Sentences - lower case with an initial capital" on page 232 explains how you can enter your e-mail messages so that ZMAIL automatically uses a capital letter at the start of each sentence.

But sometimes you need to capitalize a character that is not at the start of a sentence.

To capitalize a character in your e-mail message, you insert an asterisk (*) immediately before that character. If you want an actual asterisk in your e-mail, enter two asterisks (Internet etiquette suggests using asterisks to emphasize a word, like *this*).

For example:

If you input this...	...ZMAIL sends
*MRS DE*HAMEL	Mrs deHamel
*I*B*M	IBM
THIS IS AN **IMPORTANT** WORD	this is an *important* word
HE IS THE ***KING**	he is the *King*

Punctuation

All ALC terminals (or at least, all the ones we know about) allow you to enter the following punctuation marks:

.
Period

-
Minus or hyphen

*****
Asterisk

/
Slash

$
Currency (shows as £ on some equipment)

Note: Some ALC terminals can enter other punctuation marks such as comma, left and right parenthesis, plus, lozenge (&loz.), end-item (&bx1212.), Cross-of-Lorraine, window, and so on. We advise you *not* to use these characters in the ZMAIL command. We recommend that you restrict the punctuation marks you use to those marked with "yes" in the "Safe" column of Table 21 on page 235.

You can use periods and hyphens anywhere in your e-mail messages in the same way you use them in normal typing. We already explained how you use asterisk in your e-mail messages (see "Other capitals" on page 233) and we are about to explain how you use slash.

Tip:

We advise you to avoid using punctuation marks which you can not enter from your terminal and to avoid using slashes.

When you must use one of these characters -- and you can not avoid using the at-sign in e-mail addresses -- then you must use the method we describe next.

But you can mostly avoid using difficult characters by wording your e-mail carefully. If you do this then probably you will enter your messages more quickly and with less mistakes.

Notice that we do not use any difficult characters in this tip. We do not use semicolons or parentheses or quotes or apostrophes. We do not even use a comma.

Probably you have noticed that ZMAIL allows you to include '/AT' to represent '@' in e-mail addresses. More generally, ZMAIL allows you to enter any US ASCII punctuation mark with a token of the form /*name*. For example, you can use '/COMMA' to represent a comma.

Table 21 on page 235 lists the US ASCII punctuation marks, together with the special tokens which you can use to enter them on an ALC terminal.

Tip:

You might want to refer to Table 21 on page 235 when you enter e-mail from an ALC terminal.

If you are reading this book from CD-ROM, you can print a copy of the table on your PC printer. If you have a printed copy of this book then you can photocopy the page.

If your terminal can enter the punctuation mark you want, then you do not need to use the tokens shown in Table 21 on page 235, except that you **must** use tokens for the asterisk (*) and slash (/) characters.

The following paragraphs explain how you use the tokens shown in Table 21 on page 235 in your e-mail messages.

Long forms and short forms

The column headed "Long form" shows the full-length token for representing each punctuation mark. To reduce the amount of typing you need, you can shorten some tokens. For example, you can shorten 'COMMA' to 'COMM', 'COM', or 'CO'

We list the shortest form that ZMAIL understands in the column "Short form". Notice that this column omits the trailing slash character - we explain this next.

Spacing

The slashes at the start and end of each token shown in the "Long form" column of Table 21 on page 235 allow ZMAIL to recognize the token and separate it from other text.

If you want a space (blank) or new line after your punctuation mark then you can (if you want) omit the trailing slash.

If you do *not* want a space or new line after your punctuation mark then you must include the trailing slash.

For example:

If you input this...	**...ZMAIL sends**
SOMEONE /AT/ SOMEPLACE.COM	someone @ someplace.com
SOMEONE /AT SOMEPLACE.COM	someone @ someplace.com
SOMEONE/AT SOMEPLACE.COM	someone@ someplace.com
SOMEONE/AT/SOMEPLACE.COM	someone@someplace.com
/DQ/SOME ONE/DQ//AT/SOMEPLACE.COM	"some one"@someplace.com

Alternative forms

The "Alternative" column of Table 21 on page 235 shows special tokens for some common punctuation marks. Do **not** add a trailing slash to these special tokens. For example:

If you input this...	**...ZMAIL sends**
IT*-S A LOVELY DAY*. BE HAPPY.	it's a lovely day, be happy.

Table 21. US ASCII punctuation marks for e-mail

	Safe	**Name**	**ALC representation**			**Full Stop**
			Long form	**Short form**	**Alternative**	
.	yes	Dot	/DOT/	/DO	.	yes
?	yes	Query	/QUERY/	/QU		yes
,	yes	Comma	/COMMA/	/CO	*.	
:	yes	Full colon	/FCOLON/	/FC		
'	yes	Single quote, Apostrophe	/SQUOTE/	/SQ	*-	
=	yes	Equal	/EQUAL/	/EQ		
-	yes	Hyphen, Minus	/HYPHEN/	/HY	-	
+	yes	Plus	/PLUS/	/PL		
/	yes	Slash	/SLASH/	/SL	*/	
(	yes	Left parenthesis	/LPARENTHESIS/	/LP		
)	yes	Right parenthesis	/RPARENTHESIS/	/RP		
!		Exclamation	/EXCLAMATION/	/EX		yes
;		Semicolon	/SCOLON/	/SC		
"		Double quote	/DQUOTE/	/DQ		
&		And	/AND/	/AN		
@		At	/AT/	/AT		
\		Back slash	/BSLASH/	/BS		
$		Currency	/CURRENCY/	/CU	$	
`		Grave	/GRAVE/	/GR		
^		Hat, Caret	/HAT/	/HA		
%		Percent	/PERCENT/	/PE		
#		Pound, Hash	/POUND/	/PO		
*		Star, Asterisk	/STAR/	/ST	**	
~		Tilde	/TILDE/	/TI		
_		Underscore	/USCORE/	/US		
|		Vertical bar	/VBAR/	/VB		
{		Left brace	/LBRACE/	/LBRACE		
}		Right brace	/RBRACE/	/RBRACE		
[		Left bracket	/LBRACKET/	/LBRACK		
]		Right bracket	/RBRACKET/	/RBRACK		
<		Left angle, Less than	/LANGLE/	/LA		
>		Right angle, Greater than	/RANGLE/	/RA		

Table 21. US ASCII punctuation marks for e-mail (continued)						
	Safe	**Name**	**ALC representation**			**Full Stop**
			Long form	**Short form**	**Alternative**	
Multipurpose Internet mail Extensions (MIME) Part Five: Conformance Criteria and Examples, RFC 2049, lists punctuation marks that are interpreted consistently on all e-mail systems. In this table, these punctuation marks are indicated by "yes" in the "Safe" column.						

Using ZMAIL to send an e-mail message from an ALC terminal

The following example shows a variety of e-mail address formats, all of which are allowed in the ALCS ZMAIL command:

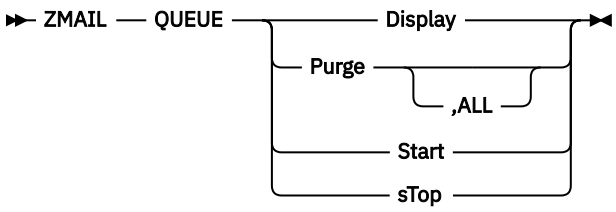
```
zmail
/TO GFISHER  /AT  UK.IBM.COM
/CC ARSMITH  /AT  UK.IBM.COM
/CC /DQ/A. N. OTHER/DQ/ /AT SOMEPLACE.COM
/BCC PERSON /AT ANOTHER.PLACE.ORG
/FROM TJONES /AT UK.IBM.COM
/SUBJECT  SUCCESSFUL SALE
HI.
I WANTED TO TELL YOU ABOUT MY SUCCESSFUL HARDWARE SALE.
WHEN CAN I PHONE YOU/QU
BRGDS. TOM
```

ZMAIL -- Control the outbound e-mail queue handler

Use the ZMAIL command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal*** -- to control the outbound e-mail queue handler.

The format of the command is:

ZMAIL syntax



Where:

Display
Display the status of the outbound e-mail queue (**started** or **stopped**) and the number of messages currently on queue.

Purge
Discard the first message from the outbound e-mail queue.

Purge,ALL
Discard all messages from the outbound e-mail queue.

Start
Start sending messages from the outbound e-mail queue. Use this command to resume the transmission of e-mail messages after a previous ZMAIL QUEUE,STOP command.

sTop
Stop sending messages from the outbound e-mail queue. Use this command to suspend the transmission of e-mail messages, for example when you change the IP address of the local message transfer agent (MTA) for outbound e-mail.

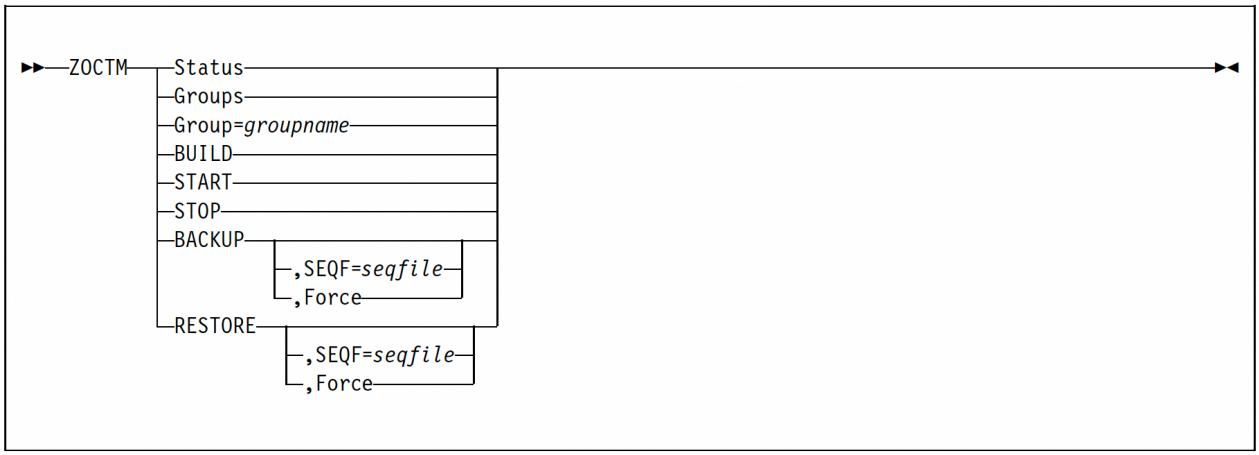
ZOCTM -- Control online communication table maintenance

Use the ZOCTM command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal only*** -- to obtain information about the status of Online Communication Table Maintenance (OCTM) and about the communication resources and communications groups on the OCTM database.

Use the ZOCTM command -- ***from a Prime CRAS authorized terminal only*** -- to build the initial OCTM database, to control access to the OCTM database, and to perform backup and restore functions on the OCTM database.

Generally, ZOCTM commands are only allowed if the ALCS system is using OCTM (OCTM=YES on the communication generation COMGEN macro). ZOCTM RESTORE is an exception. In an emergency, you can use ZOCTM RESTORE to restore the OCTM database, even if ALCS is not using OCTM.

The format of the command is:



Where:

Status
Obtain the current status of OCTM and display information about the contents of the OCTM database.

Groups
Obtain information about each OCTM communications group currently allocated on the OCTM database.

Group=*groupname*
Obtain information about all the communication resources that belong to the OCTM communications group *groupname*.

BUILD
Perform the initial build of the OCTM database from the communication resources that are defined in the communication generation load modules.

START
Enable access to the OCTM database for the purpose of adding, replacing and deleting communication resources.

STOP
Inhibit access to the OCTM database for the purpose of adding, replacing and deleting communication resources.

BACKUP
Copy the contents of every record on the OCTM database to the CMB sequential file (CMB is the symbolic name for the sequential file).

BACKUP , SEQF=*seqfile*
Copy the contents of every record on the OCTM database to the sequential file whose symbolic name is *seqfile*.

BACKUP,Force
Reset the status of the OCTM database backup function when it has failed to complete successfully.

RESTORE
Restore the complete OCTM database from the CMR sequential file (CMR is the symbolic name for the sequential file). The sequential file should have been either directly or indirectly created by the OCTM database backup function.

For ZOCTM RESTORE, ALCS must be in IDLE state.

After ZOCTM RESTORE completes, you must restart the ALCS system in order to rebuild the online communication table from the restored OCTM database.

RESTORE , SEQF=*seqfile*
Restore the complete OCTM database from the sequential file whose symbolic name is *seqfile*. The sequential file should have been either directly or indirectly created by the OCTM database backup function.

RESTORE,Force
Reset the status of the OCTM database restore function when it has failed to complete successfully.

Restriction

When you issue the ZOCTM command with the RESTORE parameter, and the sequential file contains an OCTM database with a different ALCS system ID, ALCS requires you to confirm the command (see “Confirmation of commands” on page 60).

Normal responses

Normal response to ZOCTM Status:

```
DXC5118I CME i  hh.mm.ss OCTM STATUS current_status
Total of BASE physical records     - aaa
Total of CHANGE physical records   - bbb
Total of Group Name Table entries  - ccc
Total of GROUPS                    - ddd
```

Where:

current_status
The current status of the OCTM database. It can be one of the following:

ALLOCATE COMPLETE
BUILD COMPLETE
ACTIVE - ACCESS ALLOWED
ACTIVE - ACCESS NOT ALLOWED

aaa
The count of base physical records that reside in the OCTM database. Each physical record can contain a maximum of 15 communication resource definitions.

bbb
The count of change physical records that reside in the OCTM database. Each physical record can contain a maximum of 7 communication resource definitions.

ccc
The count of entries that are in use in the communications group table in the OCTM database. There is an entry for each communications group that is currently allocated, plus an entry for each non-group communications change request.

ddd
The count of OCTM communications groups that are currently allocated.

Normal response to ZOCTM Groups:

```
DXC5120I CME i  hh.mm.ss OCTM  GROUPS overview
Groupname    Allocate           Change             Total Status Processing
groupname    yyyy.ddd hh.mm.ss  yyyy.ddd hh.mm.ss  aaaa  bb     cc
```

Where:

groupname
Name of a currently allocated OCTM communications group.

Allocate
Date and time when this OCTM communications group was allocated.

Change
Date and time when the last change occurred on the OCTM database for the communications group (and any resources belonging to it).

yyyy.ddd
Year, and day number in the year.

hh.mm.ss
Time in hours, minutes, and seconds.

aaaa
The count of communication resources that belong to this OCTM communications group.

bb
The current status of the OCTM communications group.

EMPTY
No change requests exist for the communications group (anymore).

NOT LOADED
Change requests have been received for the communications group, but are not loaded.

LOADED
Change requests for the communications group have been loaded.

LOADED/ERR
Not all the change requests for the communications group have been loaded.

BACKED OUT
Change requests for the communications group have been backed out.

CONFIRMED
Change requests for the communications group have been confirmed.

cc
The current processing status of the OCTM communications group. The status should normally be NONE (no process in progress), but if it remains a different status this indicates that the COMTC that is currently being processed for the communications group has failed to complete successfully. In this situation, repeat the last COMTC action again.

NONE
No COMTC actions are currently processed for the communications group.

CHANGE REQ
A change request, submitted via a COMTC macro is currently being processed for the communications group.

LOAD
A COMTC LOAD macro is currently being processed for the communications group.

BACKOUT
A COMTC BACKUP macro is currently being processed for the communications group.

CONFIRM
A COMTC CONFIRM macro is currently being processed for the communications group.

COMMIT
A COMTC COMMIT macro is currently being processed for the communications group.

Normal response to ZOCTM Group=*groupname*:

```
DXC5127I CME i  hh.mm.ss OCTM  GROUP overview
Groupname    Allocate           Change             Total Status Processing
groupname    yyyy.ddd hh.mm.ss  yyyy.ddd hh.mm.ss  aaaa  bb     cc
Resource Status Resource Status Resource Status Resource Status Resource Status
crn      status crn      status crn      status crn      status crn      status
```

Where:

groupname
Name of a currently allocated OCTM communications group.

Allocate
Date and time when this OCTM communications group was allocated.

Change
Date and time when the last change occurred on the OCTM database for the communications group (and any resources belonging to it).

yyyy.ddd
Year, and day number in the year.

hh.mm.ss
Time in hours, minutes, and seconds.

aaaa
The count of communication resources that belong to this OCTM communications group.

bb
The current status of the OCTM communications group.

EMPTY
No change requests exist for the communications group (anymore).

NOT LOADED
Change requests have been received for the communications group, but are not loaded.

LOADED
Change requests for the communications group have been loaded.

LOADED/ERR
Not all the change requests for the communications group have been loaded.

BACKED OUT
Change requests for the communications group have been backed out.

CONFIRMED
Change requests for the communications group have been confirmed.

cc
The current processing status of the OCTM communications group. The status should normally be NONE (no process in progress), but if it remains a different status this indicates that the COMTC that is currently being processed for the communications group has failed to complete successfully. In this situation, repeat the last COMTC action again.

NONE
No COMTC actions are currently processed for the communications group.

CHANGE REQ
A change request, submitted via a COMTC macro is currently being processed for the communications group.

LOAD
A COMTC LOAD macro is currently being processed for the communications group.

BACKOUT
A COMTC BACKUP macro is currently being processed for the communications group.

CONFIRM
A COMTC CONFIRM macro is currently being processed for the communications group.

COMMIT
A COMTC COMMIT macro is currently being processed for the communications group.

crn
The name of a communication resource that belongs to this communications group.

status
The type of change request that resides on the OCTM database for this communication resource. It can be one of the following, ADD, DELETE, or CHANGE.

Normal responses to ZOCTM BUILD:

```
DXC5101I CME i  hh.mm.ss OCTM  BUILD started
```

This message is output to Prime and RO CRAS when ALCS starts building the OCTM database.

```
DXC5102I CME i  hh.mm.ss OCTM  BUILD completed
```

This message is output to Prime and RO CRAS when ALCS completes the building of the OCTM database.

Normal response to ZOCTM START:

```
DXC5104I CME i  hh.mm.ss OCTM  Access allowed
```

The OCTM database can now be accessed for the purpose of processing COMTC macros (for processing change requests etc.).

Normal response to ZOCTM STOP:

```
DXC5106I CME i  hh.mm.ss OCTM  Access not allowed
```

Access to the OCTM database is now inhibited for the purpose of processing COMTC macros (for processing change requests etc.).

Normal responses to ZOCTM BACKUP:

```
DXC5108I CME i  hh.mm.ss OCTM  BACKUP started
```

This message is output to Prime and RO CRAS when ALCS starts a backup of the OCTM database.

```
DXC5110I CME i  hh.mm.ss OCTM  BACKUP finished
```

This message is output to Prime and RO CRAS when ALCS completes the backup of the OCTM database.

Normal responses to ZOCTM RESTORE:

```
DXC5114I CME i  hh.mm.ss OCTM  RESTORE started
```

This message is output to Prime and RO CRAS when ALCS starts a restore of the OCTM database.

```
DXC5115I CME i  hh.mm.ss OCTM  RESTORE finished
```

This message is output to Prime and RO CRAS when ALCS completes the restore of the OCTM database.

Normal response to ZOCTM BACKUP,Force and ZOCTM RESTORE,Force:

```
DXC5112I CME i hh.mm.ss OCTM  FORCE ACCEPTED
```

The status of the OCTM database backup or restore function has been reset. If a backup or restore is now required, enter the appropriate command to start it.

ZPCTL -- Load/unload programs and installation-wide monitor exits

Use the ZPCTL command to load a load set or to load and unload program load modules and installation-wide monitor exit load modules. If your ALCS system is using the program configuration data set (CDS1):

- Additional functions are performed when program and installation-wide monitor exit load modules are load and unload.
- You can use the ZPCTL command to load and unload a program configuration table. This table contains a list of the program and installation-wide monitor exit load modules that must be reloaded during a restart of the ALCS system.

You can therefore use the ZPCTL command to:

- Load and unload programs or installation-wide monitor exits for system-wide use.

  Load a program load module or an installation-wide monitor exit load module for system-wide use. Unload (backout) a load module and delete it from memory. Promote (commit) a load module and change its attribute (status) to permanent so that it can not be unloaded.

  If your system is using the program configuration data set (CDS1), the following additional functions are provided. The `Load` command will insert the name of the load module in a load list on CDS1. The `CONfirm` command will mark the load module as confirmed in a load list on CDS1 (this ensures that the load module will be reloaded during the next ALCS restart). The `Unload` (`Backout`) command will not only delete the load module from memory but also mark it as backed out (unloaded) in a load list on CDS1. This ensures that the load module will not be reloaded during the next ALCS restart. The `Promote` (`COMmit`) command will not only change the attribute (status) of the load module to permanent, but also mark it as committed in a load list on CDS1. This ensures that it can not be deleted from the CDS1 load list.
- Load and unload programs for use by a single terminal.
- Display load module status information.

  Display status information about program load modules and installation-wide monitor exit load modules that have been loaded for system-wide use and for single terminal use.
- Load program configuration table.

  If your system is using CDS1, you can load a new program configuration table onto CDS1. The table includes a load list that contains the names of system-wide program and installation-wide monitor exit load modules. You can confirm a new program configuration table that has been recently loaded (this requests ALCS to use it on the next ALCS restart). You can backout a new program configuration table and mark it as backed out on CDS1. You can commit a new program configuration table that has been verified by a restart of the ALCS system. A status report of the program configuration tables on CDS1 can be obtained, together with the status of the load modules referenced by the program configuration tables.

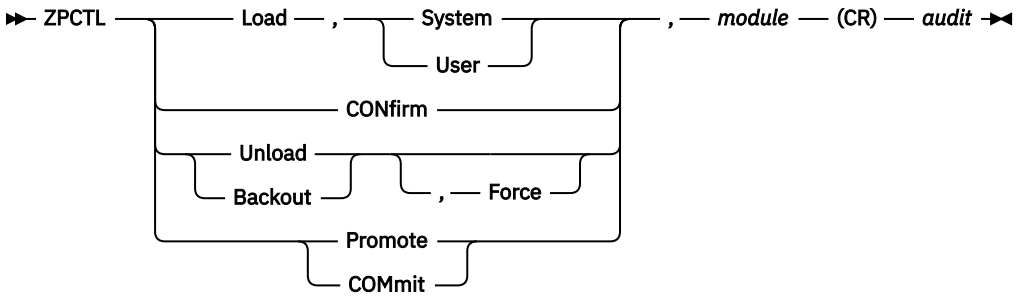
For examples of the LOAD parameter see Normal response to commands that load/unload programs or installation-wide exits, Normal response to commands that load/unload programs for use by a single terminal, and Normal response to ZPCTL LOAD,SET.

ZPCTL -- Load/unload programs or installation-wide exits for system-wide use

Use the ZPCTL command -- ***from a Prime CRAS authorized terminal only*** -- to load a program load module or an installation-wide monitor exit load module for system-wide use. Unload (backout) a load module and delete it from memory or promote (commit) a load module and change its attribute (status) to permanent so that it can not be unloaded. If your ALCS system is using the program configuration data set (CDS1), the following additional functions are provided. The `Load` command will insert the name of the load module in a program configuration table load list on CDS1. The `CONfirm` command will mark the load module as confirmed on CDS1 and ensure that it is reloaded during the next ALCS restart. The `Backout` (`Unload`) command will mark the load module as backed out (unloaded) on CDS1. The `COMmit` (`Promote`) command will mark the load module as committed (promoted) on CDS1.

The format of the command is:

ZPCTL syntax



Where:

Load,System
Load an ECB-controlled program load module for system-wide use. This command is not preserved across an ALCS restart. If ALCS restarts, it will restart without the program load module *module*. If CDS1 is in use on the ALCS system, the name of the load module is inserted in the CDS1 load list.

Load,User
Load an installation-wide monitor exit load module for system-wide use. This command is not preserved across an ALCS restart. If ALCS restarts, it will restart without the installation-wide monitor exit load module *module*. If CDS1 is in use on the ALCS system, the name of the load module is inserted in the CDS1 load list.

CONfirm
Confirm that the specified load module *module*, previously loaded for system-wide use, is safe to preserve across an ALCS restart. Use the confirm function only when CDS1 is in use on the ALCS system. The load module is marked as confirmed in the CDS1 load list, and if a restart of the ALCS system occurs, the load will be reloaded during the ALCS restart.

Unload
Unload (stop using) the specified program or installation-wide monitor exit load module *module*. When there are no more processes using the load module *module*, delete it from memory. If CDS1 is in use on the ALCS system, the load module is marked as backed out (unloaded) in the CDS1 load list. This ensures that the load module will not be reloaded during the next ALCS restart. You can not unload a load module that has been committed (via the `ZPCTL COMmit` or `ZPCTL Promote` commands).

Note: ALCS accepts `ZPCTL Backout` as a synonym for `ZPCTL Unload`.

Unload,Force
Force the unload of the program or installation-wide monitor exit load module *module* from memory, even if there are processes created before the start of unloading this module that still exist. Use

Force only if Unload without Force never completes. If CDS1 is in use on the ALCS system, the load module is marked as backed out (unloaded) in the CDS1 load list.

Note: It is possible for an unload to take several minutes to complete. When it does complete, a message is sent to the RO CRAS.

Backout
Backout (stop using) the specified program or installation-wide monitor exit load module *module*. When there are no more processes using the load module *module*, delete it from memory. If CDS1 is in use on the ALCS system, the load module is marked as backed out (unloaded) in the CDS1 load list. This ensures that the load module will not be reloaded during the next ALCS restart. You can not backout a load module that has been committed (via the ZPCTL COMmit or ZPCTL Promote commands).

Note: ALCS accepts ZPCTL Unload as a synonym for ZPCTL Backout.

Backout,Force
Force the backout of the program or installation-wide monitor exit load module *module* from memory, even if there are processes created before the start of unloading this module that still exist. Use Force only if Backout without Force never completes. If CDS1 is in use on the ALCS system, the load module is marked as backed out (unloaded) in the CDS1 load list.

Note: It is possible for an unload to take several minutes to complete. When it does complete, a message is sent to the RO Cras.

Promote
Change the attribute (status) of the program or installation-wide monitor exit load module *module* to permanent. When a load module is promoted (committed), the ZPCTL Backout and ZPCTL Unload commands can not be used to unload (backout) the load module. If CDS1 is in use on the ALCS system, the load module is marked as committed in the CDS1 load list.

Note: ALCS accepts ZPCTL COMmit as a synonym for ZPCTL Promote.

COMmit
Change the attribute (status) of the program or installation-wide monitor exit load module *module* to permanent. When a load module is committed (promoted), the ZPCTL Backout and ZPCTL Unload commands can not be used to unload (backout) the load module. If CDS1 is in use on the ALCS system, the load module is marked as committed in the CDS1 load list.

Note: ALCS accepts ZPCTL Promote as a synonym for ZPCTL COMmit.

module
The name of a program or installation-wide monitor exit load module (up to 8 characters); this must be unique.

audit
Audit-trail information. Up to 80 characters of text.

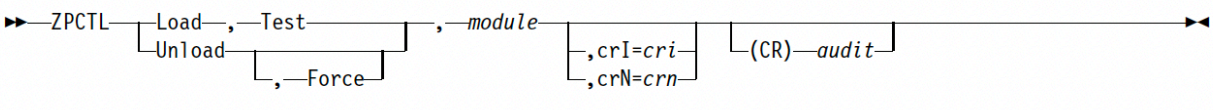
For examples of the LOAD parameter see Normal response to commands that load/unload programs or installation-wide exits.

ZPCTL -- Load/unload programs for use by a single terminal

Use the ZPCTL command -- ***from any CRAS authorized terminal*** -- to load and unload an ECB-controlled program load module for use by a single terminal.

The format of the command is:

ZPCTL syntax



Where:

Load,Test
Load the program load module *module* for use by a single terminal. This command is not preserved across an ALCS restart. If ALCS restarts, it will restart without the program load module.

Unload
Unload (stop using) the program load module *module*. When there are no more processes using the program load module, it is deleted from memory.

Unload,Force
Force the unload of the program load module *module* from memory, even if there are entries created before the start of unloading this module that still exist. Use `Force` only if `Unload` without `Force` never completes.

Note: It is possible for an unload to take several minutes to complete. When it does complete, a message is sent to the RO CRAS.

module
The program load module name, 1-8 characters.

`crI=`*cri*`|crN=`*crn*
Where:

cri
CRI of the test terminal that owns this program load module.

crn
CRN of the test terminal that owns this program load module.

If omitted, this defaults to the CRI of the input terminal.

audit
Audit-trail information. Up to 80 characters of text. The audit-trail information is optional.

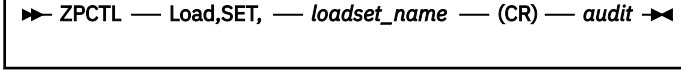
For examples of the LOAD parameter see Normal response to commands that load/unload programs for use by a single terminal.

ZPCTL -- Load a load set

Use the ZPCTL command -- ***from a Prime Cras authorized terminal only*** -- to load a load set. The load set may contain up to a 1000 system-wide program load modules.

The format of the command is:

ZPCTL syntax



Where:

LOAD,SET
Load a load set. This command is not preserved across an ALCS restart. However the load modules of the load set can be confirmed or committed when CDS1 is in use on the ALCS system.

loadset_name
The name of the load set.

See "Updating and generating a load set" in *ALCS Installation and Customization* for information about how to update and to generate a load set.

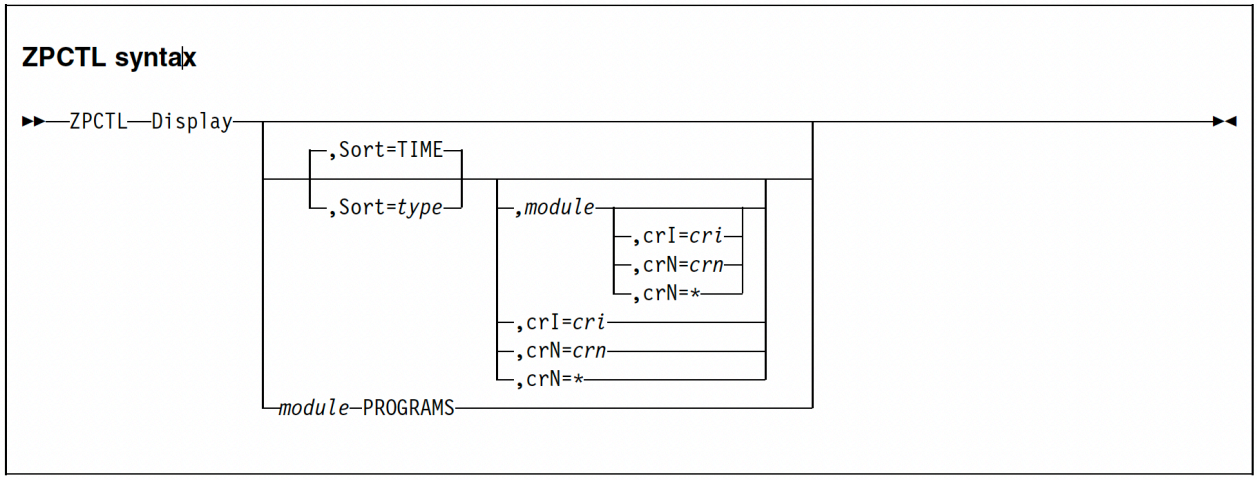
audit
Audit-trail information. The information can be up to 80 characters of text.

For examples of the LOAD parameter see Normal response to ZPCTL LOAD,SET.

ZPCTL -- Display load module status information

Use the ZPCTL command -- ***from any terminal*** -- to display status information about program load modules and installation-wide monitor exit load modules that have been loaded for system-wide use or single terminal use.

The format of the command is:



Where:

Display
Display program load module and installation-wide monitor exit load module information (status of module, module owner, and so on). Without further parameters, `Display` displays a line of information for every module that is currently loaded.

Display,Sort=*type*
Where *type* is one of the following:

TIME
Sort output on date and time of load (this is the default).

NAME
Sort output on module name.

TYPE
Sort output on module type.

STATUS
Sort output on module status.

DATE
Sort output on date and time of load.

NONE
Do not sort output.

module
The load module name, 1-8 characters.

crI=*cri*|crN=*crn*|(crN=*
Where:

cri
CRI of the test terminal that owns this load module. Omit *module* to display all the load modules that are owned by this test terminal.

crn
CRN of the test terminal that owns this load module. Omit *module* to display all the load modules that are owned by this test terminal.

*****
Display all the test terminals that own this load module. Omit *module* to display all the load modules that are owned by all test terminals.

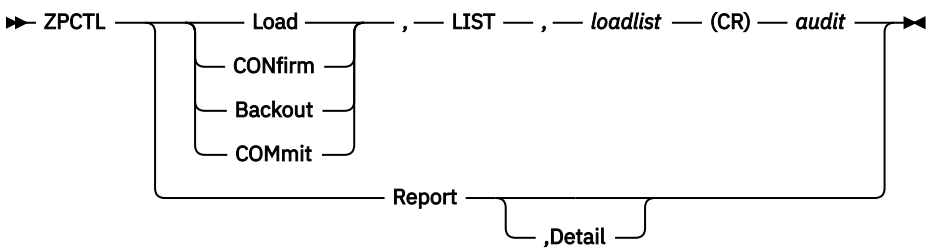
PROGRAMS
List all programs contained in a load module. This option ignores modules containing installation-wide user exits and test load modules loaded from other terminals.

See Normal response to ZPCTL DISPLAY for an example of the ZPCTL DISPLAY command.

ZPCTL -- Load program configuration table

Use the ZPCTL command -- ***from a Prime CRAS authorized terminal only*** -- to load a new program configuration table onto the program configuration data set (CDS1). The table contains a list of system-wide program and installation-wide monitor exit load modules that must be loaded on the next restart of the ALCS system. After a new program configuration table has been loaded, you can request ALCS to use it on the next system restart by confirming the table. A backout of a new program configuration table can be requested after it has been loaded or after it has been confirmed. After the next restart of the ALCS system, if the table has been successfully verified by the ALCS restart, it can be committed. A status report of the program configuration tables on CDS1 can be obtained, together with the status of the load modules referenced by the program configuration tables.

ZPCTL syntax



Where:

Load,LIST
Load the program configuration table *loadlist*. ALCS will not use this table until it has been confirmed and the system has been restarted. A new program configuration table can not be loaded until the previous table has been committed or backed out.

CONfirm,LIST
Confirm the program configuration table *loadlist*. This table, which must have been previously loaded, will be used during the next restart of the ALCS system.

Backout,LIST
Backout the program configuration table *loadlist*. This table, which must have been previously loaded and may have been previously confirmed, will not be used during the next restart of the ALCS system. The previous table on CDS1 will now be used during the next restart of the ALCS system. Alternatively, a replacement program configuration table could now be loaded for usage during the next ALCS restart.

COMmit,LIST
Commit the program configuration table *loadlist*. The table must have been previously confirmed. A table can not be committed until after it has been verified by a restart of the ALCS system. When a table has been committed, it can not be backed out.

loadlist
The name of the program configuration table load module (up to 8 characters); this must be unique.

audit
Audit-trail information. Up to 80 characters of text.

Report
Display a status report of the program configuration load list(s) plus the loaded, confirmed, committed program configuration load modules on CDS1.

Report,Detail
Display a status report of the program configuration load list(s) plus all program configuration load modules on CDS1. This includes backed out (unloaded) and never loaded modules.

See Normal response to ZPCTL module PROGRAMS for an example of the PROGRAM parameter and Normal response to ZPCTL REPORT for an example of the REPORT parameter.

Normal responses

Normal response to commands that load/unload programs or installation-wide exits for system-wide use.

```
DXC8340I CMD i hh.mm.ss PCTL
Module MODN module loaded by CRN--crn
```

The module is loaded and available to new entries.

```
DXC8344I CMD i hh.mm.ss PCTL
Module MODN module confirmed by CRN--crn
```

The module is confirmed and will be reloaded during the next ALCS system restart

```
DXC8341I CMD i hh.mm.ss PCTL
Module MODN module promoted by CRN--crn
```

The module is promoted (committed) and can not be backed out.

```
DXC8336I CMD i hh.mm.ss PCTL
Module MODN module unload started by CRN--crn
```

The module is not available to new entries, but remains available to existing entries.

```
DXC8337I CMD i hh.mm.ss PCTL
Module MODN module unload force started by CRN--crn
```

The module is not available to new or existing entries.

```
DXC8342I CMD i hh.mm.ss PCTL
Module MODN module unloaded for CRN--crn
```

Response sent to the RO CRAS when ZPCTL UNLOAD,module has successfully completed.

Normal response to commands that load/unload programs for use by a single terminal

```
DXC8347I CMD i hh.mm.ss PCTL
Test module MODN module loaded by CRN1--crn1 for CRN2--crn2
```

The test module is loaded and available to new entries.

```
DXC8332I CMD i hh.mm.ss PCTL
Test module MODN module unload started by CRN1--crn1 for CRN2--crn2
```

The test module is not available to new entries, but remains available to existing entries.

```
DXC8346I CMD i hh.mm.ss PCTL
Test module MODN module unload force started by CRN1--crn1 for CRN2--crn2
```

The test module is not available to new or existing entries.

```
DXC8342I CMD i hh.mm.ss PCTL
Module MODN module unloaded for CRN--crn
```

Response sent to the RO CRAS when ZPCTL UNLOAD,module has successfully completed.

Normal response to ZPCTL LOAD,SET

```
DXC5344I CME i  hh.mm.ss  PCTL
All modules of load set successfully loaded
```

All the load modules of this load set are loaded.

Normal response to ZPCTL DISPLAY

```
DXC8345I CMD i hh.mm.ss PCTL
Module    Type        CRN/Status   Load Date/Time        Unload Date/Time
module    type        xxxxx        yyyy.ddd hh.mm.ss.lll yyyy.ddd  hh.mm.ss.lll
```

Where:

module
Load module name.

type
Load module type: Monitor, ECB, or HLLload, where:

Monitor
A module loaded on behalf of the ALCS Monitor

ECB
An E-Type program

HLLload
A module loaded using the HLL Load interface

xxxxx
CRN of the terminal that owns a test load module, or the status of a system-wide load module. For system-wide load modules that have been loaded but not promoted (committed), the status field is left blank. When the ZPCTL Promote or ZPCTL COMmit command has been used to promote (commit) a load module, the status field contains PERMANENT.

yyyy.ddd
Year, and day number in the year.

hh.mm.ss.lll
Time in hours, minutes, seconds, and milliseconds.

Load Date/Time
Date and time when the load started.

Unload Date/Time
Date and time when the unload started (appears only if an unload is started but not complete). Unload does not complete until all entries created before the Unload Date/Time complete processing.

Normal response to ZPCTL module PROGRAMS

```
DXC5320I CME i  hh.mm.ss PCTL
Program   Version   Module            Program   Version   Module
prog      vv        module            prog      vv        module
```

Where:

prog
Program name

vv
Version number

module
Name of the application load module that contains the program.

Normal response to commands that load/confirm/backout/commit program configuration tables

```
DXC8925I CMD i hh.mm.ss PCTL
List MODN module loaded
```

The program configuration table has been loaded.

```
DXC8926I CMD i hh.mm.ss PCTL
List MODN module confirmed
```

The program configuration table has been confirmed. It will be used during the next ALCS system restart.

```
DXC8928I CMD i hh.mm.ss PCTL
List MODN module backed out
```

The program configuration table has been backed out.

```
DXC8927I CMD i hh.mm.ss PCTL
List MODN module committed
```

The program configuration table has been committed.

Normal response to ZPCTL REPORT

```
DXC8931I CMD i hh.mm.ss PCTL  CDS -- Using rrrrrrr slot s
ttttttt slot u
Module   Status    Load Date/Time    Update Date/Time
zzzzzzz  module   xxxxxx    yyyy.ddd hh.mm.ss yyyy.ddd hh.mm.ss list  audit trail

ttttttt slot u
Module   Status    Load Date/Time    Update Date/Time
zzzzzzz  module   xxxxxx    yyyy.ddd hh.mm.ss yyyy.ddd hh.mm.ss list  audit trail
```

A status report of the program configuration tables on CDS1, together with the status of the load modules referenced by the program configuration tables, where:

rrrrrrr
The CDS1 slot that was used during the last restart of the ALCS system. This slot is either CURRENT or ALTERNATE.

s
The name of the CDS1 slot that was used during the last restart of the ALCS system. This is either A or B.

ttttttt
The CDS1 slot for which the following status report relates to. This slot is either CURRENT or ALTERNATE.

u
The name of the CDS1 slot for which the following status report relates to. This is either A or B.

zzzzzzz
When this field contains LOAD LIST the rest of the line contains details of a program configuration table load list. When this field contains LOADMOD the rest of the line contains details of a load module. There are two program configuration table load lists included in the status report. Details of the first load list are given in the first line of the report, with the following lines containing details of each load module that is referenced in that load list. This is followed by details of the second load list and its load modules.

module
The name of the load module. It will be a program configuration table load module, or a program load module, or an installation-wide monitor exit load module.

xxxxx
The status of the load module. It will be a program configuration table load module, or a program load module, or an installation-wide monitor exit load module. The status will normally be one of the following; LOADED, CONFIRMED, COMMITTED or BACKED OUT.

yyyy.ddd
Year, and day number in the year.

hh.mm.ss
Time in hours, minutes, and seconds.

Load Date/Time
Date and time when this load module was loaded on CDS1

Update Date/Time
Date and time when the status of this load module was last updated on CDS1.

list
This field contains LIST when the program or installation-wide monitor exit load module was in the load list at the time when the program configuration table was loaded onto CDS1. For load modules that have been loaded online, this field is left blank.

audit trail
The audit trail information that was included in the ZPCTL command. If your display terminal is 80 columns wide, you will need to use the ZSCRL Right command to view the audit trail information.

ZPDAR -- Maintain and display the PDAR Table

Use the ZPDAR command ***from a Prime CRAS authorized terminal only*** to maintain the ALCS Pool dispensing array for restore (PDAR) table.

Use the ZPDAR command ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1-AT16 authorized terminal only*** to display information about the ALCS PDAR table.

The format of the command is:

Where:

DISplay
Display PDAR status and number of reserved records.

CREATE
Initialize PDAR structure table. PDAR pool records are reserved.

PURge
PDAR structure table is discarded. PDAR reserved pool records are released.

DELete
Same as PURGE.

See "Pool dispensing array for restore -- PDAR" on page 20 for further details.

Normal responses

Normal response to ZPDAR CREATE:

```
DXC5400I CME i hh.mm.ss PDAR  Table successfully created
```

Normal response to ZPDAR DELETE:

```
DXC5402I CME i hh.mm.ss PDAR Table successfully cleared
```

Normal response to ZPDAR DISPLAY:

```
DXC5407I CME i hh.mm.ss PDAR  Dispensing is inactive

      Reserved records counts :

      LT L1 ......nnnnnn
      LT L2 ......nnnnnn
      LT L3 ......nnnnnn
      ⋮

OR

   DXC5404I CME i hh.mm.ss PDAR  Dispensing is active

      Reserved records counts :

      LT L1 ......nnnnnn
      LT L2 ......nnnnnn
      LT L3 ......nnnnnn
      ⋮
```

Where :

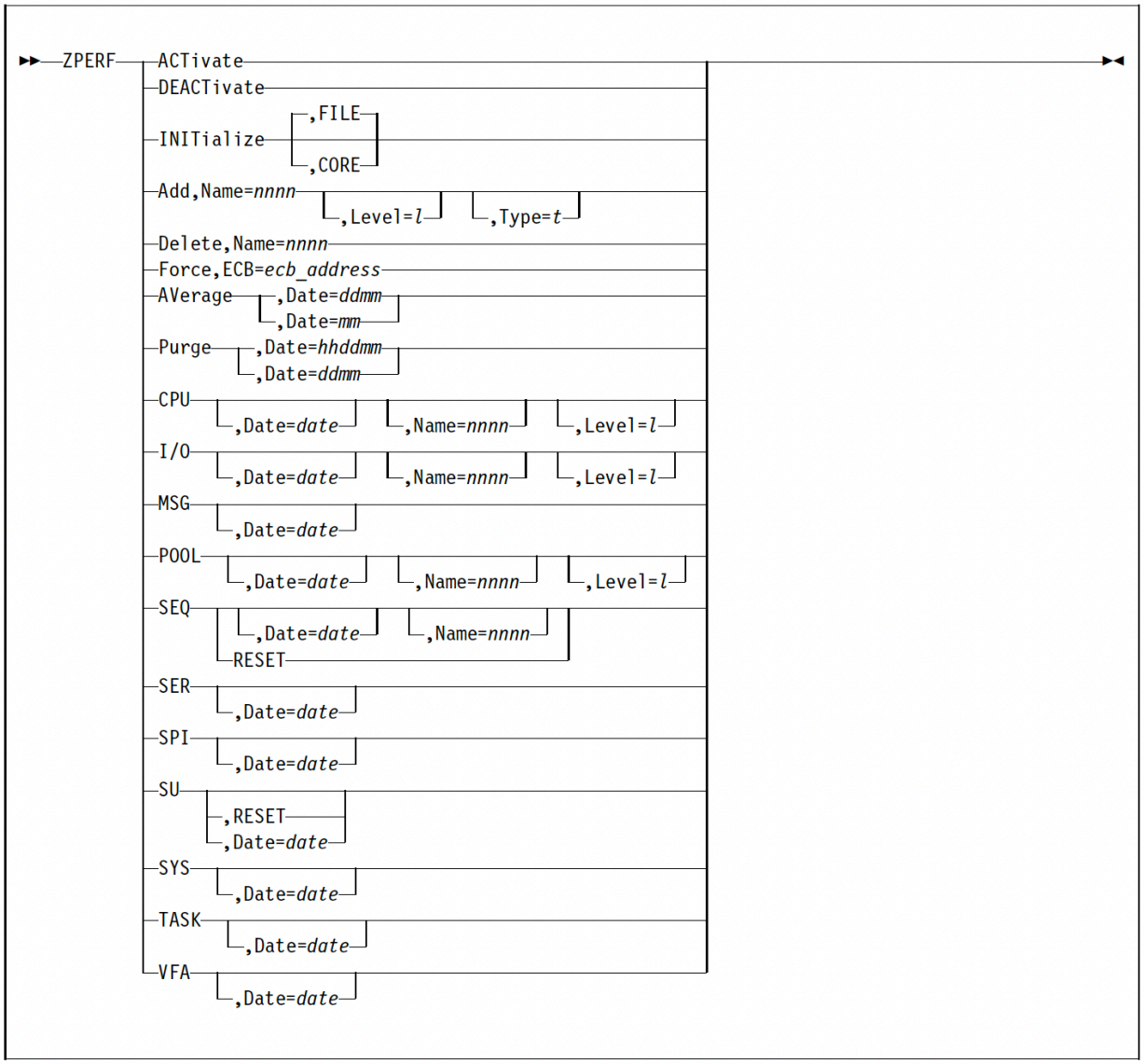
nnnnnn
Number of remaining reserved records.

ZPERF -- Control and display the ALCS performance monitor

Use the ZPERF command -- ***from a Prime CRAS authorized terminal only*** -- to control the ALCS performance monitor.

Use the ZPERF command -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal only*** -- to display ALCS performance monitor information.

The format of the command is:



Where:

ACTivate
Activate performance monitor.

DEACTivate
Deactivate performance monitor.

INITialize
Initialize the performance monitor table.

FILE
Initialize the table on file and in memory.

CORE
Initialize the table in memory only.

Add
Add performance monitor application. This change is preserved over an ALCS restart.

Delete
Delete performance monitor application. This change is preserved over an ALCS restart.

Force
Identify an active entry - and any entries it subsequently creates with CREMC (`cremc`), CREDC (`credc`), CREXC (`crexc`), or CREEC (`creec`) - to the ALCS performance monitor, using the PERF application. You must have previously defined PERF to the performance monitor using the ZPERF command, with level 1, 2, or 3. This command overrides any previous performance monitor application that the entry was using at the same level as PERF.

AVerage
Calculate the average of the performance monitor results for the specified day or month (only possible with the current year results).

Purge
Ignore the performance monitor results for the specified hour or day when calculating averages. (only possible with the current year results).

CPU
Display CPU usage.

I/O
Display I/O usage.

MSG
ZPERF MSG is a synonym of ZSTAT MSG (see "ZSTAT -- Display current system load" on page 297).

POOL
Display long-term pool usage.

SEQ
Display sequential file information.

RESET
Reset sequential file information.

SER
Display serialization information.

SPI
Display SPIs (System Performance Indicators).

SU
Display storage unit usage.

RESET
Reset and display storage unit usage.

SYS
Display ALCS system information.

TASK
Display CPU usage for ALCS tasks.

VFA
ZPERF VFA is a synonym of ZSTAT VFA (see "ZSTAT -- Display current system load" on page 297).

l
Performance monitor application level. Specify 1 (default) or 2 or 3.

nnnn
Performance monitor application name or sequential file name.

hh
Hour.

dd
Day.

mm
Month.

t
Performance monitor entry type. Specify A (default) for application type or S for sequential file type.

yy
Year.

date
Date in format *hhddmmyy*, or *ddmmyy*, or *mmyy*. Display performance monitor information for the specified hour, day, or month.

ecb_address
Address of the ECB to be forced.

Normal responses

Normal response to ZPERF ACTivate:

```
DXC5144I CME i  hh.mm.ss PERF  performance monitor activated
```

Normal response to ZPERF DEACTivate:

```
DXC5145I CME i  hh.mm.ss PERF  performance monitor deactivated
```

Normal response to ZPERF INITialize:

```
DXC5143I CME i  hh.mm.ss PERF  Performance Table initialized
```

Normal response to ZPERF Add:

```
DXC5169I CME i  hh.mm.ss PERF  performance monitor application added
```

Normal response to ZPERF Delete:

```
DXC5170I CME i  hh.mm.ss PERF  performance monitor application deleted
```

Normal response to ZPERF Force:

```
DXC5171I CME i  hh.mm.ss PERF  ZPERF FORCE finished
```

Normal response to ZPERF AVerage:

```
DXC5172I CME i  hh.mm.ss PERF  ZPERF AVERAGE finished
```

Normal response to ZPERF Purge:

```
DXC5173I CME i  hh.mm.ss PERF  ZPERF PURGE finished
```

Normal response to ZPERF CPU:

```
DXC5151I CME i  hh.mm.ss PERF  CPU
Appl   C-Time  P-Time  A-Time  S-Time   A-SPI  E-Life MCR/ECB   ECBs
SYST   c-time  p-time  a-time  s-time   a-spi  e-life macros    ecbs
DFT1   c-time  p-time  a-time  s-time   a-spi  e-life macros    ecbs
appl   c-time  p-time  a-time  s-time   a-spi  e-life macros    ecbs
⋮
```

Where:

appl
Performance monitor application.

c-time
CPU usage for this performance monitor application (percentage in format *nn.nn*). Calculated over the standard processors only. IBM System z® Application Assist Processors (zAAP) and IBM System z9® Integrated Information Processors (zIIP) are not considered in the calculation.

p-time
Total process time for this performance monitor application. This is the total of *a-time* and *s-time* (percentage in format *nn.nn*).

a-time
Application (ECB) process time for this performance monitor application (percentage in format *nn.nn*).

s-time
ALCS monitor system process for this performance monitor application (percentage in format *nn.nn*).

a-spi
SPI (system performance indicator) for this performance monitor application.

e-life
Average ECB life in milliseconds for this performance monitor application.

macros
Number of authorized monitor-request macro calls per ECB.

ecbs
Number of ECBs per second.

Note: *c-time*, *p-time*, *a-time*, and *s-time* are approximate percentages.

Normal response to ZPERF I/O:

```
DXC5155I CME i  hh.mm.ss PERF  I/O
Appl    T-Read    R-Read    V-Read    T-Write    R-Write    V-Write
SYST    t-read    r-read    v-read    t-write    r-write    v-write
DFT1    t-read    r-read    v-read    t-write    r-write    v-write
appl    t-read    r-read    v-read    t-write    r-write    v-write
⋮
```

Where:

appl
Performance monitor application.

t-read
Number of reads per second for this performance monitor application (includes reads as a result of FINDs, reads as a result from FILEs for LT pool, and reads for records in update mode).

r-read
Real I/O reads per second for this performance monitor application.

v-read
VFA reads per second for this performance monitor application.

t-write
Number of writes per second for this performance monitor application.

r-write
Real I/O writes per second for this performance monitor application.

v-write
VFA writes per second for this performance monitor application.

Normal response to ZPERF POOL:

```
DXC5157I CME i  hh.mm.ss PERF  POOL
Appl  L1-Pool L2-Pool L3-Pool L4-Pool L5-Pool L6-Pool L7-Pool L8-Pool
SYST  l1-pool l2-pool l3-pool l4-pool l5-pool l6-pool l7-pool l8-pool
DFT1  l1-pool l2-pool l3-pool l4-pool l5-pool l6-pool l7-pool l8-pool
appl  l1-pool l2-pool l3-pool l4-pool l5-pool l6-pool l7-pool l8-pool
⋮
```

Where:

appl
Performance monitor application.

l1-pool
Number of L1 long-term pool dispenses per second.

l2-pool
Number of L2 long-term pool dispenses per second.

l3-pool
Number of L3 long-term pool dispenses per second.

l4-pool
Number of L4 long-term pool dispenses per second.

l5-pool
Number of L5 long-term pool dispenses per second.

l6-pool
Number of L6 long-term pool dispenses per second.

l7-pool
Number of L7 long-term pool dispenses per second.

l8-pool
Number of L8 long-term pool dispenses per second.

Normal response to ZPERF SEQ:

```
DXC5271I CME  i  hh.mm.ss PERF  SEQ
Seq-F    Records      Blocks     Mean IOQT    Max IOQT
nnnn       s-rec       s-blk          ioqt    max-ioqt
```

Where:

nnnn
Sequential file name

s-rec
Number of logical I/O per second.

s-blk
Number of physical I/O per second.

ioqt
Average of physical I/O queue time.

max-ioqt
Max physical I/O queue time.

Normal response to ZPERF SER:

```
DXC5275I CME  i  hh.mm.ss PERF  SER
  Lock per ecb..................locks
        Lock contention per ecb.......l-cont
        %Lock contention wait.........l-wait
        Dispatch cycle per ecb........disp
        Dispatch redrive per ecb......d-redr
```

Where:

locks
Number of locked dispatch cycles per ECB

l-cont
Number of locked dispatch contentions per ECB

l-wait
Percentage of lock contention wait

disp
Number of dispatch cycles per ECB

d-redr
Number of dispatch re-drives per ECB

Normal response to ZPERF SPI:

```
DXC5154I CME i  hh.mm.ss PERF  SPI
APPL.........................a-spi
CPU..........................c-spi
I/O..........................i-spi
SYSTEM I/O...................s-spi
MSG..........................m-spi
VFA..........................v-spi
```

Where:

a-spi
Application SPI (application system performance indicator).

c-spi
CPU SPI (CPU system performance indicator).

i-spi
I/O SPI (I/O system performance indicator).

s-spi
System I/O SPI (System I/O system performance indicator).

m-spi
MSG SPI (Message system performance indicator).

v-spi
VFA SPI (VFA system performance indicator).

Note: A SPI is a indicative number calculated by the performance monitor. Changed SPIs indicate that the ALCS system and/or applications have changed and now behave differently.

Normal response to ZPERF SU:

```
DXC5166I CME i  hh.mm.ss PERF  SU
                         ---- Peak ECB ------
SU     ECB %   Mean      Max  Act  Prog   Tod
type   usage   mean      max  act  prog   tod
```

Where:

type
Storage Unit type. Either TYPE-1, or TYPE-2, or TYPE-3

usage
Percentage of ECBs that use this storage unit type.

mean
Mean number of Storage Units required by ECBs that use this storage unit type, or * when no information is available.

For the ECB that has used the maximum storage units of a single type since restart, or ZPERF SU,RESET the following is displayed. (Note that these values are obtained when the ECB terminates).

max
Maximum number of Storage Units of this type used by a single ECB.

act
Input message action code or blanks for peak entry.

prog
Program name for peak entry.

tod
Tod clock value in format *yyyy.ddd hh.mm.ss* for peak entry.

Normal response to ZPERF SYS:

```
DXC5153I CME i  hh.mm.ss PERF  SYS
% MVS CPU time.............time
% ALCS WAIT time............wait
Cycle time.................cycle
Nbr of TCB's...............tcb
Nbr of CPU's...............cpu
DASD I/O per second........dasd
I/O queue time.............queue
I/O per ECB................i/o
Msg's per second...........msg
ECB's per second...........ecbs
```

Where:

time
CPU utilization (percentage in format *nn.nn*). Calculated over the standard processors only. IBM System z Application Assist Processors (zAAP) and IBM System z9 Integrated Information Processors (zIIP) are not considered in the calculation. This value does not include blocked I/O to a test database.

wait
ALCS wait (percentage in format *nn.nn*).

cycle
performance monitor cycle time. This value should be close to 1.6 seconds.

tcb
Number of CPU loop TCB's.

cpu
Total number of processors in this LPAR. The zIIP and zAAP processors are not counted.

dasd
Total number of DASD I/O per second (this number does not include test database I/O).

queue
Average I/O queue time in milliseconds.

i/o
Average DASD I/O per second per ECB.

msg
Number of input messages per second.

ecbs
Number of ECBs per second.

Note: *time* and *wait* are approximate percentages.

Normal response to ZPERF TASK:

```
DXC5159I CME i  hh.mm.ss PERF  TASK
Other.....................aaaa
Timer.....................bbbb
CPU loop..................cccc
SEQ files.................dddd
MQS support...............eeee
TCP/IP support............ffff
LU 6.2 support............gggg
SQL support...............hhhh
PPI support...............iiii
SLC support...............jjjj
WAS support...............kkkk
```

Where *aaaa*, *bbbb*, *cccc*, *dddd*,, *eeee*, *ffff*, *gggg*, *hhhh*,, *iiii*, *jjjj*, and *kkkk* are approximate CPU utilization percentages (in format *nn.nn*) for each type of ALCS task.

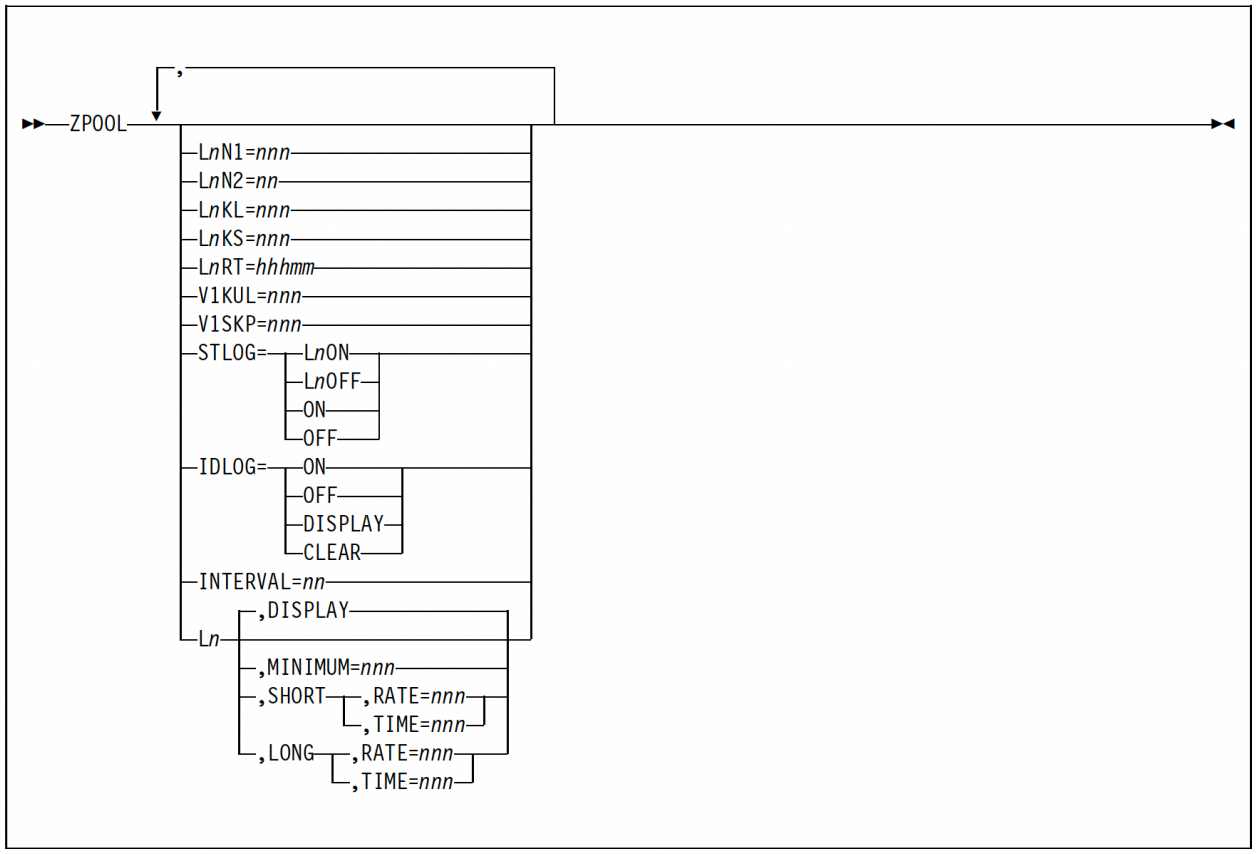
ZP00L -- Alter/display pool file status

Use the ZP00L command -- ***from any CRAS authorized terminal*** -- to display pool counts and the options that control the processing of the ALCS pool file.

Use the ZP00L command -- ***from a Prime CRAS authorized terminal only*** -- to modify the options that control processing of the ALCS pool file, or to set threshold values for the long-term pool activity monitor.

There are two kinds of pool support used by ALCS. These are called type 1 and type 2 pool support. Type 1 pool support is used in ALCS/MVS/XA and ALCS Version 2 Release 1.1, whereas type 2 pool support can only be used in ALCS Version 2 Release 1.3 and subsequent releases. See *ALCS Concepts and Facilities* for a full explanation of pool support.

The format of the command is:



ZP00L with no parameters displays the pool counts and the current setting of some of the options.

Where:

n
The record size defined in the ALCS generation.

L*n*N1=*nnn*
When a short-term pool-file record is dispensed to an application, but is not released by the application then ALCS will eventually redispense the record as it is timed out. Use this parameter to set the number of loops round the short-term pool between dispense and timeout. Specify a value between 1 and 189.

L*n*N2=*nn*
When a short-term pool record is released by an application then ALCS treats the record as available for redispense but only after a time interval. Use this parameter to set the number of loops round the short-term pool between release and redispense. Specify a value between 1 and 25.

L*n*KL=*nnn*
Use this parameter to alter the long-term pool keypoint update level. Specify a numeric value between 1 and 999.

L*n*KS=*nnn*
Use this parameter to alter the short-term pool keypoint update level. Specify a value between 1 and 999.

L*n*RT=*hhhmm*
Short-term pool recycle time in hours and minutes. If the interval between short-term pool recycles is less than *hhhmm*, ALCS sends an attention message to the RO CRAS. *hhh* is a value between 001 and 499. *mm* is a value between 00 and 59.

V1KUL=*nnn*
Number of pool-file addresses that ALCS dispenses before saving the directory status information on DASD. Specify a numeric value between 1 and 999.

V1SKP=*nnn*
Number of pool-file addresses that ALCS skips at restart after an unplanned shutdown. Specify a numeric value between 1 and 999. This value must be greater than the keypoint update parameter (**KUL**=). See *ALCS Installation and Customization* for further details.

STLOG={L*n*|ON|L*n*OFF}
Use this parameter to enable (L*n*ON) or disable (L*n*OFF) short-term pool event logging for record size L*n*. Event logging is disabled unless ZP00L has been used to enable it.

When event logging is enabled, ALCS records dispenses, finds, files, and releases for short-term pool records of this size. You can display these events using the ZDFIL command (see “ZDFIL -- Display contents of a DASD record” on page 177 for an explanation of how to use this command).

When short-term pool error logging is enabled, ALCS also writes these events to the ALCS diagnostic file. (See “Short-term pool errors” on page 357 for a description of pool usage errors printed by the ALCS diagnostic file processor.)

STLOG={OFF|ON}
Use this parameter to disable (OFF) or enable (ON) short-term pool error logging. Error logging is enabled unless ZP00L has been used to disable it.

IDLOG={ON|OFF}
Use this parameter to enable (ON) or disable (OFF) the counting of pool dispenses by record ID.

IDLOG=DISPLAY
Use this parameter to display a table of dispense counts.

IDLOG=CLEAR
Use this parameter to display a table of dispense counts and then to clear those counts to zero.

INTERVAL=*nn*
The long-term pool activity monitor samples the pool counts every minute. It calculates the dispense rate and time to depletion over time intervals specified in minutes by this parameter.

Every minute, it calculates the amount of pool depleted during the preceding interval's worth of minutes. This is shown in the ZP00L L*n*,DISPLAY response as "pool usage during the last smoothed interval". It also calculates the dispense rate and time to depletion based on the preceding interval's worth of depleted records, and compares these with the "short rate" and "short time" thresholds described below.

At the end of every interval, it calculates the amount of pool depleted during that interval and stores amounts for the most recent 24 intervals. These are shown in the ZP00L L*n*,DISPLAY response as "pool usage during the last 24 intervals". It also calculates the dispense rate and time to depletion based on the preceding 24 intervals' worth of depleted records, and compares these with the "long rate" and "long time" thresholds described below.

There is one time interval for all pool sizes. Specify a numeric value between 2 and 60. The default time interval is 10 minutes.

MINIMUM=*nnn*
This parameter specifies the lowest number of records in a pool you are prepared to accept before a warning is given. Set this value for each of the pool sizes that you are using. Specify a numeric value (minimum is 100).

SHORT,RATE=*nnn*
This parameter specifies the highest usage rate in records per second you are prepared to accept over a period of one monitor interval before a high usage warning is given. Set this value for each of the pool sizes that you are using. Specify a numeric value (minimum is 1).

SHORT,TIME=*nnn*
This parameter specifies the minimum remaining pool you are prepared to accept before a warning is given. It is expressed in hours as the time to exhaustion based on the current short rate. Set this value for each of the pool sizes that you are using. Specify a numeric value (minimum is 1).

LONG,RATE=*nnn*
This parameter specifies the highest usage rate in records per second you are prepared to accept over a period of 24 monitor intervals before a high usage warning is given. Set this value for each of the pool sizes that you are using. Specify a numeric value (minimum is 1).

LONG,TIME=*nnn*
This parameter specifies the minimum remaining pool you are prepared to accept before a warning is given. It is expressed in hours as the time to exhaustion based on the current long rate. Set this value for each of the pool sizes that you are using. Specify a numeric value (minimum is 1).

DISPLAY
Use this parameter to display the pool activity monitor values set for any particular pool size. The display also indicates the pool usage during the last 24 intervals. This is the default for ZP00L L*n* with no additional parameters.

Migration consideration:

The parameters on this command are used with either type 1 or type 2 short-term pool support as follows:

L*n*KL
Use with type 2 long-term pool support only.

L*n*KS, L*n*N1, L*n*N2, STLOG
Use these with type 2 short-term pool support only.

L*n*RT, V1SKP, V1KUL
Use these with type 1 short-term pool support only.

Normal responses without parameters

The normal responses to the ZP00L command vary depending on whether you are using type 1 or type 2 pool support. Only some of these responses apply after you have migrated to type 2 pool support.

For a full explanation of partitioning and of migration considerations, see *ALCS Installation and Customization*.

To ZP00L (without parameters) for Type 1 long-term and type 1 short-term support

```
DXC8700I CMD i hh.mm.ss POOL  Pool file counts and dispense options
Long term pool
          Total      Available   Dispense
        records        records       rate
L1LT      nnnnn          nnnnn       nnnn
L2LT      nnnnn          nnnnn       nnnn
L3LT      nnnnn          nnnnn       nnnn
⋮
Short term pool - Type 1 support
          Total  Dispense  Recycle time   Restart  Keypoint
        records      rate       hhh  mm     skips     level
L1ST      nnnnn      nnnn       nnn  nn       nnn       nnn
L2ST      nnnnn      nnnn       nnn  nn       nnn       nnn
L3ST      nnnnn      nnnn       nnn  nn       nnn       nnn
⋮
```

Where:

Total records
The total number of records in this pool.

Available records
The total number of records in this pool that are available for dispense.

Dispense rate
Number of records being dispensed each second (smoothed).

Recycle time
The time interval for reuse of unreleased short-term pool records below which an attention message is sent to the operator.

Restart skips
The number of short-term pool records skipped at restart time, to prevent redispense of records dispensed before the system outage.

Keypoint level
The number of updates to a pool directory which triggers keypointing.

To ZPOOL (without parameters) for Type 1 long-term and type 1 short-term support (ordinal range restricted)

```
DXC8700I CMD i hh.mm.ss POOL  Pool file counts and dispense options
Long term pool
          Total      Available   Dispense
        records        records       rate
L1LT      nnnnn          nnnnn       nnnn
L2LT      nnnnn          nnnnn       nnnn
L3LT      nnnnn          nnnnn       nnnn
⋮
Short term pool - Type 1 support - Ordinal range restricted
          Total  Maximum  Dispense  Recycle time   Restart  Keypoint
        records  ordinal      rate       hhh  nm     skips     level
L1ST      nnnnn    nnnnn      nnnn       nnn  nn       nnn       nnn
L2ST      nnnnn    nnnnn      nnnn       nnn  nn       nnn       nnn
L3ST      nnnnn    nnnnn      nnnn       nnn  nn       nnn       nnn
⋮
```

To ZPOOL (without parameters) for Type 1 long-term and type 2 short-term support (ordinal range restricted)

```
DXC8700I CMD i hh.mm.ss POOL  Pool file counts and dispense options
Long term pool
           Total      Available   Dispense
         records        records       rate
L1LT       nnnnn          nnnnn       nnnn
L2LT       nnnnn          nnnnn       nnnn
L3LT       nnnnn          nnnnn       nnnn
⋮
Short term pool - Type 2 support - Ordinal range restricted
           Total  Minimum  Dispense   Percent  Directory Loops  Keypoint
         records  ordinal      rate  occupied  timeout/release     level
L1ST       nnnnn    nnnnn      nnnn        nn      nnn      nn       nnn
L2ST       nnnnn    nnnnn      nnnn        nn      nnn      nn       nnn
L3ST       nnnnn    nnnnn      nnnn        nn      nnn      nn       nnn
⋮
```

Where:

Directory Loops - timeout
The number of times ALCS searches the short-term pool directory looking for an available record before treating a record that has been dispensed but not released as available for redispense.

Directory Loops - release
The number of times ALCS searches the short-term pool directory looking for an available record before treating a record that has been released as available for redispense.

Maximum ordinal
The highest ordinal that ALCS dispenses when using type 1 short-term pool support with restricted range.

Minimum ordinal
The lowest ordinal that ALCS dispenses when using type 2 short-term pool support with restricted range.

Percent occupied
The percentage of records that the short-term pool directory shows cannot be dispensed. ALCS calculates this percentage dynamically as it dispenses short-term pool. If the dispense rate is too low, ALCS cannot calculate the percentage and displays * in this column.

Other fields are described in To ZP00L (without parameters) for Type 1 long-term and type 1 short-term support.

To ZP00L (without parameters) for Type 2 long-term and type 2 short-term support (after running Recoup)

```
DXC8700I CMD i hh.mm.ss POOL  File counts and dispense options
Allocatable pool
     Allocatable    Allocatable   Available  Dispense  Keypoint
           total  less reserved     records      rate     level
L1         nnnnn          nnnnn       nnnnn      nnnn       nnn
L2         nnnnn          nnnnn       nnnnn      nnnn       nnn
L3         nnnnn          nnnnn       nnnnn      nnnn       nnn
⋮
Short term pool - Type 2 support
           Total  Dispense   Percent  Directory Loops  Keypoint
         records      rate  occupied  timeout/release     level
L1ST       nnnnn      nnnn         *      nnn      nn       nnn
L2ST       nnnnn      nnnn         *      nnn      nn       nnn
L3ST       nnnnn      nnnn         *      nnn      nn       nnn
⋮
```

Where

Allocatable total
The total of allocatable records in this pool (including short-term and fixed records).

Allocated less reserved
The total of allocatable records in this pool (excluding short-term and fixed records).

The following message is added when partitioning is used on a system to reserve part of the pool for live system use and part of the pool for test system use.

```
Pool partitioning in use -- n percent reserved for test users
Available records adjusted for pool partitioning
```

The following message is added when a test database is in use.

```
Test database in use
```

The following message is added when type 2 short-term pool error logging is disabled.

```
ST Pool error logging is disabled
```

The following message is added when type 2 short-term pool event logging is enabled:

```
ST Event logging is enabled for LnST pool
```

where *n* is the record size for which logging is enabled.

The following message is added when ID dispense counting is enabled:

```
ID Dispense counting is enabled
```

Normal responses with parameters

To ZPOOL IDLOG=DISPLAY (display of pool dispense counts by record ID)

```
DXC8724I CMD i  hh.mm.ss POOL    POOL  Dispense counts since hh.mm.ss
                                       Counting terminated   hh.mm.ss
LT pool types
id  size   count   id  size   count   id  size   count
iiii ll nnnnnnnn   iiii ll nnnnnnnn   iiii ll nnnnnnnn
iiii ll nnnnnnnn   iiii ll nnnnnnnn   iiii ll nnnnnnnn
iiii ll nnnnnnnn   iiii ll nnnnnnnn   iiii ll nnnnnnnn
⋮
ST pool types
id  size   count   id  size   count   id  size   count
iiii ll nnnnnnnn   iiii ll nnnnnnnn   iiii ll nnnnnnnn
iiii ll nnnnnnnn   iiii ll nnnnnnnn   iiii ll nnnnnnnn
iiii ll nnnnnnnn   iiii ll nnnnnnnn   iiii ll nnnnnnnn
⋮
```

Where:

iiii
: Record ID in 2 character or 4 hexadecimal character format.

ll
: Record size in the format L1, L2 etc.

nnnnnn
Count of dispenses during the stated period.

Note: If dispense counting is enabled then the `Counting terminated` line does not appear.

To ZPOOL L*n*,DISPLAY (the pool activity monitor)

```
DXC8735I CMD i  hh.mm.ss POOL  LnLT Monitor thresholds
Monitor interval ................... nnn minutes
Minimum pool count ................. nnn
Short dispense rate threshold ...... nnn per second
Short depletion time threshold ..... nnn hours
Long dispense rate threshold ....... nnn per second
Long depletion time threshold ...... nnn hours

Long-term pool usage during the last 24 intervals
 Oldest in first row, starting on the left
 Most recent in last row, ending on the right
    ppp    ppp    ppp    ppp    ppp    ppp
    ppp    ppp    ppp    ppp    ppp    ppp
    ppp    ppp    ppp    ppp    ppp    ppp
    ppp    ppp    ppp    ppp    ppp    ppp
Long-term pool usage during the last smoothed interval
    sss
```

Where:

L*n*LT
The pool size for which these values apply.

nnn
The value previously set by a ZP00L command.

ppp
The number of records depleted during a single interval.

sss
The number of records depleted during the preceding interval's worth of minutes.

ZPURG -- Purge an entry or purge VFA

Use the ZPURG command -- ***from a Prime CRAS authorized terminal only*** -- to force a specified entry to exit, or to purge VFA. To force a specified entry to exit you should first use the ZDECB command to obtain the address of the ECB.

The format of the command is:

Where:

ecb_address
Address of ECB to be purged, up to 8 hexadecimal digits, leading zeros optional.

Force
Force the ECB to be purged. Use `Force` only if a previous ZPURG *`ecb_address`* does not terminate the entry.

ZPURG *`ecb_address`* purges an entry at the next opportunity when the entry is added to or removed from an internal queue.

By contrast, ZPURG *`ecb_address`*`,Force` purges an entry immediately, regardless of its current activity. ALCS unholds any records or resources held by the entry, and unassigns any sequential files assigned to the entry. ALCS also marks the storage unit containing the ECB for the entry, together with any storage units that are chained from it, as *quarantined*. This ensures that the storage unit(s) will not be dispensed again until ALCS is restarted.

Note: Use of the ZPURG *`ecb_address`*`,Force` command can compromise the operation and integrity of ALCS and you might have to re-start ALCS.

VFA
Use this option to write to DASD all the output records currently in VFA.

Restriction

The *ecb_address* option requests confirmation, see “Confirmation of commands” on page 60.

Normal responses

Normal response to ZPURG *`ecb_address`*:

```
DXC8355I CMD i hh.mm.ss PURG  Purge request accepted
```

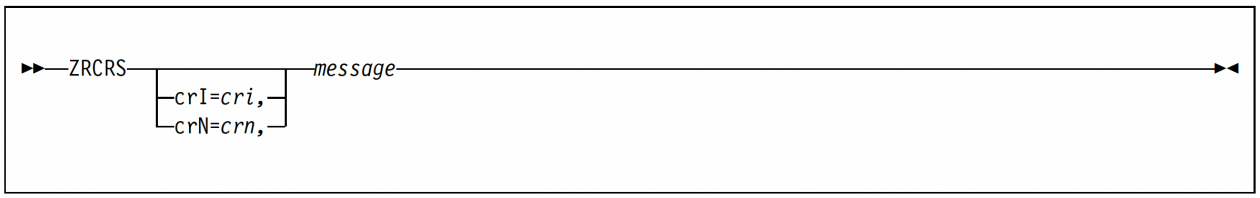
Normal response to ZPURG VFA:

```
DXC8357I CMD i hh.mm.ss PURG  All VFA records purged
```

ZRCRS -- Send a message to a CRAS printer

Use the ZRCRS command -- ***from any terminal*** -- to send a message to the RO CRAS, to a specified CRAS printer, or to the printer associated with a specified CRAS display.

The format of the command is:



Where:

cri
CRI of the CRAS display or printer (6 hexadecimal digits).

crn
CRN of the CRAS display or printer (1 to 8 alphanumeric characters).

message
One to 70 characters of text.

If no display or printer is specified, the message is directed to the RO CRAS.

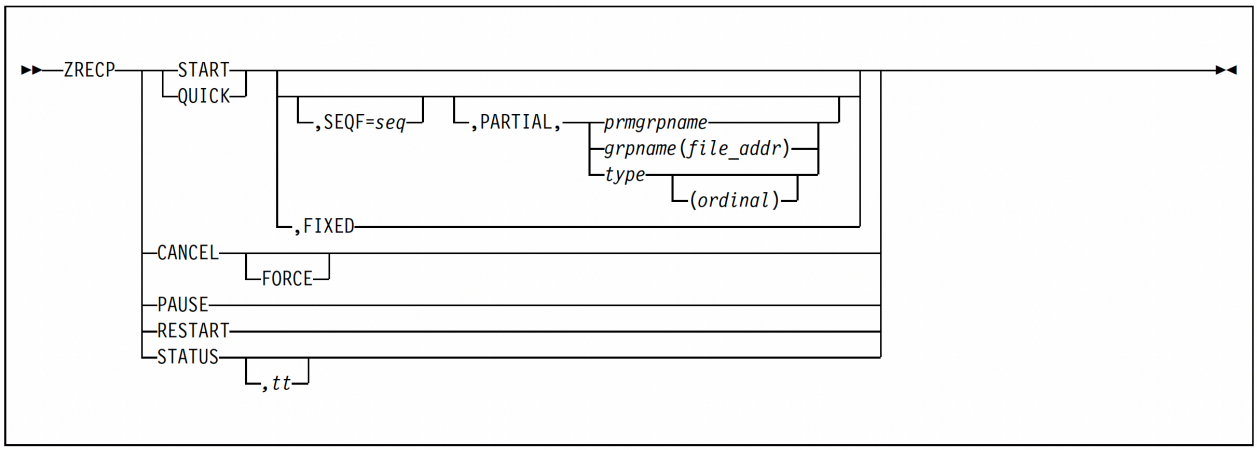
ZRECP -- Control Recoup

Use the ZRECP command -- ***from a Prime CRAS authorized terminal only*** -- to control the Recoup database validation utility.

Use ZRECP STATUS -- ***from any CRAS authorized terminal*** -- to display Recoup status.

Recoup uses significant system resource, so although it gives priority to other work, it is better run at off-peak times.

The format of the command is:



Where:

START
Start Recoup.

QUICK
Start reduced write Recoup.

SEQF=*seq*
Writes statistical information on the general sequential file *seq* for subsequent offline analysis.

PARTIAL,{*prmgrpname*|.* Ann Evans - Jan 2007 - shorten file_address for BOO file *grpname*(*file_addr*)|*type*[(*ordinal*)]}
Recoup performs validation only, without freeing any pool.

prmgpname
Recoup chain-chases only those chains that belong to the named prime group.

grpname
Recoup chain-chases records in any group (prime or non-prime) starting at the record specified in *file_addr* and with pointers to refer to records described by the group name *grpname* and any associated INDEX and GROUP macros.

file_addr
File address (8 hexadecimal digits).

type
Recoup chain-chases only those chains where the start-of-chain record has the fixed record type *type* (specified as a symbol).

ordinal
Recoup chain-chases only those chains where the start-of-chain record has the fixed record type *type* and the record ordinal *ordinal*. *ordinal* is 1 to 6 decimal digits.

FIXED
Recoup reads only fixed-file records, without freeing any pool. You can use Recoup in this way in order to verify that the start-of-chain fixed records have been correctly initialized.

CANCEL
Cancel Recoup.

FORCE
Cancel Recoup regardless the status of the Recoup Keypoint. When you issue ZRECP CANCEL FORCE, ALCS requires you to confirm the command. See “Confirmation of commands” on page 60

PAUSE
Pause Recoup. It is not necessary to issue this command if the system state alters (ZASYS command), since ALCS automatically pauses Recoup and sends a message to the operator.

RESTART
Restart Recoup (after a pause or a system outage).

STATUS
Recoup prints a report on its current status.

tt
Recoup produces status reports at intervals of tt minutes. *tt* is 1 or 2 decimal digits. Specify time interval of 0 minutes to suppress time-initiated reports. Recoup also produces a status report after validating each group of records.

For an overview of Recoup see Chapter 5, “Recoup,” on page 52; and for a detailed description of Recoup, see *ALCS Installation and Customization*.

Restriction

It is not possible for two Recoup runs (whether partial or full) to be in progress at the same time. To find out whether Recoup is active enter: ZRECP STATUS.

Normal responses

```
DXC8053I CMD i hh.mm.ss RECP  Not active


DXC8047I CMD i hh.mm.ss RECP  Started


DXC8051I CMD i hh.mm.ss RECP  Being cancelled
DXC8048I CMD i hh.mm.ss RECP  Cancelled


DXC8052I CMD i hh.mm.ss RECP  Being paused
DXC8049I CMD i hh.mm.ss RECP  Paused


DXC8050I CMD i hh.mm.ss RECP  Restarted


DXC8367I CMD i hh.mm.ss RECP  Being paused by system sate change
```

ALCS prints this message if the ZASYS command changes the system state while Recoup is running:

```
DXC8368I CMD i hh.mm.ss RECP  RESTART awaited
```

ALCS prints this message after an ALCS restart that interrupts a Recoup run. Reply with ZRECP RESTART or ZRECP CANCEL. Delay the reply if it is convenient to reach a particular system state before restarting Recoup.

```
DXC8370I CMD i hh.mm.ss RECP  Chain chase complete

Chain chase elapsed time hh.mm.ss
```

ALCS prints this message to show that the first phase of Recoup, called chain-chase, has completed. Recoup continues with the second phase, called directory build.

```
DXC8405I CMD i hh.mm.ss RECP  Directory build -- pool LnLT scan x complete
```

ALCS prints this message during directory build, where *n* is the size of the pool for which the directory records are being built, and *x* is the number of times the Recoup general file has been scanned.

```
DXC8371I CMD i hh.mm.ss RECP  Directory build complete
```

ALCS prints this message to show that the second phase of Recoup, called directory build, has completed. Recoup is no longer active.

```
DXC8381I CMD i hh.mm.ss RECP  Fixed mode run complete
```

ALCS prints this message to show that a Recoup fixed run has completed. Recoup is no longer active.

```
DXC8382I CMD i hh.mm.ss RECP  Partial run complete
```

ALCS prints this message to show that a Recoup partial run has completed. Recoup is no longer active.

Status messages

The format of the status message during chain chase is as follows:

```
DXC8404I CMD i hh.mm.ss RECP
Descriptor aaaa Group bbbbbbbb
This prime group  Total reads.nnn nnn nnn   Pool reads.mmm mmm mmm
This Recoup       Total reads.ppp ppp ppp   Pool reads.qqq qqq qqq
Errors I/O.rrrr ID.ssss RCC.tttt F/A.uuuu Control field.vvvv
```

Where:

aaaa
Name of the descriptor program currently being processed.

bbbbbbbb
Name of the prime group currently being processed.

nnn nnn nnn
Number of records read (fixed and pool) for this prime group.

mmm mmm mmm
Number of records read (pool only) for this prime group.

ppp ppp ppp
Number of records read (fixed and pool) so far in this Recoup run.

qqq qqq qqq
Number of records read (pool only) so far in this Recoup run.

rrrr
Total number of I/O errors so far in this Recoup run.

ssss
Total number of ID errors so far in this Recoup run.

tttt
Total number of record code check (RCC) errors so far in this Recoup run.

uuuu
Total number of invalid file addresses so far in this Recoup run.

vvvv
Total number of control information errors so far in this Recoup run. An example of this sort of error is as follows: Suppose a field in a record contains an item count. If the item count is higher than the number of items that the record can hold, then this is a control information error.

The format of the status messages during type 1 directory build is as follows:

```
DXC8405I CMD i hh.mm.ss RECP
Recoup directory build -- pool LnLT scan x in progress


DXC8405I CMD i hh.mm.ss RECP
Recoup directory build -- pool LnLT scan x complete
```

The format of the status message during fixed file tagging is as follows:

```
DXC8406I CMD i hh.mm.ss RECP
Recoup directory build -- Fixed file tagging type ordinal nnnn
```

The format of the status message during short-term pool tagging is as follows:

```
DXC8407I CMD i hh.mm.ss RECP
Recoup directory build -- Short term pool tagging in progress
```

The format of the status message during type 2 directory build is as follows:

```
DXC8408I CMD i hh.mm.ss RECP
Recoup directory build -- Recoup general file scan in progress
```

The format of the status message during fixed file / short-term pool scanning is as follows:

```
DXC8409I CMD i hh.mm.ss RECP
Recoup directory build -- Fixed file / ST pool scan in progress
```

ZRELO -- Control the relocating loader

About this task

The ALCS relocating loader is used to migrate a portion of a database from one system to another, for example from a live system to a test system. The migration involves five stages:

Procedure

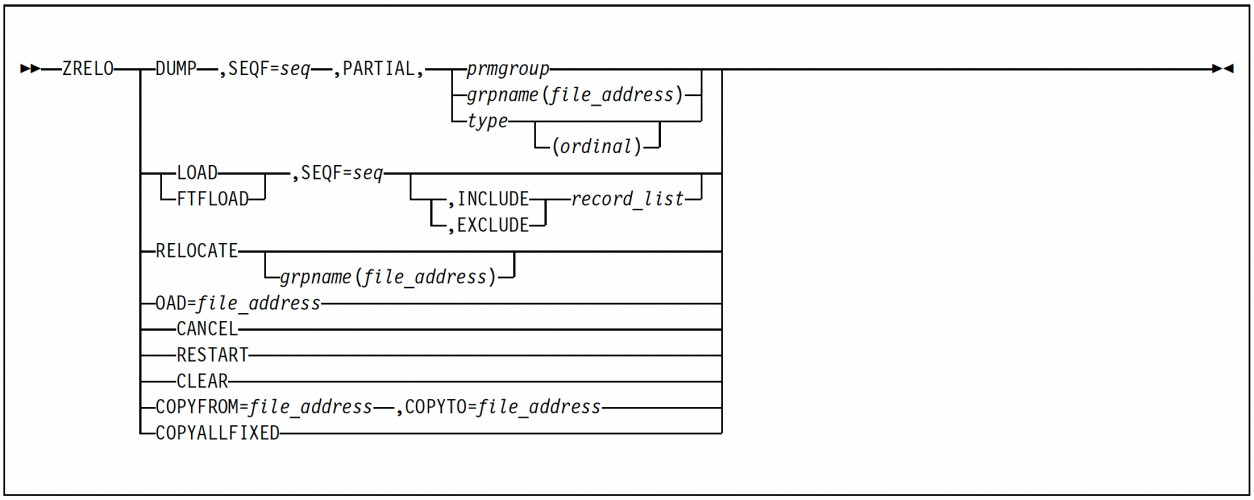
1. A sequential file containing the records to be migrated is produced on the migrate-from system (ZRELO DUMP).
2. The migrate-to system reads the records from the sequential file and places them in records obtained from long-term pool (ZRELO LOAD or ZRELO FTFLOAD).
3. Any imbedded file addresses are translated to the new addresses of the records on the migrate-to system (ZRELO RELOCATE).
4. The head of chain records are copied to the appropriate records in the migrate-to database (ZRELO COPYFROM or ZRELO COPYALLFIXED).
5. The relocate tables are cleared (ZRELO CLEAR).

Results

Use the ZRELO command in any system state -- ***from a Prime CRAS authorized terminal only*** -- to:

- Dump records to a sequential file
- Restore records from a sequential file
- Relocate restored records

The format of the command is:



Where:

DUMP,SEQF=*seq*
Dump records from a real-time database or general file to the general sequential file *seq*.

PARTIAL,{*prmgroup*|*grpname*(*file_address*)|*type*[(*ordinal*)]}
This parameter specifies which records to dump to the sequential file *seq*..

prmgroup
Dump the records in the prime group specified in *prmgroup* and refer-to records described in any associated INDEX and GROUP macros in the Recoup descriptor program.

grpname
Dump records in any group (prime or non-prime) starting at the record specified in *file_address* and refer-to records described by the group name *grpname* and any associated INDEX and GROUP macros in the Recoup descriptor program.

file_address
File address (8 hexadecimal digits).

type
Dump records with the fixed record type *type* (specified as a symbol) and refer-to records described in any associated INDEX and GROUP macros.

ordinal
Dump records with the fixed record type *type* and the record ordinal *ordinal* and refer-to records described in any associated INDEX and GROUP macros. *ordinal* is 1 to 6 decimal digits.

LOAD,SEQF=*seq*
Load records (fixed and pool) from the general sequential file *seq* to the database in long-term pool file records. This command also builds two relocate tables containing the migrate-from and the migrate-to addresses of the records. These tables are subsequently used by the ZRELO RELOCATE command.

FTFLOAD,SEQF=*seq*
This 'fixed-to-fixed' option is the same as ZRELO LOAD,**SEQF**=*seq* except that the fixed records read from the sequential file are placed directly into the fixed records on the database and not into pool records.

INCLUDE|EXCLUDE
Include or exclude specific records during a LOAD operation.

INCLUDE
Load only the records in *record_ list*.

EXCLUDE
Do not load the records in *record_list*.

Omit this parameter to load every record from the sequential file to the real-time database and general files.

record_list
A list of one or more record selections. A comma or space must separate each selection. A selection can be any of the following:

file_address
File address; 8 hexadecimal digits.

type
Any of the following:

GF-*nnn*
nnn is the general file number; 3 decimal digits.

#xxxxx
Fixed record type.

LsLTpool
Long-term pool record, where L*s* identifies the record size.

LsSTpool
Short-term pool record, where L*s* identifies the record size.

***type*(*n*)**
Where *n* is the record ordinal (decimal).

***type*(*n*--*n*)**
Where *n*--*n* is a range of record ordinals (decimal).

type*--*id
Where *id* is the record identifier; 2 alphanumeric characters or 4 hexadecimal digits.

FIXED
All fixed-file records.

POOL
All pool-file records.

SIZEL*n*
All records of size L*n*.

RELOCATE
This command starts a special type of partial Recoup run. If an imbedded file address (forward chain, backward chain or reference indicated by INDEX macro) found during chain chasing appears in Relocate Table 1 as a migrate-from address then that address is changed to the corresponding migrate-to address before the refer-to record is read. All records containing migrate-from addresses are filed after the migrate-from addresses have been replaced by the migrate-to addresses.

grpname
Group name (not necessarily a prime group)

file_address
The migrate-from file address of the record at start of chain.

If you use ZREL0 RELOCATE without specifying a group name, it acts on all those chains starting with fixed records with addresses in Relocate Table 2.

Note: ZREL0 RELOCATE can be used only after a ZREL0 LOAD command.

OAD=*file_address*
file_address is a migrate-from file address (8 hex digits)

Use this command to inspect Relocate Table 1 and display the migrate-to address corresponding to a migrate-from address.

CANCEL
Cancel ZREL0 after ZREL0 DUMP or ZREL0 LOAD or ZREL0 RELOCATE.

RESTART
Restart ZREL0 after an ALCS failure.

CLEAR
Release records from both relocate tables.

COPYFROM=*file_address*,COPYTO=*file_address*
Copy one record to another.

The records must be of equal size.

Notes:

1. Both of the file addresses are migrate-to file addresses.
2. The relocate tables are not used.

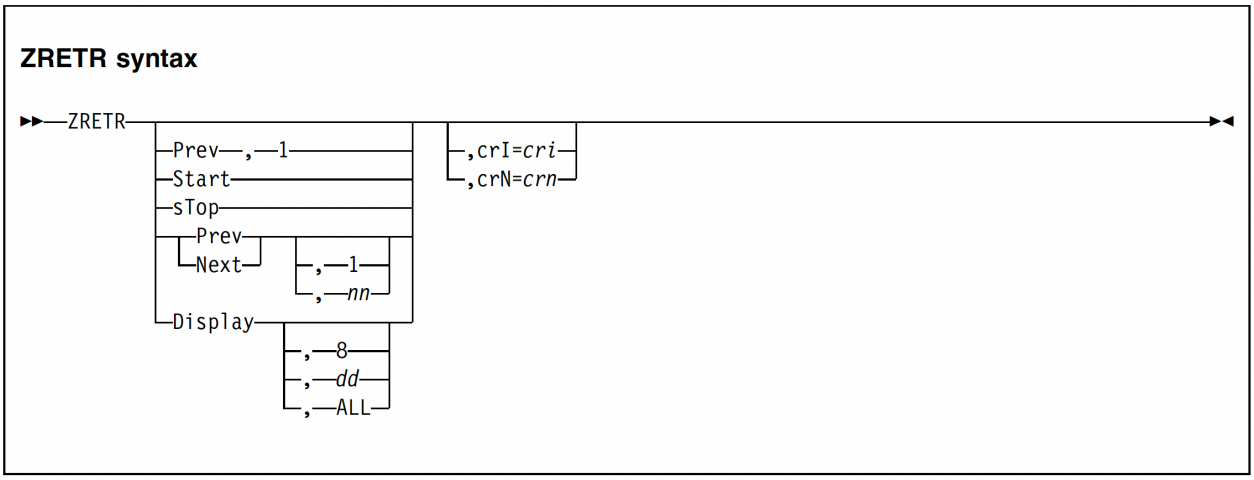
COPYALLFIXED
Enter this command to read all the pool records obtained during ZREL0 LOAD to store records which originated as fixed records on the migrate-from system and copy these pool records to the appropriate fixed records on the migrate-to system.

ZRETR -- Retrieve (recall) previously entered input message

Use the ZRETR command to display input messages that were previously issued. This command can be issued ***from any terminal*** unless the `CRI` or `CRN` parameter is specified. The `CRI` or `CRN` parameter can only be specified from a ***Prime CRAS authorized terminal***.

ZRETR `Prev` or ZRETR `Next` display the input message on the screen. Pressing the ENTER key causes the message to be re-issued. ZRETR `Display` displays a list of messages which were previously issued. Messages in this list are not re-issued if the ENTER key is pressed.

The format of the command is:



Where:

Start
Starts retrieve for this terminal. Subsequent input messages are saved for retrieval.

sTop
Stops retrieve for this terminal. Input messages are no longer saved for retrieval. Messages previously saved are discarded.

Prev
Requests the previous message to be displayed. You can display messages from the most-recently saved or the most-recently shown message to the oldest message saved or shown by issuing a series of ZRETR `Prev` commands. Wrapping occurs when the oldest message is retrieved.

Next
Requests the next message to be displayed. You can display messages from the oldest message saved or shown to the most-recently saved or shown message by issuing a series of ZRETR `Next` commands. Wrapping occurs when the most-recent message is retrieved.

Notes:

1. Whenever a non-ZRETR command is received, ZRETR `Prev` is reset to start with the most-recently saved or shown message and ZRETR `Next` is reset to the oldest message saved or shown.
2. ZRETR with no parameters defaults to ZRETR `Prev 1`
3. PF key input is not saved for later retrieval.
4. 3270 input is not saved for later retrieval.

nn
Specifies the stepping factor to be used to select the message or command to be displayed. For example, 1 (the default) indicates that the message immediately next or previous is to be displayed,

2 indicates that the adjacent next or previous message should be skipped, and in general *nn* indicates that *nn*-1 adjacent messages should be skipped.

Display
Requests a list of the messages which have been saved by the retrieve facility.

dd
Specifies the number of messages to be displayed. The default is eight. The most recently saved message is the last message to be displayed.

ALL
Display all messages that have been saved.

crI=*cri*|(crN=*crn*
Where:

crI=*cri*
CRI of another terminal.

crN=*crn*
CRN of another terminal.

You can use this parameter ***from a Prime CRAS authorized terminal*** to start or stop the retrieve function for another terminal, or to display input messages that are entered at another terminal.

Normal responses

To a request to start or stop retrieve (ZRETR `Start|sTop`):

```
DXC8028I CMD i hh.mm.ss RETR OK
```

To a request for a previously saved input message (ZRETR `Prev|Next`):

```
message text
```

To a request for a list of previously saved input messages (ZRETR `Display`):

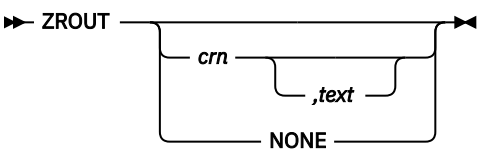
```
message text
message text
message text
```

ZROUT -- Set routing for a terminal

Use the ZROUT command -- ***from any terminal*** -- to establish or change the application to which ALCS routes messages from the input terminal.

You can also use the ZROUT command -- ***from any terminal*** -- to send a message to a specific ALCS application.

The format of the command is:



Where:

ZROUT with no parameters displays the name of the application to which ALCS routes messages from this terminal.

crn
CRN of the application to which ALCS is to route messages from this terminal.

or:

crn,text
Send a message to the application with this CRN.

where:

text
Text of the message to be sent.

Note: The routing for the terminal remains unchanged.

NONE
Remove routing for the terminal. That is, only ALCS commands will be allowed from this terminal.

For the ALCS equivalent of the TPF ZROUT command, see "ZACOM -- Alter communication resource information" on page 69.

Normal responses

```
DXC8520I CMD i hh.mm.ss ROUT  Not routed to an ALCS application
```

Only operator commands are allowed from this terminal.

```
DXC8521I CMD i hh.mm.ss ROUT  Routed to ALCS application crn
```

Where:

crn
CRN of the application to which ALCS routes application messages from this terminal. If the application can process operator commands, then ALCS also routes operator commands, except ZROUT and ZLOGF, to the application.

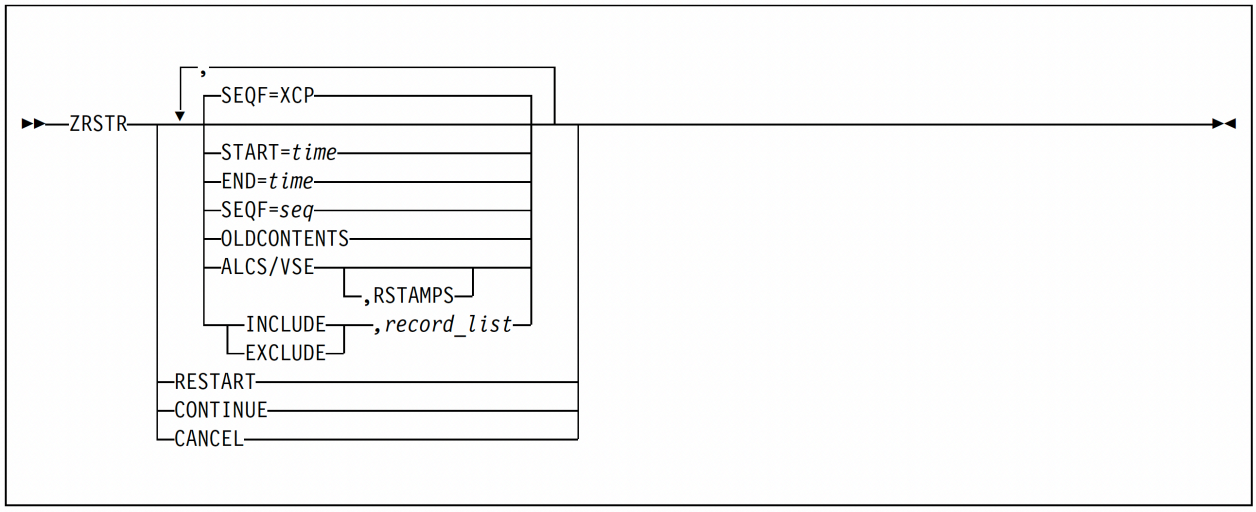
The following responses are self-explanatory:

```
DXC8522I CMD i hh.mm.ss ROUT  Terminal routing removed

DXC8523I CMD i hh.mm.ss ROUT  Terminal routing updated
```

ZRSTR -- Restore logged records

Use the ZRSTR command -- ***from a Prime CRAS authorized terminal only*** -- to restore the DASD record updates from the ALCS update log file. See “Logging and restoring database updates” on page 18 for information. The system must be in IDLE state. No other utilities can be active.

The format of the command is:



Where:

START=*time*
The TOD clock time and date of the first logged records to restore. Omit START to start at the beginning of the ALCS update log file.

END=*time*
The TOD clock time and date of the last logged records to restore. Omit END to continue to the end of the ALCS update log file.

time
MVS GMT (TOD clock) time and date in the format *hh.mm.ss yyddd*, where:

hh.mm.ss
Hours, minutes, and seconds.

yyddd
The 2-digit year and 3-digit day of the year.

SEQF={XCP|*seq*}
Symbolic name of the input general sequential file. This is the ALCS update log file. Note that a data set has one name (for example "LOG") when it is an output system sequential file, and a different name (for example "XCP") when it is an input general sequential file.

OLDCONTENTS
Restore records to their before-update contents, from either the backward or the merged ALCS update log file. The usual reason for using the before-update contents is that a database record has been corrupted. Instead of restoring the entire database, you can just restore the pre-corruption state of the individual record.

Omit this parameter to restore records to their after-update contents, from either the forward or the merged logging sequential file.

ALCS/VSE
The sequential file contains records logged on an ALCS/VSE system.

RSTAMPS
When using ZRSTR to read a RTA tape produced by pre-VFA ALCS/VSE, add the operand RSTAMPS to the end of the message.

The additional operand RSTAMPS tells ALCS to reposition the time stamps in the long-term pool records. This is needed because the displacement of pre-VFA ALCS/VSE time stamps is slightly different from the displacement of ALCS V2 time stamps.

INCLUDE
Restore only the records in *record_list*.

EXCLUDE
Do not restore the records in *record_list*.

Omit INCLUDE and EXCLUDE to restore all records from the logging sequential file, apart from the following records, which are not restored:

- Pool file directory records, and some other records reserved for ALCS use
- Resource control records for CRAS terminals (#CPRCR).

record_list
Specify a list of up to 40 record selections in *record_list*. A comma or space must separate each selection. A selection can be any of the following:

file_address
File address; 8 hexadecimal digits.

type
Any of the following:

#xxxxx
Fixed record type.

LsLTpool
Long-term pool record, where L*s* identifies the record size.

LsSTpool
Short-term pool record, where L*s* identifies the record size.

type(n)
Where *n* is the record ordinal (decimal).

type(n--n)
Where *n*--*n* is a range of record ordinals (decimal).

type--id
Where *id* is the record identifier; 2 alphanumeric characters or 4 hexadecimal digits.

FIXED
All fixed file records.

POOL
All pool file records.

STPOOL
All short-term pool file records.

LTPOOL
All long-term pool file records.

SIZEL*n*
All records of size L*n*.

{RESTART|CONTINUE}
Restart the restore after an ALCS restart, or restore the next input sequential file.

CANCEL
Cancel the restore.

When ALCS writes a record to the ALCS update log file, it writes a time stamp in the record using the TOD clock (system GMT). ZRSTR uses these time stamps to calculate the date and time of the database updates. It does not restore records logged before the **START**=, or after the **END**=, date and time.

Normal responses

```
DXC8555I i hh.mm.ss RSTR  Summary of restore
    Seq file reads......nnnnnnn
    DASD writes.........nnnnnnn
    Records not written.nnnnnnn
    File address errors.nnnnnnn
```

ZRSTR sends this summary message when end of file condition occurs on the input sequential file, or when it finds a time stamp later than the time specified by the END parameter (completion of restore). Note that the count of DASD writes can differ from the count of sequential file reads when records are selectively restored.

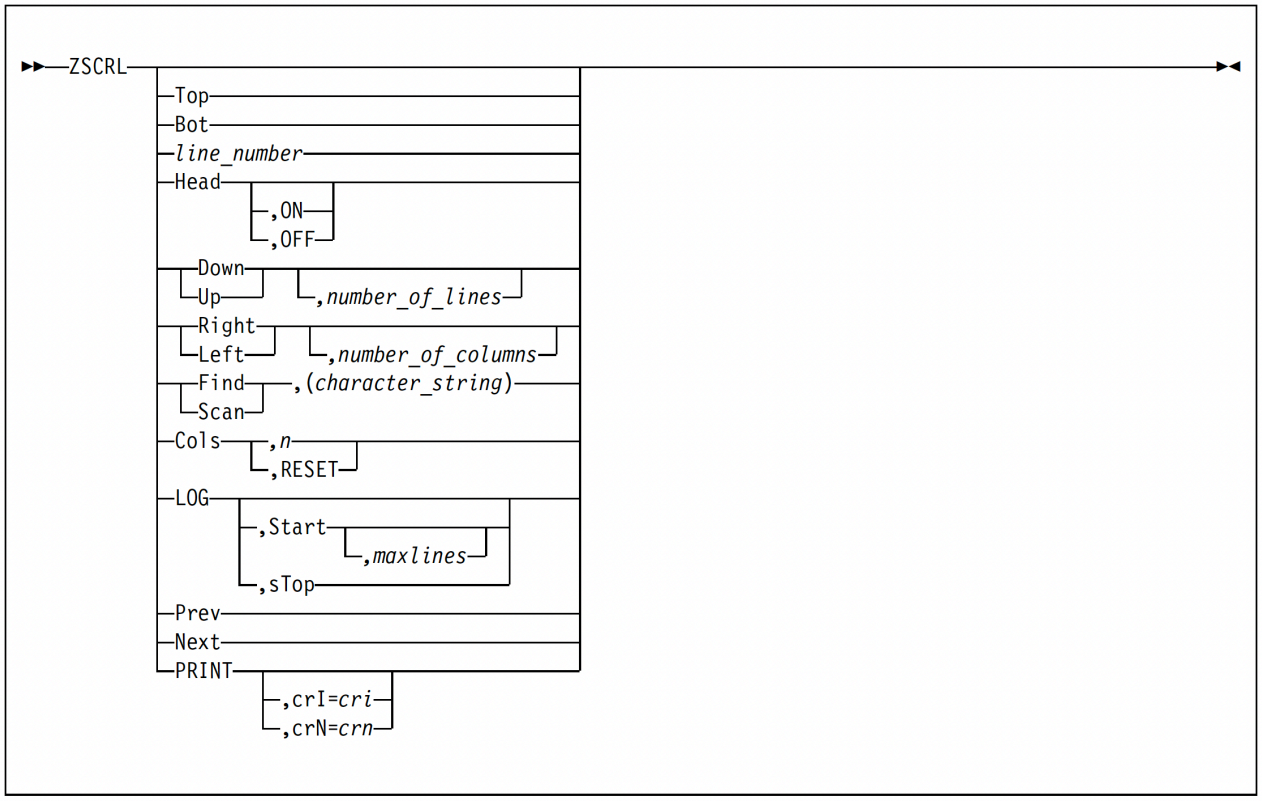
```
DXC8558I i hh.mm.ss RSTR  End of sequential file
    Mount next sequential file and enter 'ZRSTR CONTINUE'
```

ZRSTR has reached the end of the input sequential file without finding a record with a time stamp later than the time specified by the END parameter. Use the ZASEQ command to change the specification of the input sequential file to the next ALCS update log file. Then issue the ZRSTR CONTINUE command. If there are no more files to process, issue ZRSTR CANCEL to end the restore.

ZSCRL -- Scroll information on the screen

Use the ZSCRL command -- ***from any terminal*** -- to view other parts of the output file currently being used by the terminal. Alternatively, use the ZSCRL command to swap to a different file attached to the terminal. See “Scrolling displays” on page 58 for more information about scrolling.

The format of the command is:



Where:

Top|Bot
Show:

Top
Show top of display.

Bot
Show end of display.

line_number
Display the specified line number as the first line of the display. You can specify any number between 1 and 99999999. You can omit leading zeros.

Head[,OFF|,ON]
Switch the standard header on (ON) or off (OFF) on the display (see “Scrolling displays” on page 58). Enter Head with no other parameters to use the default which is specified by the application (see the description of the DISPC macro in *ALCS Application Programming Guide* for details).

Down[,*number_of_lines*]|(Up[,*number_of_lines*]
Change the data displayed on the screen by moving down or up by the number of lines specified in the *number_of_lines* parameter. If you do not specify the number of lines, the display moves down or up by one screen.

Left[,*number_of_columns*]|(Right[,*number_of_columns*]
Change the data displayed on the screen by moving left or right by the number of columns specified in the *number_of_columns* parameter. If you do not specify the number of columns, the display moves left or right by one screen.

Find|Scan,(*character_string*)
Search for the data specified in *character_string*. You can specify up to 50 characters and can omit the parentheses if the string does not contain any blanks.

Cols,{*n*|RESET}
Where:

n
Restrict the number of columns displayed to *n*. The minimum number of columns you can specify is 2. When you have set the number of columns to *n*, you can enter `Right` or `Left` to scroll through the display by this number of columns.

RESET
Reset the display to its original size.

LOG{,Start[,*maxlines*]|,sTop}
Controls scroll log mode.

Start[,*maxlines*]
Start scroll log mode. *maxlines* is the number of lines which will be saved in the scroll log. This is a decimal number of 1 to 8 digits. The default and maximum for this value is specified at ALCS generation time by the SCRLLOG parameter of the SCTGEN macro. This is described in *ALCS Installation and Customization*

Note: The ZSCRL LOG command captures only the first screen of a scrollable display. Scrolling moves forward and back in the log but not in the previous display. This is done because the previous display can be very large.

sTop
Stop scroll log mode.

Prev
Show the previous file (if any).

Next
Show the next file (if any).

PRINT[,crI=*cri*|,crN=*crn*]
Print the active display on the printer specified. If you do not specify a printer, the active display is printed on the associated printer, if one is defined.

Normal response

See "Scrolling displays" on page 58 for some sample responses.

ZSNDU -- Send or receive an unsolicited message

Use the ZSNDU command -- ***from any terminal*** -- to send a message to another display or printer, or to display the next unsolicited message directed to this terminal.

Use the ZSNDU command -- ***from a Prime CRAS authorized terminal only*** -- to send a broadcast message (a message directed to more than one terminal) or to purge one or more broadcast messages.

Installation-wide exit ACM0 can be modified to allow other terminals to send a broadcast message. *ALCS Installation and Customization* describes the ACM0 installation-wide exit.

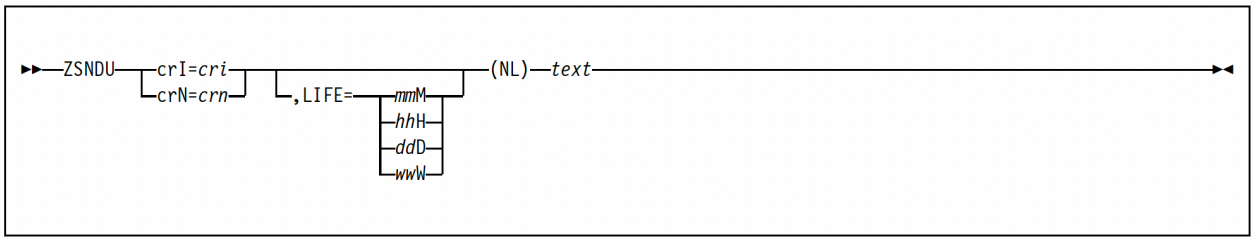
Use the ZSNDU command -- ***from any CRAS authorized terminal*** -- to see all live (unexpired) broadcast messages.

Installation-wide exit ACM0 can be modified to allow other terminals to see all broadcast messages. *ALCS Installation and Customization* describes the ACM0 installation-wide exit.

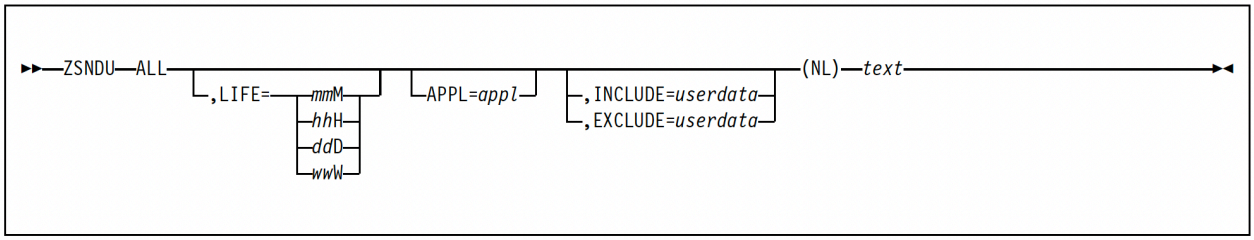
This command invokes the ALCS scrolling function. See "Scrolling displays" on page 58 for details.

The format of the command is:

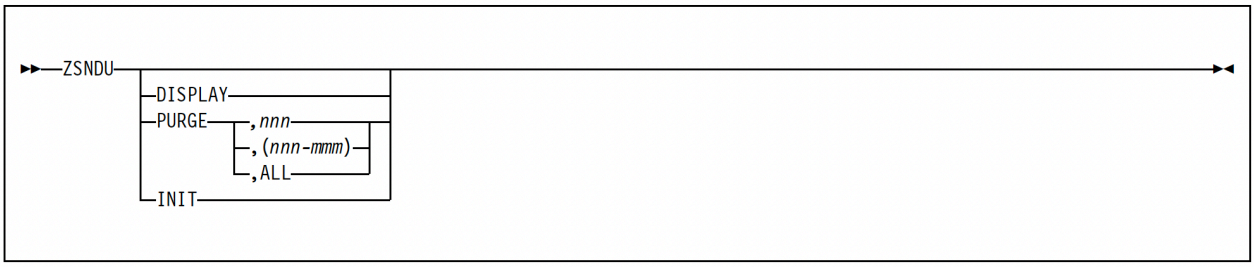
To send a message



To broadcast a message



To display or purge a message



Where:

ZSNDU with no parameters displays the next outstanding unsolicited message (if any) for the terminal from which the command is entered.

crI=*cri*
CRI of the resource to which the message is sent (6 hexadecimal digits).

crN=*crn*
CRN of the resource to which the message is sent (1 to 8 alphanumeric characters).

You can specify a range of terminals as shown in the following examples.

crN=*abcd**
Sends a message to all terminals with a CRN starting with the (1 to 7) character string *abcd*.

crN=abcd***
Sends a message to all terminals with a CRN ending with the (1 to 7) character string *abcd*.

crN=abcd****
Sends a message to all terminals with a CRN containing the (1 to 6) character string *abcd*.

Each of the above creates a broadcast message.

Note: In the syntax diagram we show the **crI**=*cri* and **crN**=*crn* separately from the ALL parameter because you can use them to send to a specific CRN or CRI.

LIFE={*mm*|M|*hh*|H|*dd*|D|*ww*W}
Lifetime of the message. After this time the message becomes unavailable for display. You can specify the lifetime in minutes (*mm***M**), hours (*hh***H**), days (*dd***D**), weeks (*ww***W**).

Where:

mm
A decimal number from 1 to 99.

hh
A decimal number from 1 to 99.

dd
A decimal number from 1 to 28.

ww
A decimal number from 1 to 4.

Note: The default ALCS value is 3 hours; this value can be altered during installation.

text
On the ***next line*** (indicated by NL in the syntax diagram), is the text of the message you want to send. The message can contain significant newline characters.

ALL
Send a broadcast message to all terminals.

APPL=*appl*
Send the broadcast message only to terminals routed to the specified application name. For example, the message could be sent to all terminals routed to a departure control application.

INCLUDE={*userdata1*,[*userdata2*,...]}
Send the broadcast message only to terminals matching criteria specified in the installation-wide exit routine (or routines) related to the specified user parameters *userdata1*, *userdata2*, and so on. Your system programmer can tell you which names are allowed in your installation. Userdata items have a maximum length of 8 characters. See *ALCS Installation and Customization* for a full explanation of the installation-wide exit AUM1.

EXCLUDE={*userdata1*,[*userdata2*,...]}
Send the broadcast message only to terminals not matching criteria specified in the installation-wide exit routine (or routines) related to the specified user parameters *userdata1*, *userdata2*, and so on. Your system programmer can tell you which names are allowed in your installation. Userdata items have a maximum length of 8 characters. See *ALCS Installation and Customization* for a full explanation of the installation-wide exit AUM1.

DISPLAY
Display details of all live (unexpired) broadcast messages. (See the normal response To a request to display all broadcast messages.)

PURGE,{*nnn*|(*nnn-mmm*)|ALL}
Purge one or more broadcast messages:

nnn
Purge message number *nnn*.

nnn-mmm
Purge message number *nnn* through message number *mmm*.

ALL
Purge all unsolicited messages.

Message numbers are 10 digits long but you need not key in leading zeros.

You can use ZSNDU DISPLAY to see the message numbers.

INIT
Purge all unsolicited messages and reset the message number counter. The next broadcast message is number 1.

Restrictions

1. You must use a Prime CRAS authorized terminal when you want to send a message to more than one terminal (a broadcast message) or issue ZSNDU with the PURGE or INIT parameters.
2. When you issue the ZSNDU command with the INIT parameter, ALCS requires you to confirm the command (see "Confirmation of commands" on page 60).

Normal responses

To a request to display a message

(ZSNDU command entered without parameters)

For a single terminal:

```
DXC8436I CMD i hh.mm.ss SNDU
Unsolicited message from CRN-crn CRI-cri
Creation date dd/mm/yy Time hh.mm.ss
message text
```

For a broadcast message:

```
DXC8436I CMD i hh.mm.ss SNDU
Unsolicited Broadcast Message
Creation date dd/mm/yy Time hh.mm.ss.
message text
```

or, if there are no messages waiting:

```
DXC8057I CMD i hh.mm.ss SNDU  No message on file
```

To a request to send a message

(ZSNDU command entered with parameters)

```
DXC8028I CMD i hh.mm.ss SNDU  OK
```

on the originating terminal.

```
DXC8436I CMD i hh.mm.ss SNDU
Unsolicited message from CRN-crn CRI-cri
Creation date dd/mm/yy Time hh.mm.ss
message text
```

on the destination (if it is a printer).

To a request to display all broadcast messages

(ZSNDU command with the DISPLAY parameter)

```
DXC8437I CMD i hh.mm.ss SNDU
Msg-number  Appl      Lifetime       Parameters       Text
  n         appl      ww.dd.hh.mm    Crn:crn          text
  n                   ww.dd.hh.mm    User:parameters
  n                   ww.dd.hh.mm
```

Where:

n
The sequence number of the broadcast message.

appl
The application name, if one was specified in the ZSNDU broadcast command.

ww.dd.hh.mm
The time remaining (in weeks, days, hours and minutes) before the message expires.

parameters
When *userdata1, userdata2, ...* parameters are specified with the **INCLUDE**= or **EXCLUDE**= parameters in the broadcast command, the display contains the word "User:", followed by I (Include=) or E (Exclude=) followed by the actual values of the *userdata1, userdata2, ...* parameters.

text
The first 40 characters of the broadcast message text. (You may need to scroll right to see the full text.)

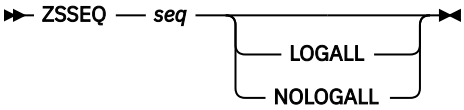
crn
CRN of the resource to which the message is sent (one to eight alphanumeric characters).

ZSSEQ -- Switch sequential file

Use the ZSSEQ command -- ***from any CRAS authorized terminal*** -- to switch real-time or system sequential file output from one data set to another. ZSSEQ closes and deallocates the old (switch from) data set so that another job can process it. This function is not available for general sequential files.

ZSSEQ may also be used turn ON or OFF the logging of all record updates.

The format of the command is:



Where:

seq
Symbolic name of the sequential file.

LOGALL
Start to log ALL record updates on the new (switch to) sequential file. May only be used when *seq* is LOG.

NOLOGALL
Resume normal logging on the new (switch to) sequential file. May only be used when *seq* is LOG.

Normal responses

```
DXC8655I CMD i hh.mm.ss SSEQ
Sequential file seq switch request accepted
```

When the switch completes, the following message is sent to the RO CRAS:

```
DXC2651I SEQ i  hh.mm.ss Sequential file seq switch complete
From Dsname=data set name
Volume=volser
```

Use ZDSEQ to display information about the new (switch to) data set. This display may not include a new standby data set. This is because ALCS does not open the standby data set until I/O is initiated on the new (switch to) data set.

ALCS also displays this message on the RO CRAS when an automatic sequential file switch completes. An automatic file switch happens when ALCS writes more than a specified number of blocks to a sequential file data set. The number of blocks is specified in the ALCS sequential file generation.

When the LOGALL option is specified the following is sent to the RO CRAS.

```
DXC2653I SEQ i  hh.mm.ss
LOGALL now active, all database updates are being logged
```

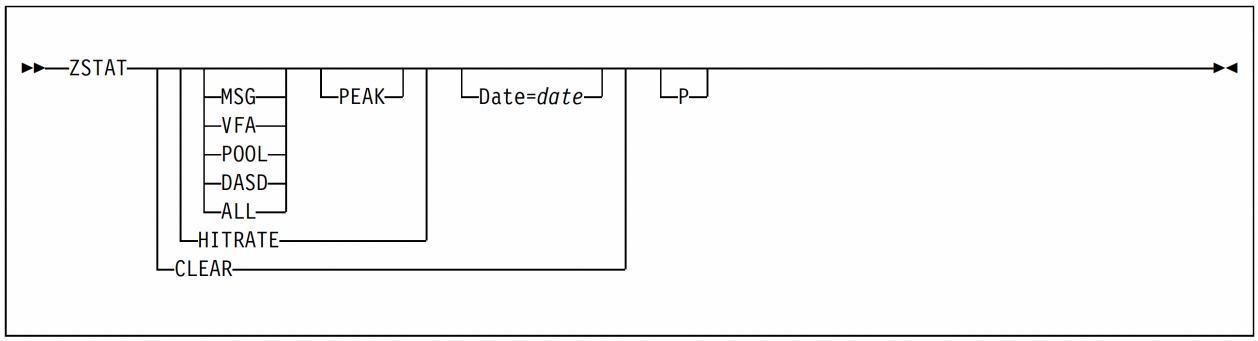
When the NOLOGALL option is specified the following is sent to the RO CRAS.

```
DXC2654I SEQ i  hh.mm.ss
LOGALL inactive, database logging is normal
```

ZSTAT -- Display current system load

Use the ZSTAT command -- ***from any terminal*** -- to display current and mean (average) or peak system activity.

The format of the command is:



where:

MSG
Display input and output message values.

VFA
Display VFA values.

POOL
Display pool values.

DASD
Display DASD values.

ALL
Display all values.

HITRATE
Display VFA hit rate values.

PEAK
Display peak values (default is mean).

Date=*date*
Date in the format *hhddmmyy*, or *ddmmyy*, or *mmyy*. Display system activity for the specified hour, day, or month. `Date` is only valid if the ALCS performance monitor is enabled.

CLEAR
Clear previously displayed peak values, so as to record only future peak values.

P
Print the message on the associated printer.

Normal responses

To a ZSTAT or ZSTAT P request:

```
DXC8590I CMD i hh.mm.ss STAT
Current Active Entries.........................ac
Current Delay / Defer Entries..................dc
Mean Entries...................................ee
Mean Message Entries ..........................em
Mean Input messages per second ................im
Mean Output messages per second ...............pm
Mean Available Storage units ..................sm
Mean Available Type2 Storage units ............tm
Mean Available I/O control blocks .............cm
Mean DASD I/O per second ......................om
Mean DASD reads per second ....................or
Mean DASD writes per second ...................ow
Mean VFA accesses per second ..................va
Mean VFA FINDs per second .....................vf
Mean VFA FILEs per second .....................vg
Mean Ln accesses per second ...................la
Mean Ln FINDs per second ......................lf
Mean Ln FILEs per second ......................lg
Mean LnST dispenses per second ................ns
Mean LnLT dispenses per second ................nl
Mean LnST directory use per second ............nsu
Mean LnLT directory use per second ............nlu
```

ALCS uses exponential smoothing to calculate the average values. You can interpret this as the average over the last 16 seconds.

To a ZSTAT PEAK or ZSTAT PEAK P request:

```
DXC8590I CMD i hh.mm.ss STAT
Current Active Entries.........................ac
Current Delay / Defer Entries .................dc
Peak Entries ..................................ee pt
Peak Message Entries ..........................em pt
Peak Input messages per second ................im pt
Peak Output messages per second ...............pm pt
Min  Available Storage units...................sm pt
Min  Available Type2 Storage units ............tm pt
Min  Available I/O control blocks .............cm pt
Peak DASD I/O per second ......................om pt
Peak DASD reads per second ....................or pt
Peak DASD writes per second ...................ow pt
Peak VFA accesses per second ..................va pt
Peak VFA FINDs per second .....................vf pt
Peak VFA FILEs per second .....................vg pt
Peak Ln accesses per second ...................la pt
Peak Ln FINDs per second ......................lf pt
Peak Ln FILEs per second ......................lg pt
Peak LnST dispenses per second ................ns pt
Peak LnLT dispenses per second ................nl pt
Peak LnST directory use per second ............nsu pt
Peak LnLT directory use per second ............nlu pt
```

To a ZSTAT ALL or ZSTAT ALL P request:

```
DXC8590I CMD i hh.mm.ss STAT
Current Active Entries.........................ac
Current Delay / Defer Entries .................dc
Mean Entries...................................ee
Mean Message entries ..........................em
Mean Input messages per second ................im
Mean Output messages per second ...............pm
Mean Available Storage units ..................sm
Mean Available Type2 Storage units.............tm
Mean Available Type3 Storage units.............tm
Mean Available I/O control blocks .............cm
Mean DASD I/O per second ......................om
Mean DASD reads per second ....................or
Mean DASD writes per second ...................ow
Mean VFA accesses per second ..................va
Mean VFA FINDs per second .....................vf
Mean VFA FILEs per second .....................vg
Mean Ln VFA DASD I/O per second ...............lm
Mean Ln VFA READs per second ..................lr
Mean Ln VFA WRITEs per second .................lw
Mean Ln accesses per second ...................la
Mean Ln FINDs per second ......................lf hr
Mean Ln FILEs per second ......................lg
Mean LnST dispenses per second ................ns
Mean LnLT dispenses per second ................nl
Mean LnST directory use per second ............nsu
Mean LnLT directory use per second ............nlu
Mean 3270 input msg per second ................im
Mean ALCI input msg per second ................im
Mean WTTY input msg per second ................im
Mean MQ input msg per second ..................im
Mean TCP/IP input msg per second ..............im
Mean APPC input msg per second ................im
Mean LU 6.1 input msg per second ..............im
Mean X.25 input msg per second ................im
Mean SLC input msg per second .................im
Mean BSC input msg per second .................im
Mean NETVIEW input msg per second .............im
Mean OSYS input msg per second ................im
Mean WAS input msg per second .................im
Mean 3270 output msg per second ...............pm
Mean ALCI output msg per second ...............pm
Mean WTTY output msg per second ...............pm
Mean MQ output msg per second .................pm
Mean TCP/IP output msg per second .............pm
Mean APPC output msg per second ...............pm
Mean LU 6.1 output msg per second .............pm
Mean X.25 output msg per second ...............pm
Mean SLC output msg per second ................pm
Mean BSC output msg per second ................pm
Mean NETVIEW output msg per second ............pm
Mean OSYS output msg per second ...............pm
Mean WAS output msg per second ................pm
```

To a ZSTAT MSG or ZSTAT MSG P request:

```
DXC8590I CMD i hh.mm.ss STAT
Mean Input messages per second .................im
Mean Output messages per second ................pm
Mean 3270 input msg per second .................im
Mean ALCI input msg per second .................im
Mean WTTY input msg per second .................im
Mean MQ input msg per second ...................im
Mean TCP/IP input msg per second ................im
Mean APPC input msg per second .................im
Mean LU 6.1 input msg per second ...............im
Mean X.25 input msg per second .................im
Mean SLC input msg per second ..................im
Mean BSC input msg per second ..................im
Mean NETVIEW input msg per second ..............im
Mean OSYS input msg per second .................im
Mean WAS input msg per second ..................im
Mean 3270 output msg per second ................pm
Mean ALCI output msg per second ................pm
Mean WTTY output msg per second ................pm
Mean MQ output msg per second ..................pm
Mean TCP/IP output msg per second ..............pm
Mean APPC output msg per second ................pm
Mean LU 6.1 output msg per second ..............pm
Mean X.25 output msg per second ................pm
Mean SLC output msg per second .................pm
Mean BSC output msg per second .................pm
Mean NETVIEW output msg per second .............pm
Mean OSYS output msg per second ................pm
Mean WAS output msg per second .................pm
```

To a ZSTAT VFA or ZSTAT VFA P request:

```
DXC8590I CMD i hh.mm.ss STAT
Mean DASD I/O per second .......................om
Mean DASD reads per second .....................or
Mean DASD writes per second ....................ow
Mean VFA accesses per second ...................va
Mean VFA FINDs per second ......................vf
Mean VFA FILEs per second ......................vg
Mean Ln accesses per second ....................la
Mean Ln FINDs per second .......................lf
Mean Ln FILEs per second .......................lg
```

To a ZSTAT DASD or ZSTAT DASD P request:

```
DXC8590I CMD i hh.mm.ss STAT
Mean DASD I/O per second .......................om
Mean DASD reads per second .....................or
Mean DASD writes per second ....................ow
Mean VFA accesses per second ...................va
Mean VFA FINDs per second ......................vf
Mean VFA FILEs per second ......................vg
Mean Ln accesses per second ....................lm
Mean Ln FINDs per second .......................lr
Mean Ln FILEs per second .......................lw
```

To a ZSTAT POOL or ZSTAT POOL P request:

```
DXC8590I CMD i hh.mm.ss STAT
Mean LnST dispenses per second ..............  ..ns
Mean LnLT dispenses per second ..............  ..nl
Mean LnST directory use per second .............nsu
Mean LnLT directory use per second .............nlu
```

To a ZSTAT HITRATE or ZSTAT HITRATE P request

```
DXC8590I CMD i hh.mm.ss STAT
Mean Ln VFA READs per second ................. ..lr
Mean Ln FINDs per second ..................... ..lf hr
```

Using parameter MSG, VFA, POOL, DASD, HITRATE, or ALL with the PEAK parameter gives displays similar to those already illustrated, except that the word Peak or Min replaces the word Mean.

Where:

Mean
ALCS uses exponential smoothing to calculate the average values. You can interpret this as the average over the last 16 seconds.

Peak
ALCS provides the highest value in any 200 millisecond period expressed as a per second value.

Min
ALCS provides the lowest value in any 200 millisecond period expressed as a per second value.

ac
The instantaneous number of active entries in the system. This includes the ZSTAT message processor. If there is no other activity in the system then *ac*=1 and *dc*=0.

dc
The instantaneous number of delay/defer entries in the system.

ee
Number of entries in the system.

em
Number of entries in the system not performing internal maintenance work (that is, created as a result of input messages).

im
Number of input messages per second. For APPC, this includes input messages identified by installation-wide monitor exit USRAPPC. For TCP/IP, this includes input messages identified by installation-wide monitor exit USRTCP1. For MQ, this includes input messages identified by installation-wide monitor exit USRMQI1. For WAS, this includes input messages identified by installation-wide monitor exit USRWAS1.

pm
Number of output messages per second. For APPC, this includes output messages identified by installation-wide monitor exit USRAPPC. For TCP/IP, this includes output messages identified by installation-wide monitor exit USRTCP1. For MQ, this includes output messages identified by installation-wide monitor exit USRMQI1. For WAS, this includes output messages identified by installation-wide monitor exit USRWAS1.

sm
Number of available storage units.

tm
Number of available high-level-language storage units.

cm
Number of available input/output control blocks.

om
Number of DASD I/O operations per second.

or
Number of DASD read operations per second.

ow
Number of DASD write operations per second.

va
Number of VFA accesses per second.

vf
Number of VFA FINDs per second.

vg
Number of VFA FILEs per second.

la
Number of VFA accesses per second for L*n* records.

lf
Number of VFA FINDs per second for L*n* records.

lg
Number of VFA FILEs per second for L*n* records.

lm
Number of VFA I/O operations per second for L*n* records.

lr
Number of VFA read operations per second for L*n* records.

lw
Number of VFA write operations per second for L*n* records.

hr
VFA hit rate percentage for Ln records.

ns
Number of L*n* short-term dispenses per second.

nl
Number of L*n* long-term dispenses per second.

nsu
Number of L*n* short-term dispenses per second plus the number of L*n* short-term records found in use per second. The ratio of *ns* to *nsu* gives an estimate of pool availability.

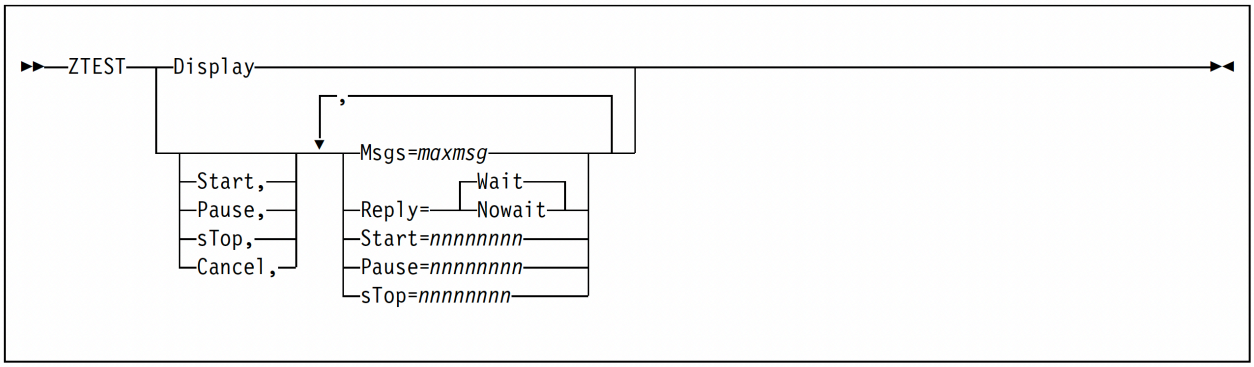
nlu
Number of L*n* long-term dispenses per second plus the number of L*n* long-term records found in use per second. The ratio of *nl* to *nlu* gives an estimate of pool availability.

pt
The time the last peak or minimum occurred.

ZTEST -- Control system test vehicle

Use the ZTEST command -- ***from a Prime CRAS authorized terminal only*** -- to control the system test vehicle (STV). STV operates only in normal (NORM) state. For performance reasons, do not operate STV during production operation. You can also use the ZTEST command -- ***from any CRAS authorized terminal*** -- to display the current status of STV.

The format of the command is:



Where:

Display
Display and print the current status of STV on the associated printer.

This is a "dummy" option: ZTEST always displays and prints the current status.

Start
Start or restart STV. STV reads input messages from the sequential file with symbolic name TUT.

Pause
Pause STV processing.

sTop
Stop STV processing.

Cancel
Cancel STV. Use ZTEST `Cancel` only if ZTEST `sTop` does not stop STV.

Msgs=*maxmsg*
The maximum number of messages that STV can initiate concurrently. When STV processes messages from the TUT file it attempts to maintain this level. The default is 1. An input STV message is included in the count of STV messages in progress until the ECB processing has terminated.

Reply=[Wait|Nowait]
Where:

Nowait
STV introduces another message into the system from a specific CRI if both of the following conditions are satisfied:

- The system is not too busy.
- The maximum number of STV messages in progress has not been exceeded.

Wait
STV introduces another message into the system from a specific CRI only if, in addition to the conditions above:

- The display has been replied to, or
- The ECB processing the previous input STV message has terminated.

Usage notes

When you specify **Reply**=Wait (this is the default), STV simulates a real device. This means that every message entered must receive a reply before another message can be entered.

When you specify **Reply**=Nowait, STV does not simulate a real device. The only limits on entering new messages into the system are the system load and the number of messages that STV is allowed to have in progress at any one time. The application that is processing the input messages must be capable of processing many simultaneous input messages from the same resource.

Note: If only one resource is used to enter messages into the system by STV, then setting **Reply**=Nowait can give a much higher throughput than **Reply**=Wait.

TPF-compatible keywords

The following keywords are included for compatibility with the TPF ZSTVS command, but are of little value to ALCS users, who can achieve the same results more easily by means of normal MVS facilities. IBM does not commit to support these keywords in any future release of ALCS.

Start=*nnnnnnnn*
Set the TUT block number at which STV begins to enter messages into the ALCS system. The default is 1.

Pause=*nnnnnnnn*
Set the TUT block number at which STV is to pause. The default is to process the whole data set.

sTop=*nnnnnnnn*
Set the TUT block number at which STV is to stop. The default is the whole data set.

Start=, **sTop=**, **Pause=**, and **Msgs=** can be specified at any time, before or during the STV run. They can be specified together on one ZTEST command. A comma or space must separate each parameter.

For further information on STV see “ALCS test facilities” on page 386.

For further information on using the STC to create test unit data sets (TUT files) see *ALCS Installation and Customization*.

Note to TPF users:

For the ALCS equivalent of the TPF ZTEST command, see “ZDRIV -- Activate application program” on page 193.

Normal responses

```
DXC8452E CMD i hh.mm.ss TEST  Not active
Start--ss, Stop--tt, Pause--pp, Reply--ww
Maximum messages--n3, Current messages--n4


DXC8453I CMD i hh.mm.ss TEST  Processing block n1
Start--ss, Stop--tt, Pause--pp, Reply--ww
Maximum messages--n3, Current messages--n4


DXC8454I CMD i hh.mm.ss TEST  Paused at block n2
Start--ss, Stop--tt, Pause--pp, Reply--ww
Maximum messages--n3, Current messages--n4
```

Where:

n1
Block number currently being processed by STV.

n2
Block number at which STV is paused.

n3
Number specified by the `Msgs=` parameter on the ZTEST command.

n4
Number of messages that STV has entered into the ALCS system that have not yet been completely processed.

ss
Number specified by the `Start=` parameter on ZTEST.

tt
Number specified by the `Stop=` parameter on ZTEST or NO.

pp
Number specified by the `Pause=` parameter on ZTEST or NO.

ww
Value specified on the `Reply=` parameter on ZTEST (WAIT or NOWAIT).

Chapter 7. Using the ALCS trace facility

ALCS provides a trace facility to help application programmers debug their programs. You can run the ALCS trace facility either on a dedicated test system or on a production system. Use it with care on a production system, because it degrades performance.

Application programs use ***monitor-request macros*** to request services from the ALCS online monitor. The ALCS trace facility monitors the execution of these macros and interrupts processing at a specified place to log selected data. See *ALCS Application Programming Reference - Assembler* for a full description of all ALCS macros, and their group names.

You can trace all monitor-request macros, or you can specify trace control options to restrict tracing to:

- Selected entries
- Selected application programs
- Selected monitor-request macros
- Selected DASD records

You can use the ALCS trace facility to initiate source level debugging of C/C++ programs on a remote workstation. This facility is called "workstation trace".

ALCS also provides the ability to trace input and output messages to an online trace area for selected communication resources. You can run this facility either on a dedicated test system or on a production system.

Destinations

You can direct the trace output to the:

- System macro trace block.
- ALCS diagnostic file.
- Terminal (this is called ***conversational tracing***)
- Wrap around online trace area (for ALCS message trace only).

The destinations provide different levels of tracing, as described next.

The system macro trace block

When operating normally, ALCS records information about each monitor-request macro call in the ***entry macro trace block***. This information relates to a specific entry. When a system error occurs, the system error dump includes the contents of the entry macro trace block. This tells you about the monitor-request macros issued by the entry before the system error.

If you start tracing to the ***system macro trace block***, the contents of this block are also included in any system error dump. The system macro trace block includes information about monitor-request macros issued by ***all*** entries in the system.

A dump can occur when two or more entries fail to work cooperatively. You may need a single trace block showing the macros issued by all the entries. This shows the relative sequence of actions for the entries.

The maximum number of monitor-request macros that can be recorded in the system macro trace block is specified by the system programmer during ALCS system generation.

You can examine the contents of the system macro trace block from your terminal (see “Tracing to the system macro trace block” on page 307).

The ALCS diagnostic file

You can obtain more detailed information if you start tracing to the ALCS diagnostic file. When you use this type of trace, you can specify what information is to be recorded and what is not.

After tracing, you can run the ALCS diagnostic file processor which produces a report containing the following information:

- ***Start-of-trace and end-of-trace identifier***

  This is a single line showing where the trace starts and stops.

- ***Macro information***

  The report contains a line for each monitor-request macro showing:

  – Address register of the ECB.
  – Program name, or question marks (?) when the online monitor issued the monitor-request macro.
  – Listing address of the monitor-request macro in the program, or question marks (?) when the monitor-request macro is not in the application program.
  – Name of the monitor-request macro.
  – Operands of the monitor-request macro (in hexadecimal).
  – CRI of the terminal that created the entry. For WTTY lines, the line number is the low-order byte of the CRI.

- ***Registers***

  The contents of the general and floating point registers as they are immediately before the monitor-request macro executes (if requested).

- ***Entry control block and attached storage blocks***

  The ALCS diagnostic file processor prints this data in the same format as a system error dump. The selection of the data depends on the options you specify when starting the trace.

Conversational tracing

Conversational tracing provides you with the most flexibility in selecting and displaying information. See "Conversational tracing" on page 313.

Online message trace

You can start, stop, clear, or display an online message trace with a minimal overhead for a maximum of 8 (either generic or non-generic) communication resources concurrently.

The tracing takes place

1. Just before control is passed to an application or to the ALCS command processor, and
2. At the start of ROUTC or SEND-type monitor-request macro processing.

IBM provides you with callable services UMSGT1 and UMSGT2 in order to perform any additional tracing.

With help of two installation-wide ECB controlled exits the online trace message display can be customized.

- See *ALCS Installation and Customization* for details of the UMSGT1 and UMSGT2 callable services.
- See "Starting, Stopping, Clearing, and Displaying an online message trace" on page 344 for more information on using online message trace.

Tracing to the system macro trace block

You can control block tracing -- ***from a Prime CRAS authorized terminal***. However the `Display` parameter can be used from any CRAS terminal.

The format of the command is:

You can examine the contents of the system macro trace block -- ***from any CRAS authorized terminal***.

The format of the command is:

Where:

Start
Start tracing to the system macro trace block.

sTop
Stop tracing.

Display
Display the contents of the system macro trace block on the terminal.

Use `Display` with no other parameters to show whether trace is active or not.

count
Display the last *count* entries from the system macro trace block, where *count* is a decimal number. If *count* is omitted or zero (the default), the response shows only whether trace is active or inactive.

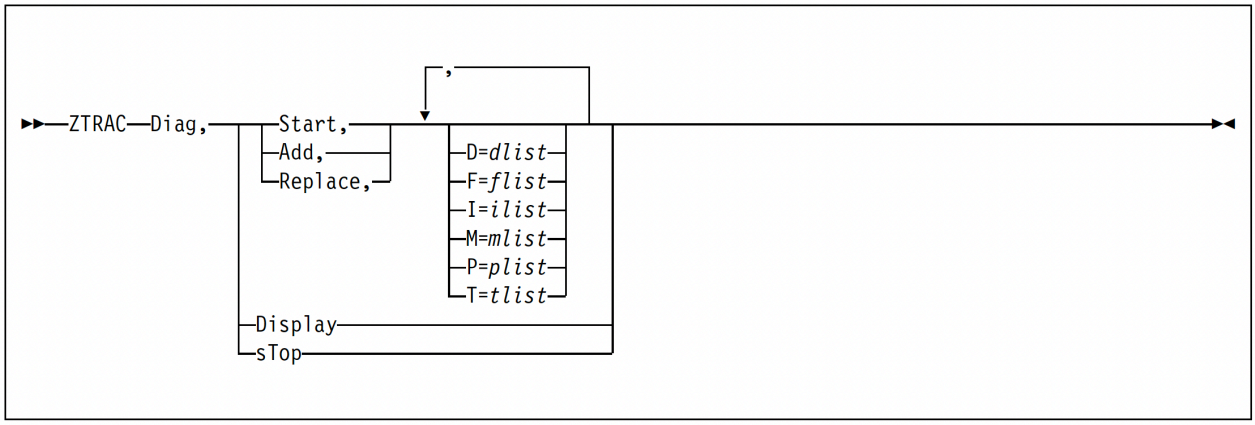
All
Display all entries, from the system macro trace block.

This command invokes the ALCS scrolling function. You can print the display by using the `ZSCRL PRINT` command. See "ZSCRL -- Scroll information on the screen" on page 289.

Tracing to the ALCS diagnostic file

Control tracing to the ALCS diagnostic file -- ***from a Prime CRAS authorized terminal***. You can run the ALCS diagnostic file processor later as a batch job. However the `Display` parameter can be used from any CRAS terminal.

The format of the command is:



Where:

Start
Start tracing to the ALCS diagnostic file, with or without trace control parameters.

Add
Add to trace control parameters.

Replace
Replace the specified trace control parameters with the new specifications.

Display
Show whether trace is active or not, and which trace control options are in effect.

sTop
Stop tracing.

Trace control parameters

You can restrict the scope of tracing by specifying trace control parameters, and you can change these parameters during tracing.

The parameters are successively restrictive. For example:

ZTRAC DIAG,START,P=(AAAA,BBBB)
Traces all monitor-request macros in programs AAAA and BBBB.

ZTRAC DIAG,START,P=(AAAA,BBBB),M=(FIND)
Traces only find-type monitor-request macros in programs AAAA and BBBB. It does not trace find-type monitor-request macros in other programs.

Incompatible trace control parameters

Do not specify either F= or I= with M=.

Multiple values for control parameters

Each control parameter can have a list of values enclosed in parentheses and separated by commas. If only one value is specified, the parentheses can be omitted. For example, **D**=*dlist* can be:

```
D=(EBW,EBX)
```

or

```
D=EBW
```

The trace control parameters are as follows

D=*dlist*
At each trace intercept, write the specified data items to the ALCS diagnostic file, where *dlist* can be one or more of the values in Table 22 on page 309.

Table 22. Tracing to the ALCS diagnostic file: The D= parameter

dlist	**Data written to the ALCS diagnostic file**	**Synonyms**
ALL	Whole ECB (including the TPFDF stack, detached blocks, and in-use DECBs)	AW EA
EBL	ECB local program work area	CE1WKC
EBW	ECB work area 1	CE1WKA EW
EBX	ECB work area 2	CE1WKB EX

Table 22. Tracing to the ALCS diagnostic file: The D= parameter (continued)		
dlist	**Data written to the ALCS diagnostic file**	**Synonyms**
EBD	ECB descriptor	
EBP	ECB prefix	
DECB	All DECB descriptors and application areas	
LEVELs	ECB storage levels and data levels	LEVs EL
DLEVELS	Storage levels and data levels for in-use DECBs	DLEVs DL
ER	Relevant parts of ECB -- those parts that the monitor-request macro references	
GPRs	General registers	Gr REGs R
FPRs	Floating-point registers	FP Fr
RS	64 bytes of storage addressed by each of the general registers	
RSA	4096 bytes of storage addressed by each of the general registers	
SA	All attached storage blocks	
SR	Relevant attached storage blocks -- those blocks that the monitor-request macro references	
MAXimum	Maximum display (includes all the above, apart from RSA)	MX

Omit D= to write minimum data to the ALCS diagnostic file giving the ECB address, the program name and listing address, and the macro and macro parameters.

F=*flist*
Trace up to 8 file addresses. *flist* is the file address (8 hexadecimal digits), or the record type and record ordinal number in the form:

```
type(ordinal)
```

Note that this restricts the trace to monitor-request macros that refer to file addresses; do not specify F= with M=.

I=*ilist*
Trace references to up to 8 record IDs. *ilist* is the record ID (4 hexadecimal digits or 2 alphanumeric characters). Note that this restricts the trace to monitor-request macros that refer to record IDs; do not specify I= with M=.

M=*mlist*
Trace up to 8 monitor-request macros and up to 32 macro groups. *mlist* is the macro request type or macro group name. The macro request type is usually the macro name, but see *ALCS Application Programming Reference - Assembler* for a list of the exceptions and a list of macro group names.

Omit M= on a start command, or specify M=ALL to trace all monitor-request macros.

P=*plist*
Trace up to 8 programs. *plist* can be:

- A 4-character program name, or
- A 1-, 2-, or 3-character string, to mean all programs whose names begin with these characters.

Prefix the string with - (minus) to ***exclude*** that program or programs.

For example, P=(A,-AB) traces all programs whose names begin with A, ***except*** those whose names begin with AB.

Omit P= on a start command, to trace all programs.

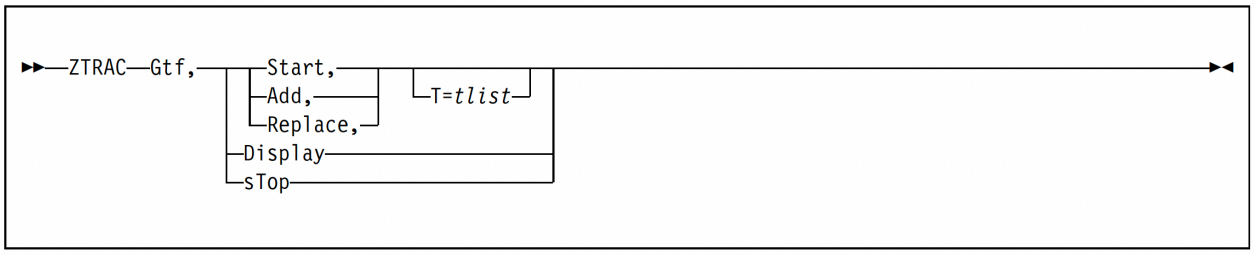
T=*tlist*
Trace messages from up to 8 terminals. *tlist* is the CRI or CRN of the selected terminal.

Omit T= on a start command, to trace all entries, including entries that are not messages.

Tracing to the MVS generalized trace facility

Control tracing of VTAM RPLs to the MVS generalized trace facility (GTF) -- ***from a Prime CRAS authorized terminal***. However the Display parameter can be used from any CRAS terminal.

The format of the command is:



Where:

Start
Start GTF tracing.

Add
Add to trace control parameters.

Replace
Replace the specified trace control parameters with the new specifications.

Display
Show whether GTF trace is active or not and which trace control options are in effect.

sTop
Stop GTF tracing.

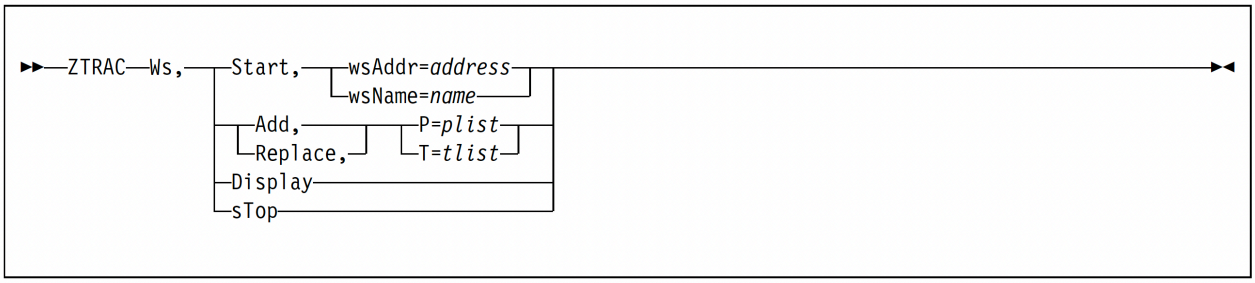
T=*tlist*
Generate GTF trace for activity related to up to 8 SNA communication resources. *tlist* is the CRI or CRN of the selected terminal. Specify T=000000 to trace SNA communication activity that is not specific to a particular resource. Omit T= on a start command to trace all SNA communication resources.

Workstation trace

Workstation trace can be controlled by entering a command - ***from a Prime CRAS authorized terminal only***. See “Conversational tracing” on page 313.

If workstation trace is active, you can run it to control the remote workstation debugger facility for C/C++ programs ***from any CRAS authorized display***.

The format of the command is:



Where:

Start
Start remote workstation debugger session.

Add
Add to trace control parameters.

Replace
Replace the specified trace control parameters with the new specifications.

P=*plist*
Trace up to 8 programs. *plist* can be:

- A 4-character program name, or
- A 1-, 2-, or 3-character string, to mean all programs whose names begin with these characters.

Prefix the string with - (minus) to ***exclude*** that program or programs.

For example, P=(A,-AB) traces all programs whose names begin with A, ***except*** those whose names begin with AB.

Omit P= on a start command, to trace all programs.

T=*tlist*
Trace messages from up to 8 terminals. *tlist* is the CRI or CRN of the selected terminal.

Omit T= on a start command, to trace all entries, including entries that are not messages.

Display
Show whether a remote workstation debugger session has been started for this terminal, and which trace control option is in effect.

sTop
Stop remote workstation debugger session.

wsAddr=*address*
address is the identity of the remote workstation, as a TCP/IP address in the dotted decimal format: *iii.iii.iii.iii*
Where *iii* is 1-3 decimal digits (0-255).

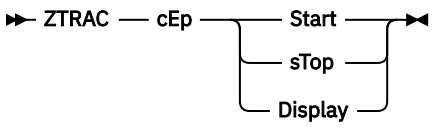
wsName=*name*
name is the identity of the remote workstation as a 1-60 character TCP/IP domain name.

TPFDF macro trace

Starting with TPFDF PUT 15 the tracing of TPFDF macro calls can be made active or inactive by entering a command -- ***from a Prime CRAS authorized terminal only.*** You can start and stop TPFDF macro trace, and obtain the status of TPFDF macro trace by entering a common entry point (CEP) command from any CRAS terminal.

The format of the command is:

OFF



Where:

cEp
The common entry point provides common processing for all TPFDF macro calls issued by ALCS application programs. It also provides trace facilities for TPFDF macro calls.

Start
Make TPFDF macro trace active. Provide TPFDF trace data in ALCS dumps (in the entry macro trace) and also provide it during conversational and diagnostic trace.

sTop
Make TPFDF macro trace inactive.

Display
Show whether TPFDF macro trace is active or inactive.

Note:

- See "DETAIL -- Display internal TPFDF macro calls" on page 326 for more information on controlling the level of internal TPFDF macro calls and other internal monitor-request macros displayed.
- The initial status of TPFDF macro trace is defined in the ALCS system generation. See *ALCS Installation and Customization* for an explanation of the SCTGEN TPFDF parameter.

TPFDF macro trace data

When TPFDF macro trace is active, ALCS obtains trace data for each TPFDF macro call issued by an application program. TPFDF trace data is provided in ALCS system error dumps (in the entry macro trace) and provided in conversational and diagnostic trace. Each TPFDF macro call is shown in the various traces as monitor-request macro TDFAC.

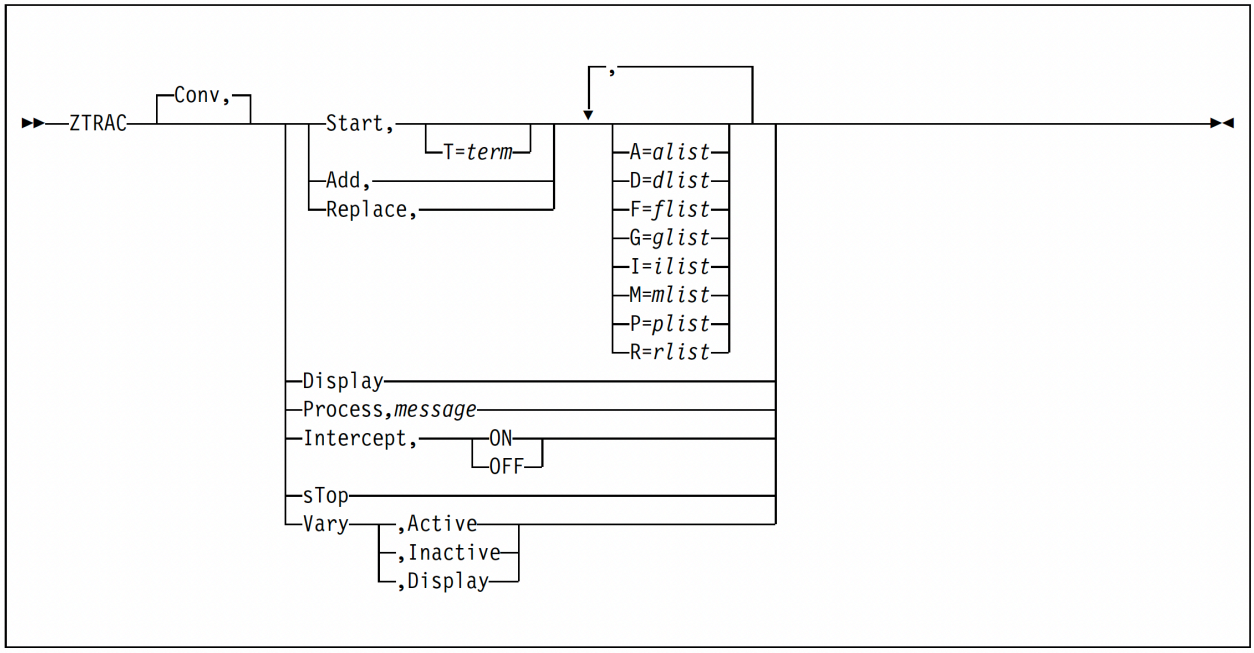
Conversational tracing

Conversational and workstation trace can be made active or inactive by entering a command -- ***from a Prime CRAS authorized terminal only.***

If conversational tracing is active, you can run it -- ***from any CRAS authorized display***.

If you are running conversational trace, you can start or stop asynchronous trace -- ***from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal only***.

The format of the command is:



Where:

Start
Start conversational tracing, with or without trace control options.

T=*term*
Trace input messages from a remote terminal. *term* is the CRI or CRN of the remote terminal.

Before you start tracing a remote terminal, be sure to check that the user at the remote terminal understands what you plan to do and agrees to it.

Note: *term* cannot be the CRN or CRI of the unique Prime CRAS.

Add
Add to trace control options.

Replace
Replace the specified trace control parameters with the new specifications.

Display
Show whether trace is active or not, and which trace control options are in effect.

Also show whether asynchronous trace is active or not.

Intercept,{ON|OFF}
Start (ON) or stop (OFF) asynchronous trace. At least one program must be included in your trace control options before starting asynchronous trace.

Note:

`Intercept,ON` and `Intercept,OFF` can only be entered from a Prime CRAS authorized terminal or an Alternate CRAS AT1 to AT16 authorized terminal.

Process,*message*
Process an input message or ALCS command without tracing it. *message* is the input message or ALCS command.

sTop
Stop trace.

Vary,{Active|Inactive|Display}
Vary conversational and workstation trace for the whole system. Where:

Active
Make conversational and workstation trace active so that any CRAS terminal can use them.

Inactive
Make conversational and workstation trace inactive so that no terminal can use them.

Display
Use this command from any CRAS display terminal to find out whether conversational and workstation trace are active or inactive.

Note: `Vary (Active)` and `Vary (Inactive)` can only be entered from a Prime CRAS authorized terminal.

Restrictions

Conversational tracing can trace entries that originate from terminal input messages. Asynchronous trace can also trace entries that are created by the system -- for example time-initiated processing, WTTY processing, and so on.

Many installations do not allow conversational tracing on production systems. If you cannot use conversational tracing, trace to the ALCS diagnostic file instead.

Conversational trace can be used to trace ALCS macros issued by C language applications through the C application programming interface routines. Instruction Stepping may not be used with C language application programs. If Instruction Stepping is active when a C language program is started, Instruction Stepping is turned off and turned back on when the program exits. See “STEP -- Control instruction stepping” on page 340.

Trace control parameters

You can restrict the scope of tracing by specifying trace control parameters, and you can change these parameters during tracing.

The parameters are successively restrictive. For example:

`ZTRAC CONV,START,P=(AAAA,BBBB)`
Traces all monitor-request macros in programs AAAA and BBBB.

`ZTRAC CONV,START,P=(AAAA,BBBB),M=(FIND)`
Traces only find-type monitor-request macros in programs AAAA and BBBB. It does not trace find-type monitor-request macros in other programs.

Incompatible trace control parameters

Do not specify either F= or `I=` with `M=`.

Multiple values for control parameters

Each control parameter can have a list of values enclosed in parentheses and separated by commas. If only one value is specified, the parentheses can be omitted. For example, **D**=*dlist* can be:

```
D=(EBW,ER,SA)
```

or

```
D=EBW
```

The trace control parameters are as follows:

A=*alist*
Set up to 8 address stops at specified program displacements. The format for each value in the A=*alist* can be either:

pname,*disp*
pname(*disp*)

where *pname* is the 4-character program name, and *disp* is the displacement in hexadecimal (as it appears in the program listing). The displacement can be any value in the range X'20' through X'FFFF'.

D=*dlist*
At each trace intercept, display the specified data on the terminal, where *dlist* is one or all of the values in Table 23 on page 316.

Table 23. Tracing to an ALCS terminal: The D= parameter

dlist	**Contents of display**	**Synonyms**
ALL[,W]	Full-screen display of registers, ECB storage and data levels, ECB work area 1 (CE1WKA), and so on	AW EA
ALL,X	Same as ALL,W but with ECB work area 2 (CE1WKB)	AX
ALL,L	Same as ALL,W but with ECB local program work area.	AL
EBL	ECB local program work area.	CE1WKC
EBW	ECB work area 1	CE1WKA EW
EBX	ECB work area 2	CE1WKB EX
LEVELs	ECB storage levels and data levels	LEVs EL
DLEVELs	Storage levels and data levels for in-use DECBs	DLEVs DL
SB*x*	Storage block attached at ECB level *x*. Specify SB0 for the block attached at level D0, and so on.	
AUT	Current automatic storage block.	
GPRs	General registers	Gr REGs R
FPRs	Floating-point registers	FP Fr

Omit D= for a minimum display giving the program name and listing address, and the macro or instruction.

Any unsupported trace display controls are ignored.

F=*flist*
Trace up to 8 file addresses. *flist* is the file address (8 hexadecimal digits), or the record type and record ordinal number in the form:

```
type(ordinal)
```

Note that this restricts the trace to monitor-request macros that refer to file addresses; do not specify F= with M=.

G=*glist*
Set up register stops for a specific register containing a specific value or range of values. This parameter sets up the register/value combinations to be used later (see “REGSTOP -- Register stop” on page 334 for more information about how they are used).

Up to 8 register/value combinations can be specified, and the format for each value in the *glist* can be either:

register(*value*)
register(*value1*-*value2*)

Where:

register
The general purpose register (decimal digits, 0-9, 14-15)

value
Stop when the register contains this value (maximum 8 hex digits)

value1-value2
Stop when the register contains a value within this range (maximum 8 hex digits, value2 > value1)

I=*ilist*
Trace references to up to 8 record IDs. *ilist* is the record ID (4 hexadecimal digits or 2 alphanumeric characters). Note that this restricts the trace to monitor-request macros that refer to record IDs; do not specify I= with M=.

M=*mlist*
Trace up to 8 specified monitor-request macros and up to 32 macro groups. *mlist* is the macro request type or macro group name. The macro request type is usually the macro name, but see *ALCS Application Programming Reference - Assembler* for a list of the exceptions and a list of macro group names.

For example:

ZTRAC CONV START M=(PROGRAM,FILE)
Start conversational trace, and trace all macros in groups PROGRAM and FILE.

ZTRAC CONV REPLACE M=(CREATE,FIND,SEND)
Replace the selection of macros to be traced by the macros in groups CREATE, FIND, and SEND.

ZTRAC CONV ADD M=(DEFRC,PROGRAM)
Add the macro DEFRC and the macros in group PROGRAM to the selection of macros already traced.

Omit M= or specify M=ALL to trace all monitor-request macros.

P=*plist*
Trace up to 8 programs. *plist* can be:

- A 4-character program name, or
- A 1-, 2-, or 3-character string, to mean all programs whose names begin with these characters.

Prefix the string with - (minus) to ***exclude*** that program or programs.

For example, P=(A,-AB) traces all programs whose names begin with A, ***except*** those whose names begin with AB.

Omit P= to trace all programs.

R=*rlist*
Trace references to an address (or range of addresses) in main storage. This parameter sets up the references to be used later (see "REFSTOP -- Reference stop" on page 333 for more information about how the references are used).

Up to 8 addresses can be specified, and the format for each value in the *rlist* is:

```
refstop(range[,S|R])
```

Where:

refstop
The hexadecimal storage address or the name of an ECB field, for example, EBW000.

range
The number of consecutive locations to be included in the address range. For example,

```
R=EBW000(3)
```

sets a check for any references to EBW000, EBW000+1, and EBW000+2. If *refstop* is the name of an ECB field, the default *range* is the length of the ECB field. Otherwise, the default *range* is one.

[S|R]
Specify S to set a stop when information is ***stored*** within the specified address range.

Specify R to set a stop when a ***reference*** is made within the specified address range.

Running a conversational trace

The normal response to ZTRAC CONV,START,... is :

```
TRACE -- ENTER THE MESSAGE TO TRACE
```

ALCS traces the next input message or ALCS command, except when it is:

Clear key
ALCS processes this without tracing it.

Null message
ALCS discards this without tracing it.

ZTRAC command
ALCS processes this without tracing it.

ZLOGF command
ALCS processes this without tracing it.

Note: The command ZTRAC PROCESS,*message* processes the message without tracing it. When you enter the input message for ALCS to trace, the ALCS trace displays a message to show that processing the new message is about to start. The message includes a header like this:

```
TRACE -- prog      NEW ENTRY
```

Where:

prog
The name of the first program that processes the new entry for the input message.

Depending on the display (**D**=) trace control option, there may be more information following the standard header.

You can now either:

- Press Enter to start processing.
- Enter a trace control command.

If you press Enter, then the processing of your input message proceeds normally until the entry executes a monitor-request macro. If your trace control options exclude that macro, processing proceeds until the entry executes a macro that is not excluded.

Processing then stops and the trace displays a message that shows the entry is paused, waiting to execute the macro. The message includes a header like this:

```
TRACE -- prog list macro operand
```

Where:

prog
Name of the program that contains the macro.

list
Listing address of the macro within the program.

macro
Request type. This is usually the macro name, but there are some exceptions (see *ALCS Application Programming Guide* for more information).

operand
Operand field of the macro. Note that this does not appear for all macros. Also the format is not necessarily identical to the program source code.

If the monitor-request macro is not within a program (for example, if it is in the ECB) then the standard header shows its 8-byte storage address instead of the program name and listing address.

Depending on the display (**D**=) trace control option, there may be more information following the standard header. You can now either:

- Press `Enter` to proceed to the next monitor-request macro.
- Enter a trace control command.

The following sections describe the ALCS trace control commands

"ADSTOP -- Address stop" on page 323
Continue processing the entry (ignoring other trace control options) until the entry reaches one of the selected address-stop locations.

"ANYSTOP -- Any stop" on page 324
Continue processing the entry (ignoring other trace control options) until the entry reaches one of the selected address-stop, reference-stop, or register-stop conditions.

"BRANCH -- Branch to address" on page 325
Continue processing the entry starting at a specified address.

"DETAIL -- Display internal TPFDF macro calls" on page 326
Display internal TPFDF macro calls and other internal monitor-request macros issued by TPFDF programs.

"DISPLAY -- Display fields" on page 327
Display storage contents, register contents, and so on.

"EXIT -- Exit current entry" on page 331
Simulate an EXITC monitor-request macro.

"FLIP -- Exchange the contents of two storage and data levels" on page 331
Exchange the contents of two storage and data levels.

"FLUSH -- Flush current entry" on page 331
Allow the entry to complete processing without further tracing.

"GET -- Get a storage block and attach it to a storage level" on page 332
Get a storage block and attach it to a storage level.

"HELP -- Display trace help information" on page 332
See also "ZHELP -- Command help facility" on page 202.

"PROCESS -- Process message" on page 332
Process an input message without tracing it.

"REFSTOP -- Reference stop" on page 333
Continue processing the entry (ignoring other trace control options) until the entry reaches one of the selected reference-stop locations.

"REGSTOP -- Register stop" on page 334
Continue processing the entry (ignoring other trace control options) until the entry reaches one of the selected register-stop conditions.

"REL -- Release a storage block from a storage level" on page 334
Release a storage block from a storage level.

“RUNSTOP -- Run stop” on page 335
Continue processing the entry (ignoring other trace control options) until the entry branches into or out of the selected run-stop location.

“SET -- Set fields” on page 336
Set storage contents, register contents, and so on.

“SKIP -- Skip next instruction or macro” on page 340
Branch past next instruction or macro.

“STEP -- Control instruction stepping” on page 340
Start or stop instruction stepping[3].

“SUBSTEP -- Subroutine stepping” on page 342
When instruction stepping is active, avoid stopping at instructions in a subroutine.

“SWAP -- Swap current entry” on page 344
Swap from tracing the current entry to tracing a created entry.

Z...
Any input message starting with Z. ALCS trace facility processes the message as an ALCS command. Note that this unconditionally passes the input message to the ALCS command processor. To pass the message to an application, use PROCESS Z....

Note that ALCS trace facility can support user-defined commands. The format and function of these user-defined commands (if any) are installation-dependent.

Restriction

You cannot use the ZAKEY command to set trace commands.

Tracing create-type macros

When processing stops at a create-type monitor-request macro (CREMC, CREDC, or CREXC), the ALCS trace facility behaves slightly differently. When you press Enter following the normal ALCS trace facility message, ALCS does not proceed immediately to the next macro to be traced. Instead, it displays a message to show that there is a new entry. This message includes a header like this:

```
TRACE -- prog   NEW ENTRY
```

Where:

prog
Name of the program that processes the new entry.

Depending on the display (**D**=) trace control option, there may be more information following the standard header.

Following this message, press Enter again to proceed to the next macro in the program that ***issues*** the create-type monitor-request macro (***not*** to the first monitor-request macro in the new entry). “SWAP -- Swap current entry” on page 344 explains how to swap from tracing one entry to another.

Tracing system errors

About this task

Program exceptions, SERRC or SYSRA monitor-request macros, and application program errors detected by the ALCS online monitor normally generate a system error dump and send a system error message to

[3] ALCS trace facility also supports the old ISTEP and UNSTEP commands for the benefit of users who are familiar with these commands. ISTEP is a synonym for STEP ON. UNSTEP is a synonym for STEP OFF.

the RO CRAS. But when ALCS trace facility is tracing an entry, it intercepts these system errors. It does not generate a dump and does not send a message to the RO CRAS. Instead, it:

Procedure

1. Stops processing the entry.
2. Displays the system error message.
3. Resets the next sequential instruction address for the entry to re-execute the failing instruction or monitor-request macro.
4. Displays the normal ALCS trace facility message (on the next line, following the system error message).

Results

This provides an opportunity to correct the error condition (see, for example, "SET -- Set fields" on page 336). After correcting the error, press Enter to re-execute the failing instruction or monitor-request macro. Alternatively, you can branch past the failing instruction or monitor-request macro (see "SKIP -- Skip next instruction or macro" on page 340 and "BRANCH -- Branch to address" on page 325).

Note: ALCS conversational tracing ***always*** traces system errors, regardless of the trace options specified.

Tracing EXITC

Eventually, all the entries that process the input message terminate with an EXITC monitor-request macro. When this happens, the ALCS trace facility starts tracing any exit intercept program set up by the SXIPC monitor-request macro, and then shows that it is ready to trace another input message by displaying:

```
TRACE -- ENTER MESSAGE TO TRACE
```

Enter the next message to trace, or ZTRAC STOP to stop conversational trace.

Note: ALCS conversational tracing ***always*** traces EXITC, regardless of the trace options you specify.

If records are held when the EXITC macro is reached then the following message is produced:

```
TRACE -- prog list EXITC -- WARNING -- n RECORDS HELD
```

If resources are held when the EXITC macro is reached then the following message is produced:

```
TRACE -- prog list EXITC -- WARNING -- n RESOURCES HELD
```

If TPFDF files are open when the EXITC macro is reached then the following message is produced:

```
TRACE -- prog list EXITC -- WARNING -- n TPFDF FILES OPEN
```

If you want a dump to be placed on the diagnostic file, reply to these messages by pressing Enter.

If you do not want a dump to be placed on the diagnostic file, reply to the messages with the EXIT command

Tracing High Level Language (HLL) Programs

You can use conversational trace to trace the execution of ALCS macros called from HLL programs. You can also trace the execution of assembler instructions in assembler library routines called from a HLL. It is not possible to trace the instructions generated by the language compilers or any code that modifies the ALCS reserved registers.

HLL language support uses the HLLCC macro to interface with ALCS. The following messages are displayed by conversational trace when a HLL program is executed:

```
TRACE - prog list HLLCC CREATING HLL ENVIRONMENT
```

A new HLL environment is created for the ECB. A control block will be attached at monitor level 3.

```
TRACE - prog list HLLCC LOADING MODULE=module
```

The module *module* is loaded (if not previously loaded). HLL support loads various modules from the LE runtime library depending on the languages being used.

```
TRACE - prog list HLLCC GETMAIN LENGTH=length
```

A contiguous area of length *length* is obtained from the entry's HLL storage units.

```
TRACE - prog list HLLCC GETMAINSHARED LENGTH=length
```

A contiguous area of length *length* is obtained from main storage.

```
TRACE - prog list HLLCC FREEMAIN LENGTH=length
LOCATION=address
```

A previously obtained area of storage is returned to the entry's HLL storage unit. The storage is at location *address* and is *length* bytes long.

```
TRACE - prog list HLLCC DELETING HLL ENVIRONMENT
```

The entry's current HLL environment is deleted. The control block at monitor level 3 and associated HLL storage units are released.

If you use the DISPLAY verb in COBOL application programs during debugging, then the messages created are routed to the display terminal.

Tracing a remote terminal

The normal response to ZTRAC CONV,START,T=term,... is:

```
TRACE -- AWAITING INPUT FROM term
```

ALCS traces the next input message or ALCS command from terminal *term*, except when it is:

Clear key
ALCS processes this without tracing it.

Null message
ALCS discards this without tracing it.

ZTRAC command
ALCS processes this without tracing it.

ZLOGF command
ALCS processes this without tracing it.

Trace continues as for normal conversational trace.

Asynchronous trace

The normal response to `ZTRAC CONV,INTERCEPT,ON` is:

```
TRACE -- AWAITING PROGRAM EXECUTION
```

ALCS traces the next entry that uses any of the programs included in your trace control options.

Processing then stops and the trace displays a message that shows the entry is paused, waiting to use the program. The message includes a header like this:

```
TRACE -- prog INTERCEPTED ENTRY
```

Where:

prog
The name of the program that is about to process the entry.

Depending on the display (`D=`) trace control option, there may be more information following the header shown above.

You can now either:

- Press Enter to start processing.
- Enter a trace control command.

Trace continues as for normal conversational trace. When you have finished tracing the entry, you can restart asynchronous trace by entering `ZTRAC CONV,INTERCEPT,ON`.

ALCS trace control commands

ADSTOP -- Address stop

When conversational tracing is active, use the ADSTOP command to continue processing the entry without tracing it, ignoring all other trace control options, until the program reaches one of the address-stop locations specified in the A= trace control parameters.

When ALCS trace facility has reached one of the address-stop locations, the ADSTOP command will process this instruction before continuing to the next address-stop location.

The format of the command is:

ALCS trace facility stops before the instruction at the address-stop location executes. Then it sends a display that is the same as the instruction step display (see “STEP -- Control instruction stepping” on page 340). Then ALCS trace facility continues as before the ADSTOP command.

It is possible that the entry never gets to an address-stop location. To guard against this, ALCS trace facility stops the entry if any of the following occurs before it gets to an address-stop location:

- The entry exits.
- The entry sends a response message to the originating terminal.
- The entry executes 10 000 monitor-request macros.
- The entry executes 30 000 instructions.

Note: For this purpose, DLAYC and DEFRC count as 100 macros to prevent a long wait before the response is received.

Normal responses

The same as for instruction step. See “STEP -- Control instruction stepping” on page 340.

If ALCS trace facility stops the entry before it gets to an address-stop location, one of the following lines can precede the standard header:

```
TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 10 000 macros

TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 30 000 instructions
```

In either case, you can re-enter ADSTOP to continue searching for the address stop location.

ANYSTOP -- Any stop

When conversational tracing is active, use the ANYSTOP command to continue processing the entry without tracing it, ignoring all other trace control options, until the program reaches one of the stop conditions specified in any of the A=, G= or R= trace control parameters.

The format of the command is:

See “ADSTOP -- Address stop” on page 323, “REFSTOP -- Reference stop” on page 333, and “REGSTOP -- Register stop” on page 334 for information as to where the trace facility stops depending on whether the stop condition reached is that specified on the A=, R= or G= trace control parameter.

When the trace facility stops it sends a display that is the same as the instruction step display (see “STEP -- Control instruction stepping” on page 340). Then ALCS trace continues as before the ANYSTOP command.

It is possible that the entry never reaches an address-stop, reference-stop or register-stop condition. To guard against this, ALCS trace facility stops the entry if any of the following occurs before it reaches an address-stop, reference-stop or register-stop condition.

- The entry exits.
- The entry sends a response message to the originating terminal.
- The entry executes 10 000 monitor-request macros.

- The entry exceeds 30 000 instructions.

Note: For this purpose, DLAYC and DEFRC count as 100 macros to prevent a long wait before the response is received.

Normal response

The same as for instruction step. See “STEP -- Control instruction stepping” on page 340.

If ALCS trace facility stops the entry before it reaches an address-stop, reference-stop or register-stop condition, one of the following lines can precede the standard header:

```
TRACE - STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 10000 macros

TRACE - STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 30000 instructions
```

In either case, you can reenter ANYSTOP to continue searching for the address-stop, reference-stop or register-stop condition.

BRANCH -- Branch to address

When conversational tracing is active, use the BRANCH command to branch to an address; that is, to set the next instruction address. BRANCH is equivalent to SET IA=.

The format of the command is:

Where:

address
Branch-to address. That is, the new value of the next sequential instruction address. One of:

hex
Main storage address in hexadecimal.

***disp*(*gpr*)**
Displacement (*disp*) in hexadecimal from the address in general register *gpr*.

Where *gpr* can be:

R0, R00, RG0, or RAC
General register 0

R1, R01, or RG1
General register 1

R2, R02, RG2, or RGA
General register 2

R3, R03, RG3, or RGB
General register 3

R4, R04, RG4, or RGC
General register 4

R5, R05, RG5, or RGD
General register 5

R6, R06, RG6, or RGE
General register 6

R7, R07, RG7, or RGF
General register 7

R8, R08, RG8, or RAP
General register 8

R9, R09, RG9, or REB
General register 9

R14, RG14, or RDA
General register 14

R15, RG15, or RDB
General register 15

***disp*(*gpr1*,*gpr2*)**
Displacement (*disp*) in hexadecimal from the address in general register *gpr2* plus the index value in *gpr1*. *gpr1* and *gpr2* can be any general register names (see the above list).

***disp*(SB*x*)**
Displacement (*disp*) in hexadecimal into the storage block attached at level *x*.

***disp*(AUT)**
Displacement (*disp*) in hexadecimal into the automatic storage block.

LIST *disp*
Displacement (*disp*) in hexadecimal into the application program.

BRANCH first evaluates *address* in any of the above formats as a 4-byte (32-bit) number. It interprets the 4-byte number as a 24-bit address (ignores the high-order ***byte***) if the entry is running in 24-bit addressing mode, or as a 31-bit address (that is, it ignores the high-order bit) if the entry is running in 31-bit addressing mode.

You can use SET IA= or BRANCH during macro trace as well as during instruction step. This allows you, for example, to ***branch around*** a monitor-request macro You can also set the instruction address to addresses outside the active program -- for example, to a subroutine in an ECB work area or in an attached block.

Normal response

The same as for instruction step. See "STEP -- Control instruction stepping" on page 340.

DETAIL -- Display internal TPFDF macro calls

When conversational trace and TPFDF macro trace are active, use the DETAIL command to display internal TPFDF macro calls and other internal monitor-request macros issued by TPFDF programs.

The format of the command is:

Where:

ON
Display internal TPFDF macro calls and other internal monitor-request macros.

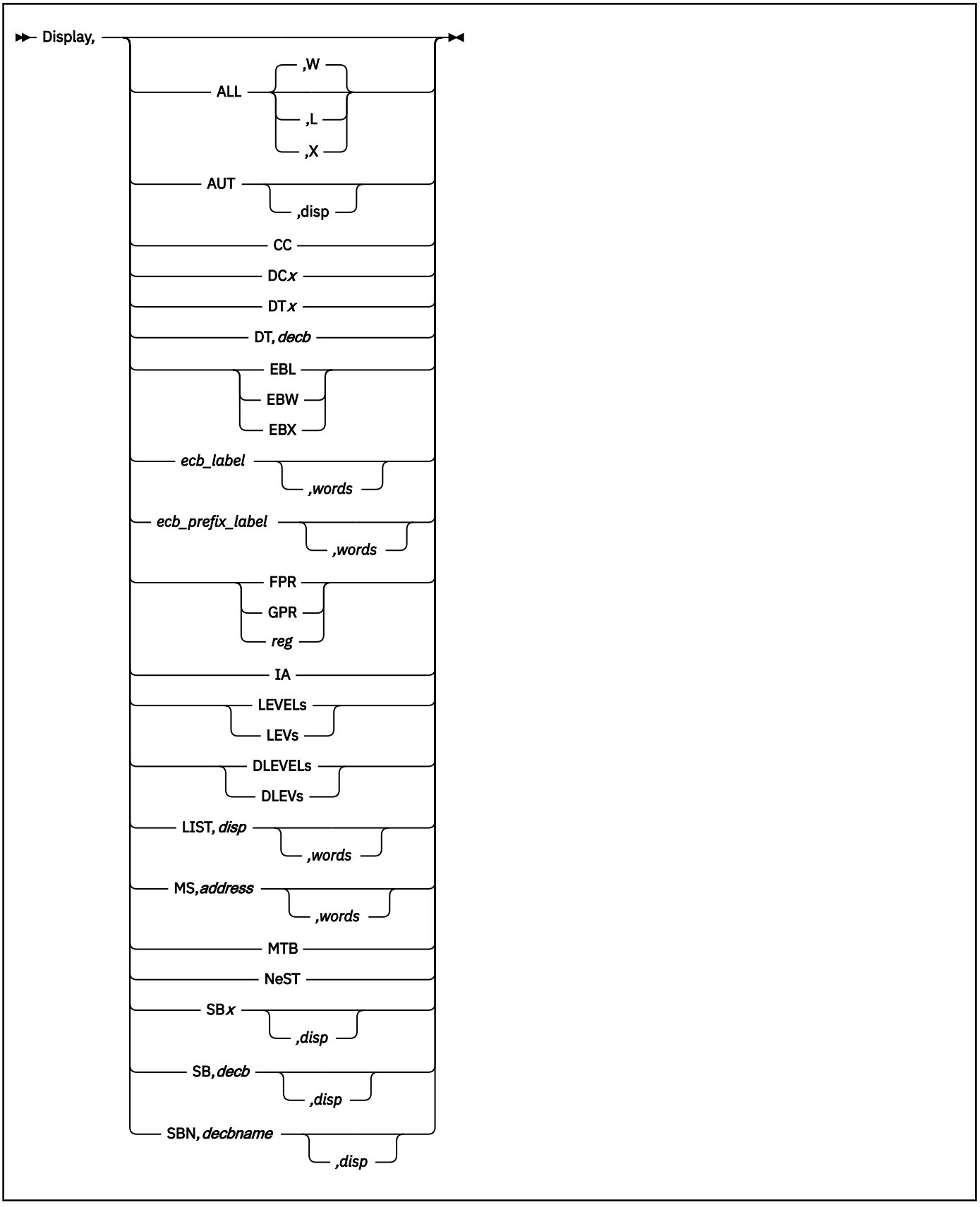
OFF
Do not display internal TPFDF macro calls and other internal monitor-request macros.

Ensure that you have previously used the ZTRAC P= trace control parameter to specify the TPFDF programs to be traced. For example ZTRAC START,P=UFB*x*.

DISPLAY -- Display fields

When conversational tracing is active, use the DISPLAY command to display the contents of main storage, registers, and so on.

The format of the command is:



Where:

`Display` with no parameters repeats the trace display for the current monitor-request macro (or for the current instruction if instruction stepping).

ALL[,W|,L|,X]
Full-screen display of registers, storage and data levels, ECB work area, and so on, where:

L
Display includes ECB local program work area (CE1WKC, fields EBL...).

W
Display includes ECB work area 1 (CE1WKA, fields EBW...).

X
Display includes ECB work area 2 (CE1WKB, fields EBX...).

AUT[,*disp*]
Display contents of the current automatic storage block. *disp* is the displacement within the block where the display starts; it is in hexadecimal and defaults to 0.

CC
Display the current setting of the condition code.

DC*x*
Display information about blocks detached by TPFDF. Specify DC0 for information about detached blocks for level D0, and so on.

DT*x*
Display information about detached blocks for level *x*. Specify DT0 for information about detached blocks for level D0, and so on.

DT,*decb*
Display information about detached blocks for the DECB at address *decb*.

EBL|EBW|EBX
Display contents of ECB local program work area CE1WKC (EBL), or ECB work area 1, CE1WKA (EBW) or ECB work area 2, CE1WKB (EBX).

***ecb_label*[,*words*]**
Display contents of ECB field. *ecb_label* is the name of the field in the EB0EB DSECT. *words* is the number of fullwords to display; it is in decimal and it defaults to 4.

***ecb_prefix_label*[,*words*]**
Display contents of ECB prefix field. *ecb_prefix_label* is the name of the field in the EB0EB DSECT. *words* is the number of fullwords to display; it is in decimal and it defaults to 4.

FPR|GPR|*reg*
Display contents of all floating-point registers (FPR), of all general registers (GPR), or of an individual register (*reg*). *reg* is one of:

F, FR, FPR, or FPRS
All floating-point registers

FP0
Floating-point register 0

FP2
Floating-point register 2

FP4
Floating-point register 4

FP6
Floating-point register 6

G, R, GR, GPR, GPRS, REG, or REGS
All general registers (except registers 10--13, which cannot be displayed)

R0, R00, RG0, or RAC
General register 0

R1, R01, or RG1
General register 1

R2, R02, RG2, or RGA
General register 2

R3, R03, RG3, or RGB
General register 3

R4, R04, RG4, or RGC
General register 4

R5, R05, RG5, or RGD
General register 5

R6, R06, RG6, or RGE
General register 6

R7, R07, RG7, or RGF
General register 7

R8, R08, RG8, or RAP
General register 8

R9, R09, RG9, or REB
General register 9

R14, RG14, or RDA
General register 14

R15, RG15, or RDB
General register 15

IA
Display the next instruction address.

LEVELs|LEVs
Display the contents of the ECB data and storage levels.

DLEVELs|DLEVs
Display the contents of the data and storage levels for in-use DECBs.

LIST,*disp*[,*words*]
Display the contents of the program that is currently executing. *disp* is the displacement (listing address) where the display starts; it is in hexadecimal. *words* is the number of fullwords to display; it is in decimal and defaults to 4.

MS,*address*[,*words*]
Display the contents of main storage at *address*, one of:

hex
Main storage address in hexadecimal.

***disp*(*gpr*)**
Displacement (*disp*) in hexadecimal from the address in general register *gpr*. *gpr* can be any general register name (see list above).

***disp*(*gpr1*,*gpr2*)**
Displacement (*disp*) in hexadecimal from the address in general register *gpr2* plus the index value in *gpr1*. *gpr1* and *gpr2* can be any general register names (see list above).

***disp*(SB*x*)**
Displacement (*disp*) in hexadecimal into the storage block attached at ECB level *x*.

***disp*(SB,*decb*)**
Displacement (*disp*) in hexadecimal into the storage block attached to the DECB at address *decb*.

***disp*(AUT)**
Displacement (*disp*) in hexadecimal into the automatic storage block.

LIST *disp*
Displacement (*disp*) in hexadecimal into the application program.

DISPLAY first evaluates *address* in any of the above formats as a 4-byte (32-bit) number. Then it interprets the 4-byte number as a 24-bit address (that is, it ignores the high-order ***byte***) if the entry is

running in 24-bit addressing mode. It interprets it as a 31-bit address (that is, it ignores the high-order bit) if the entry is running in 31-bit addressing mode.

words is the number of fullwords to display; it is in decimal and defaults to 4.

MTB
Display the entry macro trace block.

NEST|NST
Display program nesting information.

SB*x*[,*disp*]
Display the contents of the storage block attached at ECB level *x*. Specify SB0 for the block attached at level D0, and so on. *disp* is the displacement within the block where the display starts; it is in hexadecimal and it defaults to 0.

SB,*decb*[,*disp*]
Display the contents of the storage block attached to the DECB at address *decb*. *disp* is the displacement within the block where the display starts; it is in hexadecimal and it defaults to 0.

SBN,*decbname*[,*disp*]
Display the contents of the storage block attached to the DECB with name *decbname*. *disp* is the displacement within the block where the display starts; it is in hexadecimal and it defaults to 0.

EXIT -- Exit current entry

When conversational tracing is active, use the EXIT command to force the current entry to exit by simulating an EXITC monitor-request macro, including taking any exit intercept program set up by the SXIPC monitor-request macro. Unlike the real EXITC, it does not dump if it detects that records or resources are held, or general tapes are assigned. The format of the command is:

FLIP -- Exchange the contents of two storage and data levels

When conversational trace is active use the FLIP command to exchange the contents of two storage and data levels.

The format of the command is:

Where:

D*n*
Level symbol: D0 for level 0, and so on up to DF for level 15.

D*m*
Level symbol: D0 for level 0, and so on up to DF for level 15.

Normal responses

The same as for instruction stepping. See “STEP -- Control instruction stepping” on page 340.

FLUSH -- Flush current entry

When conversational tracing is active, use the FLUSH command to allow processing of the current entry to complete without further tracing. The format of the command is:

Normal responses

```
TRACE -- Hit enter to proceed
```

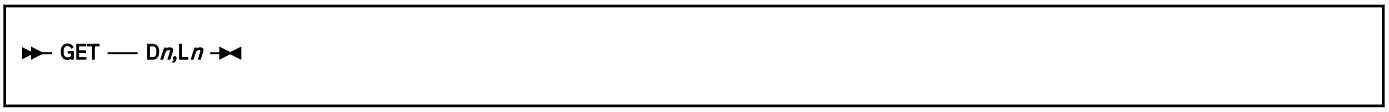
If the entry being traced generates an output message, this is sent to the terminal and processing stops. Press Enter to continue. When processing of the current entry is complete, the following message is sent to the terminal:

```
TRACE -- Enter message to be traced
```

GET -- Get a storage block and attach it to a storage level

When conversational trace is active use the GET command to get a storage block, initialize it to zeroes, and attach it to a storage level.

The format of the command is:



Where:

D*n*
Level symbol: D0 for level 0, and so on up to DF for level 15.

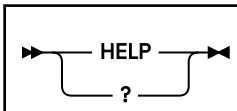
L*n*
Size symbol: L0, and so on up to L8, or LX (LX requests the largest block size that your ALCS system supports).

Normal responses

The same as for instruction stepping. See “STEP -- Control instruction stepping” on page 340.

HELP -- Display trace help information

When conversational tracing is active, use the HELP command to display help information about conversational mode trace commands. The format of the command is:



See also “ZHELP -- Command help facility” on page 202 for further information on invoking help.

PROCESS -- Process message

When conversational tracing is active, use the PROCESS command followed by any input message to process the message (without tracing it) while conversational trace is active. The normal message response appears on the screen.

The format of the command is:

Where:

message
Any input message.

Note: Some applications use terminal hold (AAA hold) to prevent concurrent processing of different input messages from the same originator. These applications either discard a second input message that arrives while the first is being processed, or do not process the second message until the first completes. Using the PR0CESS command with this type of application does not work unless the application contains special instructions to allow a second input message from a terminal that is being traced. The IPARS - ALCS application contains an example of this special logic in program UII1.

Normal responses

The response is the normal response generated by the input message.

If the input generates an output message, this is sent to the terminal and processing stops. Press Enter to continue. When processing of the current entry is complete, the following message is sent to the terminal:

```
TRACE -- Enter message to be traced
```

REFST0P -- Reference stop

When conversational tracing is active, use the REFST0P command to continue processing the entry without tracing it, ignoring all other trace control options, until the program refers to one of the reference-stop locations specified in the R= trace control parameter.

When ALCS trace facility has reached one of the reference-stop locations, the REFST0P command will process this instruction before continuing to the next reference-stop location.

The format of the command is:

ALCS trace facility stops before the instruction which refers to the reference stop. Then it sends a display that is the same as the instruction step display (see “STEP -- Control instruction stepping” on page 340). Then ALCS trace facility continues the same as before the REFST0P command.

It is possible that the entry never reaches a reference-stop location. To guard against this, ALCS trace facility stops the entry if any of the following occurs before it gets to a reference-stop location:

- The entry exits.
- The entry sends a response message to the originating terminal.
- The entry executes 10 000 monitor-request macros.
- The entry executes 30 000 instructions.

Note: For this purpose, DLAYC and DEFRC count as 100 macros to prevent a long wait before the response is received.

Normal responses

The same as for instruction step. See “STEP -- Control instruction stepping” on page 340.

If ALCS trace facility stops the entry before it gets to a reference stop location, one of the following lines can precede the standard header:

```
TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 10 000 macros

TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 30 000 instructions
```

In either case, you can re-enter REFSTOP to continue searching for the reference stop location.

REGSTOP -- Register stop

When conversational tracing is active, use the REGSTOP command to continue processing the entry without tracing it, ignoring all other trace control options, until the program reaches one of the register-stop conditions specified in the G= trace control parameter.

When ALCS trace facility has reached one of the register-stop conditions, the REGSTOP command will stop at each instruction or macro until the register-stop conditions are no longer matched.

The format of the command is:

ALCS trace facility stops after the instruction which modifies the register such that a register-stop condition is matched. When the trace facility stops it sends a display that is the same as the instruction step display (see “STEP -- Control instruction stepping” on page 340). Then ALCS trace continues as before the REGSTOP command.

It is possible that the entry never reaches a register-stop condition. To guard against this, ALCS trace facility stops the entry if any of the following occurs before it reaches a register-stop condition.

- The entry exits.
- The entry sends a response message to the originating terminal.
- The entry executes 10 000 monitor-request macros.
- The entry exceeds 30 000 instructions.

Note: For this purpose, DLAYC and DEFRC count as 100 macros to prevent a long wait before the response is received.

Normal responses

The same as for instruction step. See “STEP -- Control instruction stepping” on page 340.

If ALCS trace facility stops the entry before it reaches a register-stop condition, one of the following lines can precede the standard header:

```
TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 10 000 macros

TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 30 000 instructions
```

In either case, you can reenter REGSTOP to continue searching for the register-stop condition.

REL -- Release a storage block from a storage level

When conversational trace is active use the REL command to release a storage block from the specified storage level.

The format of the command is:

Where:

D*n*
Level symbol: D0 for level 0, and so on up to DF for level 15.

Normal responses

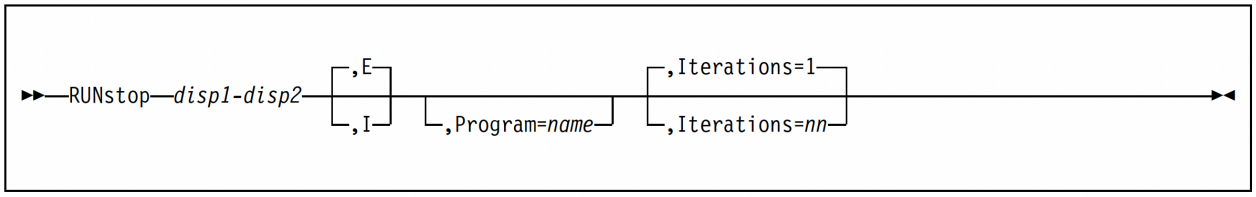
The same as for instruction stepping. See "STEP -- Control instruction stepping" on page 340.

RUNSTOP -- Run stop

When conversational tracing is active, use the RUNSTOP command to continue processing the entry without tracing it, ignoring all other trace control options, until one of the following occurs:

- The program branches out of the exclusive range of instruction addresses (if specified).
- The program reaches the next instruction after the exclusive range of instruction addresses (if specified).
- The program has performed the requested number of iterations of the exclusive range of instruction addresses (if iterations is specified).
- The program branches to an instruction with an address within the inclusive range of instruction addresses (if inclusive mode is specified).

The format of the command is:



Where:

disp1-disp2
A range of instruction addresses, each specified as a displacement in hexadecimal into the application program (as it appears in the program listing), where *disp2* is greater than *disp1*. The displacement can be any value in the range X'20' through X'FFFFFF'.

E or I
Specify E (the default) to monitor a range of instruction addresses (exclusive range mode). Monitoring starts when the program executes the instruction at displacement *disp1*. Monitoring stops when the program branches to an instruction with an address outside the range of instruction addresses or reaches the next sequential instruction after the end of the range. Tracing of the entry does not resume until monitoring has occurred (and has stopped).

Specify I to stop when the program branches to an instruction with an address within the range of instruction addresses (inclusive range mode).

Iterations=*nn*
For exclusive range mode, stop at the last instruction in the range when the range of instructions has been executed *nn* times. *nn* is a decimal number in the range 1 through 9999.

Program=*name*
The 4-character program name containing the range of instruction addresses. If not specified the range is contained in the program being traced when the RUNSTOP command is entered.

It is possible that the entry may never satisfy the specified RUNSTOP conditions. To guard against this, ALCS trace facility stops the entry if any of the following occurs before it enters or leaves the instruction address range.

- The entry exits.
- The entry sends a response to the originating terminal.
- The entry executes 10 000 monitor-request macros.
- The entry exceeds 30 000 instructions.

Note: For this purpose, DLAYC and DEFRC count as 100 macros to prevent a long wait before the response is received.

Normal responses

The same as for instruction stepping. See “STEP -- Control instruction stepping” on page 340.

If ALCS trace facility stops the entry before it enters or leaves the instruction address range, when RUNSTOP is active, one of the following lines can precede the standard header:

```
TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 10 000 macros
```

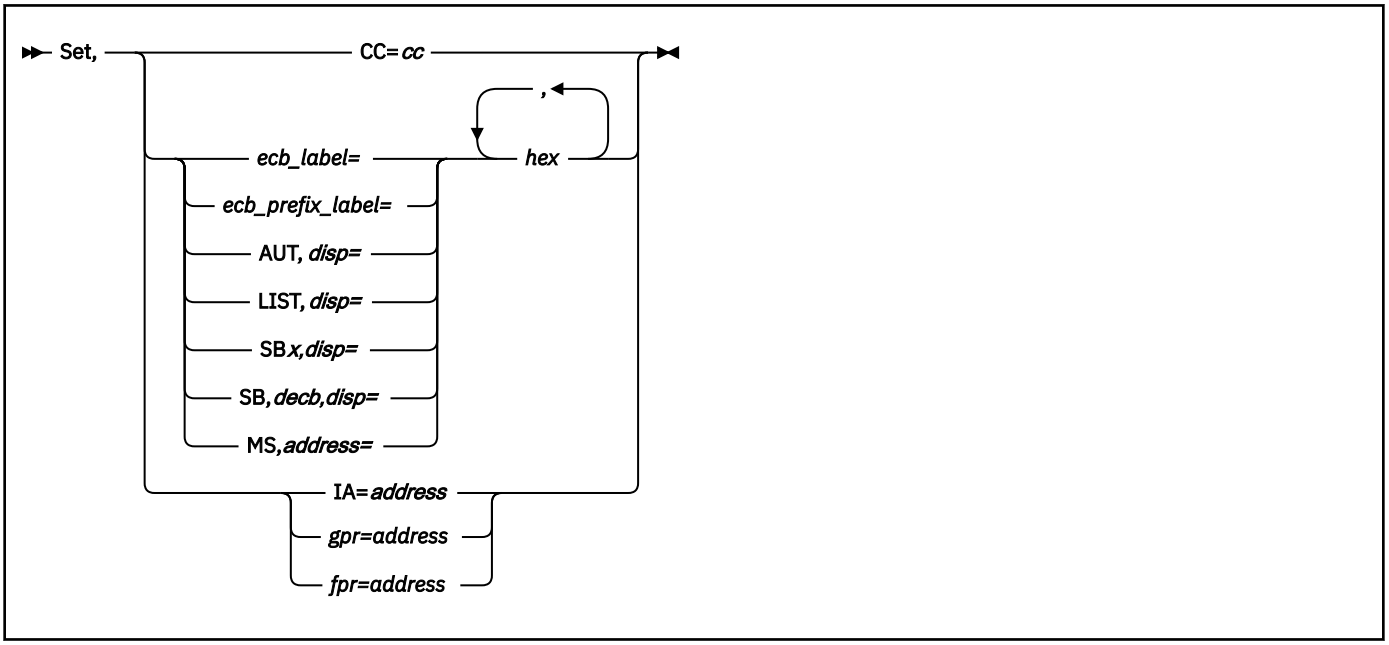
or:

```
TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 30 000 instructions
```

SET -- Set fields

When conversational tracing is active, use the SET command to set the contents of main storage, registers, and so on.

The format of the command is:



Where:

CC=*cc*
Set the condition code. *cc* is the new condition code value: 0, 1, 2, or 3.

ecb_label=hex,...
Store the replacement data *hex,...* in the ECB at field *ecb_label* where:

ecb_label
Name of field in the EB0EB DSECT.

hex,...
One or more strings each of one or more hexadecimal digits. A comma (or space) separates each string from the next. SET adds leading zeros if necessary so that each string forms a whole number of bytes. Then it updates the ECB by storing the first string at *ecb_label*. It stores the second string (if any) immediately following the first, and so on.

ecb_prefix_label=hex,...
Store the replacement data *hex,...* in the ECB prefix at field *ecb_prefix_label*, where:

ecb_prefix_label
Name of field in the EB0EB DSECT.

hex,...
One or more strings each of one or more hexadecimal digits. A comma (or space) separates each string from the next. SET adds leading zeros if necessary so that each string forms a whole number of bytes. Then it updates the ECB prefix by storing the first string at *ecb_prefix_label*. It stores the second string (if any) immediately following the first, and so on.

AUT,*disp=hex,...*
Store the replacement data *hex,...* in the automatic storage block at displacement *disp*, where:

disp
Displacement in hexadecimal within the automatic storage block.

hex,...
One or more strings each of one or more hexadecimal digits. A comma (or space) separates each string from the next. SET adds leading zeros if necessary so that each string forms a whole number of bytes. Then it updates the automatic storage block by storing the first string at displacement *disp*. It stores the second string (if any) immediately following the first, and so on.

LIST,*disp=hex,...*
Store the replacement data *hex,...* at displacement *disp* in the program that is currently executing, where:

disp
Displacement in program in hexadecimal.

hex,...
One or more strings each of one or more hexadecimal digits. A comma (or space) separates each string from the next. SET adds leading zeros if necessary so that each string forms a whole number of bytes. Then it updates the program by storing the first string at displacement *disp*. It stores the second string (if any) immediately following the first, and so on.

SET LIST can only modify a program that is loaded for test. See "ZPCTL -- Load/unload programs and installation-wide monitor exits" on page 244 for a description of this.

SB*x*,*disp=hex,...*
Store the replacement data *hex,...* in the storage block attached at ECB level *x* at displacement *disp*, where:

disp
Displacement in hexadecimal within the storage block.

hex,...
One or more strings each of one or more hexadecimal digits. A comma (or space) separates each string from the next. SET adds leading zeros if necessary so that each string forms a whole number of bytes. Then it updates the storage block by storing the first string at displacement *disp*. It stores the second string (if any) immediately following the first, and so on.

SB,*decb*,*disp=hex,...*
Store the replacement data *hex,...* in the storage block attached to the DECB at address *decb*, where:

disp
Displacement in hexadecimal within the storage block.

hex,...
One or more strings each of one or more hexadecimal digits. A comma (or space) separates each string from the next. SET adds leading zeros if necessary so that each string forms a whole number of bytes. Then it updates the storage block by storing the first string at displacement *disp*. It stores the second string (if any) immediately following the first, and so on.

MS,*address=hex,...*
Store the replacement data *hex*,... at *address* in main storage, where:

address
Main storage address in any of the formats that SET gpr=address supports.

hex,...
One or more strings each of one or more hexadecimal digits. A comma (or space) separates each string from the next. SET adds leading zeros if necessary so that each string forms a whole number of bytes. Then it updates main storage by storing the first string at address *address*. It stores the second string (if any) immediately following the first, and so on.

SET first evaluates *address* as a 4-byte (32-bit) number. Then it interprets the 4-byte number as a 24-bit address (that is, it ignores the high-order byte) if the entry is running in 24-bit addressing mode. It interprets it as a 31-bit address (that is, it ignores the high-order bit) if the entry is running in 31-bit addressing mode.

IA=*address*
Set the address of the next sequential instruction (NSI). This is the same as branching to *address*. *address* can be any of the formats that SET gpr=address supports.

SET first evaluates *address* as a 4-byte (32-bit) number. Then it interprets the 4-byte number as a 24-bit address (that is, it ignores the high-order byte) if the entry is running in 24-bit addressing mode. It interprets it as a 31-bit address (that is, it ignores the high-order bit) if the entry is running in 31-bit addressing mode.

You can use SET IA= or BRANCH during macro trace as well as during instruction step. This allows you, for example, to branch around a monitor-request macro. You can also set the instruction address to addresses outside the active program -- for example, to a subroutine in an ECB work area or in an attached block.

gpr=address
Set the contents of general register, where:

gpr
Name of general register, one of:

R0, R00, RG0, or RAC
General register 0

R1, R01, RG1, or RG1
General register 1

R2, R02, RG2, or RGA
General register 2

R3, R03, RG3, or RGB
General register 3

R4, R04, RG4, or RGC
General register 4

R5, R05, RG5, or RGD
General register 5

R6, R06, RG6, or RGE
General register 6

R7, R07, RG7, or RGF
General register 7

R8, R08, RG8, or RAP
General register 8

R14, RG14, or RDA
General register 14

R15, RG15, or RDB
General register 15

address
Replacement data. A 4-byte value that replaces the previous contents of the general register. One of:

hex
One to 8 hexadecimal digits. SET adds leading zeros if necessary to form a 4-byte value.

***disp*(*gpr*)**
Value formed by adding *disp* in hexadecimal to the contents of general register *gpr*. *gpr* can be any general register name (see list above).

***disp*(*gpr1*,*gpr2*)**
Displacement (*disp*) in hexadecimal off the address in general register *gpr2* plus the index value in *gpr1*. *gpr1* and *gpr2* can be any general register names (see list above, with the addition of REB, general register 9).

***disp*(SB*x*)**
Value formed by adding *disp* in hexadecimal to the address of the storage block attached at ECB level *x*.

***disp*(SB,*decb*)**
Value formed by adding *disp* in hexadecimal to the address of the storage block attached to the DECB at address *decb*.

***disp*(AUT)**
Value formed by adding *disp* in hexadecimal to the address of the automatic storage block.

LIST,*disp*
Storage address corresponding to displacement (*disp*) in hexadecimal into the application program. Note that this address is evaluated without regard to the address mode of the program.

fpr=hex
Set the contents of floating-point register, where:

fpr
Name of floating-point register, one of:

FP0
Floating-point register 0

FP2
Floating-point register 2

FP4
Floating-point register 4

FP6
Floating-point register 6

hex
Replacement data, 1 to 16 hexadecimal digits. SET adds leading zeros if necessary to form an 8-byte value that replaces the previous contents of the floating-point register.

Restrictions

SET MS can only modify main storage that an application program can modify; that is, entry storage and global area storage.

SET LIST and SET MS,LIST can only modify a program that is loaded for test. "ZPCTL -- Load/unload programs and installation-wide monitor exits" on page 244 describes this.

SKIP -- Skip next instruction or macro

When conversational tracing is active, use the SKIP command to skip (branch past) the next instruction or macro.

The format of the command is:

Notes:

1. When instruction stepping is active, ALCS trace facility stops twice for each monitor-request macro. It first sends the display for the BRANCH SAVE AND SET MODE (BASSM) instruction that links to the ALCS monitor. Do not enter SKIP at this time. (This is because SKIP branches past the BASSM instruction to constants that are part of the macro expansion. These constants are called macro parameters.) Instead, press Enter; ALCS trace facility then sends the display for the monitor-request macro. Now enter SKIP to skip the monitor-request macro.
2. Some monitor-request macros generate executable instructions before the BASSM and some generate executable instructions after the macro parameters. For example, FINWC generates a conditional branch following the macro parameters. When the trace display shows FINWC, then SKIP branches to the conditional branch instruction within the macro expansion, not to the next instruction following the macro.

Normal response

The same as for instruction step. See "STEP -- Control instruction stepping" on page 340.

STEP -- Control instruction stepping

When conversational tracing is active, use the STEP command to turn on (start) or turn off (stop) instruction stepping. During instruction stepping, ALCS trace facility stops at each instruction (as well as at each monitor-request macro).

The format of the command is:

Where:

ON
Turn on (start) instruction stepping.

OFF
Turn off (stop) instruction stepping.

The ALCS trace facility also supports the old ISTEP and UNSTEP commands for the benefit of users who are familiar with these commands. ISTEP is a synonym for STEP ON. UNSTEP is a synonym for STEP OFF.

It is possible that during instruction stepping the entry may enter a program which is not being traced and get into a loop in that untraced program. To guard against this, ALCS trace facility stops the entry if any of the following occurs before it returns to a traced program:

- The entry exits.
- The entry sends a response message to the originating terminal.
- The entry executes 10 000 monitor-request macros.
- The entry executes 30 000 instructions.

Note: For this purpose, DLAYC and DEFRC count as 100 macros to prevent a long wait before the response is received.

Restriction

When tracing assembler library routines called by C applications, be aware that instruction stepping is disrupted when execution switches back to the C environment. Set STEP OFF before executing any instruction that modifies an ALCS reserved register.

Normal responses

```
TRACE -- Hit enter to proceed
```

For STEP OFF, press Enter to execute the macro or instruction. ALCS trace facility then proceeds to the next monitor-request macro. Trace is no longer instruction stepping.

For STEP ON, press Enter to execute the macro. ALCS trace facility then proceeds to the next instruction (the instruction that follows the monitor-request macro). ALCS trace facility then displays a message that begins with a standard header.

Depending on the display (D=) trace control option, more information can follow the standard header.

Following the normal response, press Enter to execute the instruction. ALCS trace facility executes the instruction and then displays information about the next instruction. Press Enter again to execute that instruction, and so on. Optionally, enter any trace control command or ALCS command before pressing Enter.

Standard header:

```
TRACE -- address   opcode   operand   contents1  contents2
```

Where:

address
Location of the instruction. If this is within an application program, then it appears as the program name followed by the hexadecimal displacement (listing address). If not (for example, if it is in the ECB) it appears as the 8-digit hexadecimal storage address.

opcode
Symbolic operation code of the instruction. See *z/Architecture Principles of Operation* for details. If ALCS trace facility cannot recognize the instruction operation code, then the header includes the whole instruction in hexadecimal notation instead of the symbolic operation code and operand field.

operand
Operand field of the instruction. All terms in the operand field appear in hexadecimal. For example:

```
MVC   018(002,9),006(5)
```

shows a move to displacement hexadecimal 18 (decimal 24) from the address in general register 9 (REB).

contents
The contents or values of the operands, in hexadecimal, displayed as 2 fields of up to 8 bytes each. If the operand is a register, the display shows the contents of the register; if the operand is a storage location, the display shows the storage contents; if the operand is an address, the display shows the value at the address.

If the contents are more than 8 bytes, the last hexadecimal digit is replaced by 2 periods (..) to show that the displayed contents are not complete. If the storage location is not accessible, the display shows question marks (??).

Not all operation codes include a display of contents.

Branch instructions

If ALCS trace facility recognizes the instruction as a branch, it checks to see if the branch is taken. One of the following standard headers is displayed:

If the branch is taken:

```
TRACE -- address  opcode  operand          --> destination
```

If the branch is not taken:

```
TRACE -- address  opcode  operand             destination
```

Where:

destination
The instruction branches to this location, or would if the branch was taken.

When ALCS trace facility stops at a monitor-request macro while instruction stepping is active, it first sends the display for the BRANCH SAVE AND SET MODE (BASSM) instruction that links to the ALCS monitor. After pressing Enter, ALCS trace facility then sends the display for the monitor-request macro. Note that even during instruction stepping, ALCS trace facility does not trace the instructions in the ALCS monitor.

If ALCS trace facility stops the entry before it returns to a traced program, one of the following lines can precede the standard header:

```
TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 10 000 macros

TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 30 000 instructions
```

In either case, you can press Enter to continue instruction stepping.

SUBSTEP -- Subroutine stepping

When conversational trace is active and instruction stepping is active, use the SUBSTEP , OFF command to prevent stopping at instructions in a subroutine.

The format of the command is:

Where:

ON
Stop at instructions in subroutines. This is the default.

OFF
Do not stop at instructions in subroutines.

When instruction stepping and subroutine stepping is active (the default), ALCS trace facility stops at each instruction as well as at each monitor-request macro.

When instruction stepping is active and subroutine stepping is not active, if the instruction to be executed is a branch to a subroutine (BAL(R), BAS(R), BASSM, BRAS, BRASL) then the ALCS trace facility will proceed to, and stop at, the next sequential instruction following the branch instruction.

When ALCS trace facility is stopped at an instruction which branches to a subroutine, the SUBSTEP,OFF command will have no effect on this instruction.

It is possible that when subroutine stepping is not active, the entry may never return to the next sequential instruction following the branch instruction. To guard against this, ALCS trace facility stops the entry if any of the following occurs before it returns to the NSI.

- The entry exits.
- The entry sends a response to the originating terminal.
- The entry executes 10 000 monitor-request macros.
- The entry exceeds 30 000 instructions.

Note: For this purpose, DLAYC and DEFRC count as 100 macros to prevent a long wait before the response is received.

If instruction stepping is inactivated, subroutine stepping will be set to active (the default). A subsequent invocation of instruction stepping will step through a subroutine unless the SUBSTEP, OFF trace command is reissued.

Normal responses

The same as for instruction stepping. See “STEP -- Control instruction stepping” on page 340.

If ALCS trace facility detects a branch to a subroutine and stops at the next sequential instruction, when subroutine stepping is not active, one of the following line will precede the standard header:

```
TRACE -- prog SUBROUTINE AT LOCATION address NOT TRACED
```

Where:

prog
Program name.

address
Location of the branch instruction (4-digit hexadecimal program listing address).

or:

```
TRACE -- SUBROUTINE AT LOCATION address NOT TRACED
```

Where:

address
Location of the branch instruction (8-digit hexadecimal storage address).

If multiple subroutines are coded back to back then SUBROUTINE will be replaced by SUBROUTINES in the line.

If ALCS trace facility stops the entry before it reaches the next sequential instruction, when subroutine stepping is not active, one of the following lines can precede the standard header:

```
TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 10 000 macros
```

or:

```
TRACE -- STEP/ADSTOP/REFSTOP/REGSTOP/RUNSTOP/SUBSTEP halt after 30 000 instructions
```

SWAP -- Swap current entry

When more than one entry (ECB) is processing a traced input message, use the SWAP command, to swap from tracing one entry to another.

The format of the command is:

Initially only one entry processes an input message. However, that entry can create other entries. The original entry and the created entries are all processing the same input message.

Conversational trace can trace only one entry at a time. It keeps all the entries that are processing the input message in a ***stack***, with the entry that it is tracing at the top of the stack. If that entry creates another entry, the ALCS trace facility adds the new entry to the bottom of the stack.

SWAP moves the current entry to the bottom of the stack and makes the second entry in the stack the current entry. If there are many entries in the stack, SWAP can be entered repeatedly to make any entry the current entry. This command has no effect when there is only one entry processing the message.

Restrictions

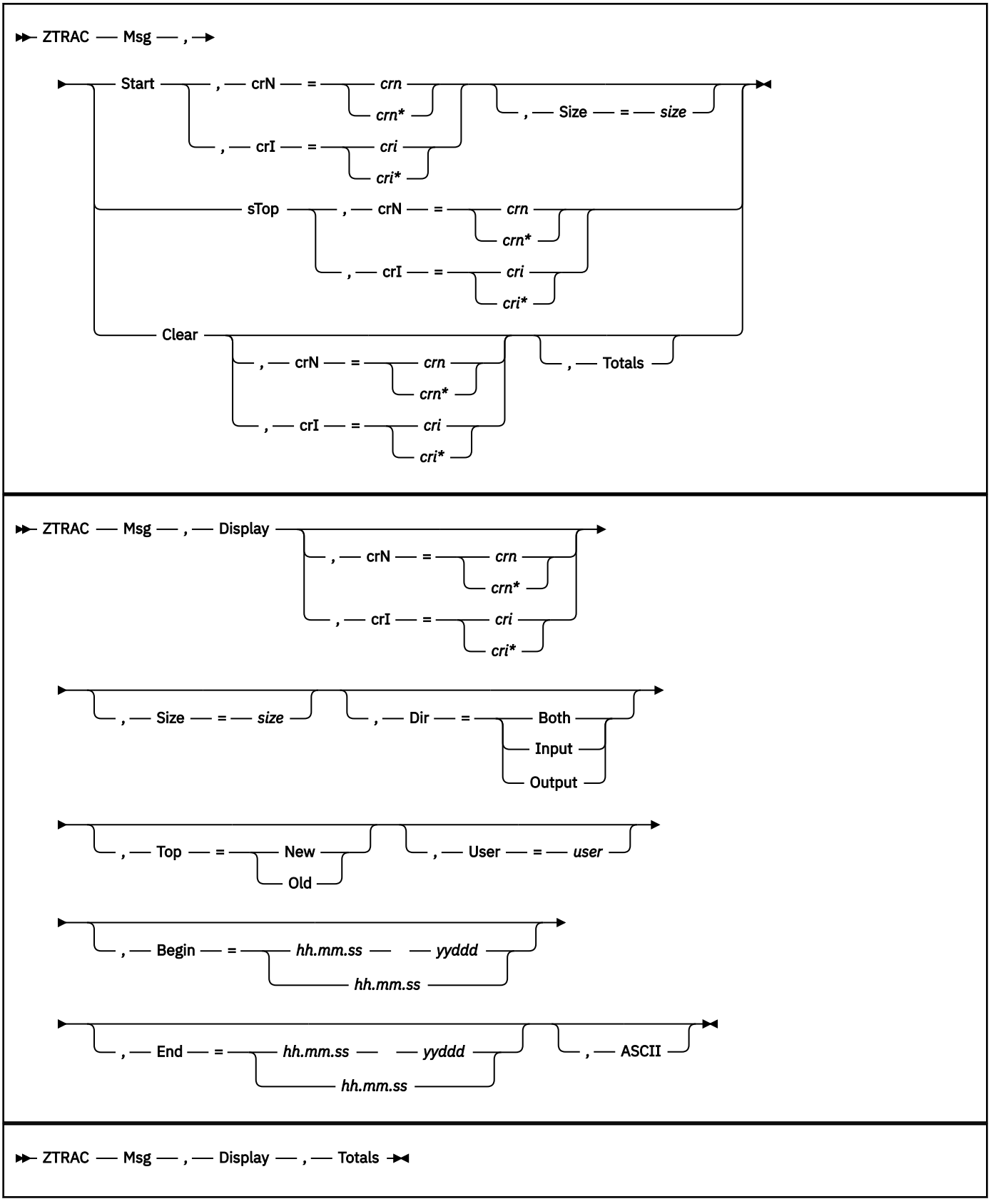
SWAP is not allowed when the entry being traced (the entry at the top of the stack):

- Holds records (for example by FIWHC macro).
- Holds resources (for example by CORHC or ENQC macro), or
- Assigns sequential files (by TOPNC or TASNC macro).

Starting, Stopping, Clearing, and Displaying an online message trace

You can control tracing -- ***from a Prime CRAS*** or ***AT1-16 CRAS authorized terminal***. You can display the trace from any ***CRAS terminal***.

The format of the command is:



Where:

Start
Start a trace

sTop
Stop a trace.

Clear
Clear a trace and message totals, or clear only message totals.

You can clear all resouces or clear only a specific resource. When you clear for a specific resource, then the display for that resource does not show any messages that were traced before the clear was issued. However you still may display those traced messages when you display all resouces.

Display
Display a trace or display message totals.

crI=*cri*
CRI of the resource (6 hexadecimal digits). CRIs starting with 00 or 01 (the CRAS resources) are not valid.

crN=*crn*
CRN of the resource (1 to 8 alphanumeric characters). Special CRNs; such as PRC, ROC, ATx, APx, are not valid.

crI=*cri
A generic CRI of the resource of 2 or 4 hexadecimal digits with a trailing asterisk. For example, 04* or 020A*.

crN=*crn**
A generic CRN of the resource of 1 through 7 characters with a trailing asterisk.

Size=*size*
Maximum size of a message before truncation occurs. You can specify any decimal value from 1 up to 3900.

Size zero (0) is a special case. Here ALCS only updates message totals but does not trace any message.

Totals
The Totals output depends on the selected parameter. Either

- For the Display parameter - output traced input and output message totals, or
- For the Clear parameter - clear input and output message totals.

Dir=Both|Input|Output
Display either both input and output (default), only Input, or only Output.

Top=New|Old
Display either the newest (New) message at the top of the display (default) or display the oldest (Old) message at the top of the display.

USER=*user*
Up to 8 characters of user information, which is passed to the ALCS online trace display installation-wide exit program AMG2. See *ALCS Installation and Customization* for details.

Begin=*hh.mm.ss*|*hh.mm.ss yyddd*
The TOD begin time in format hh.mm.ss (and optionally the date in format yyddd) of the first message to be displayed. If the date is omitted todays date will be used.

End=*hh.mm.ss*|*hh.mm.ss yyddd*
The TOD end time in format hh.mm.ss (and optionally the date in format yyddd) of the last message to be displayed. If date is omitted todays date will be used.

ASCII
Interpret the binary data as ASCII characters. Omit this parameter to interpret the data as EBCDIC characters (default).

Normal responses

Normal response to ZTRAC `msg start`, ZTRAC `msg stop`, and ZTRAC `msg clear` commands.

```
+     DXC8028I CMD  i hh.mm.ss TRAC  OK
```

Normal responses

Normal response to ZTRAC `msg display`.

```
+    DXC8088I CMD i hh.mm.ss  TRAC Traced messages
+    tod          CRN=crn SIZE=ffff eeeee

+    oooo **  dddddddd dddddddd dddddddd dddddddd   *ccccccccccccccc*
+     ⋮
```

Where:

eeeee
Direction indicator. Either INPUT or OUTPUT.

ffff
Size of message displayed.

oooo
Relative storage address of the message displayed.

****
Indicates that the line is repeated on the display.

tod
TOD clock value in format *hh.mm.ss.t yyddd* (hours, minutes, seconds, tenth of seconds, year, day).

dd...ddd
Contents (hexadecimal).

cc...ccc
Contents (character).

Notes:

1. ALCS traces input and output messages to the wrap around online trace area for the selected communication resources after it conditionally calls installation-wide exits USRCOM2 and USRCOM4 when those exits return with return code 0.
2. The TOP=Old display re-orders the traced messages. In order to do this, ALCS will use L3 short term pool records.

Normal responses

Normal response to ZTRAC `msg display,totals`

```
+    DXC8089I CMD i hh.mm.ss  TRAC Totals
+    CRN                                                REQUESTOR  DDHHMM
+    crn             c  Total input  messages.........im  orgcrn   ddhhmm
+                       Total output messages.........pm
+     ⋮

+    CRI                                                REQUESTOR  DDHHMM
+    cri             c  Total input  messages.........im  orgcrn   ddhhmm
+                       Total output messages.........pm
+     ⋮
```

Where

c
Counts only indicator. Either blank or C (for counts only).

im
Total number of input messages.

pm
Total number of output messages.

orgcrn
CRN of the originator of the trace request.

ddhhmm
Time of the original trace request.

Note: The totals include the callable service calls for a message.

Chapter 8. Maintaining ALCS

In many cases, the system programmers in an ALCS installation will determine and fix any problems that arise by means of the facilities provided by MVS and subsystems such as VTAM, using the appropriate MVS and subsystem manuals.

The *Network Program Products: General Information* provides a summary of network problem determination facilities for standard programs such as VTAM, Network Control Program, and NetView.

Specialized programs such as NPSI, NTO, NEF, and ALCI all have their own problem determination processes, which are documented in their program libraries.

This section describes some aspects of maintaining ALCS, including:

- Fixing system problems
- Tuning ALCS

MVS diagnostic facilities

This section contains descriptions of the following:

- Generalized trace facility (GTF)
- MVS dumps
- SLIP command
- Interactive problem control system (IPCS)

Generalized trace facility (GTF)

The generalized trace facility is a service aid program that is available for determining and diagnosing system problems. GTF records system and user-defined program events. Through GTF you can trace:

- Any combination of system events, such as all I/O interruptions and all SVC interruptions.
- Specific incidences of one type of system event, such as all I/O to a particular device.
- User-defined events that are generated by the GTRACE macro. ALCS uses GTRACE to record the beginning and completion of VTAM operations. VTAM uses GTRACE for terminal buffer tracing.

GTF produces output trace records either to buffers in virtual storage or to a data set.

You should use GTF for tracing events that cannot be recorded by ALCS problem determination facilities. For example, use GTF to obtain a buffer trace of VTAM I/O to a terminal, or for VTAM request parameter lists (RPLs) used by ALCS.

The following example shows how to start GTF to record VTAM RPLs used by ALCS. See the *MVS Service Aids* manual for detailed information on how to use GTF, and for an explanation of the user options shown in the example.

User	s gtf.gtf
System	$HASP100 GTF ON STCINRDR $HASP373 GTF STARTED TRACE=RNIO,USR,SLIP AHL121I SYS1.PARMLIB INPUT INDICATED AHL103I TRACE OPTIONS SELECTED --USR,RNIO,SLIP *28 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
User	28trace=usr,jobnamep

System	IEE600I REPLY TO 28 IS;TRACE=USR,JOBNAMEP *29 AHL101A SPECIFY TRACE EVENT KEYWORDS --JOBNAME=
User	29jobname=alcsrun
System	IEE600I REPLY TO 29 IS;JOBNAME=ALCSRUN *30 AHL102A CONTINUE TRACE DEFINITION OR REPLY END
User	30end
System	IEE600I REPLY TO 30 IS;END AHL103I TRACE OPTIONS SELECTED --USR AHL103I JOBNAME=(ALCSRUN) *31 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
User	31u
System	IEE600I REPLY TO 31 IS;U AHL031I GTF INITIALIZATION COMPLETE

To stop GTF:

User	P gtf
System	$HASP395 GTF ENDED AHL006I GTF ACKNOWLEDGES STOP COMMAND

After stopping GTF, use IPCS to format, display, and print the GTF output. See “Interactive problem control system (IPCS)” on page 354.

MVS dumps

About this task

Use MVS dump services when you need information that is not dumped by ALCS (for example, the ALCS communication table).

MVS produces four types of dumps:

Procedure

1. SYSUDUMP is a dump of user areas. It is formatted, so it can be printed directly. MVS produces this dump when a job abends and there is a SYSUDUMP DD statement in the JCL.
2. SYSABEND is a dump of user and system areas. It is formatted, so it can be printed directly. MVS produces this dump when a job abends and there is a SYSABEND DD statement in the JCL.
3. SYSMDUMP is an unformatted dump of user and system areas. MVS produces this dump when a job abends and there is a SYSMDUMP DD statement in the JCL. You must use IPCS to format, print or display SYSMDUMP dumps (see “Interactive problem control system (IPCS)” on page 354).
4. SVC dump is an unformatted dump of user and system areas. You do not need to supply a DD card to point to the dump data set, as MVS uses pre-allocated dump data sets with a data set name of SYS1.DUMP*xx*, where *xx* is a 2-digit decimal number ranging from 00 to an installation specified maximum. You must use IPCS to format, print or display SVC dumps (see “Interactive problem control system (IPCS)” on page 354).

   You can take an SVC dump of the ALCS address space by using the DUMP MVS system command (see the example below). You can also direct MVS to take an SVC dump when ALCS abends by setting a SLIP trap (see “SLIP command” on page 351). If dump speed is important to your installation, you should use SVC dumps, as they are faster than the other types of dump.

   ALCS can optionally take an SVC dump, when it terminates abnormally. ALCS will dump all the private storage in the address space along with the z/OS system trace (various internal system events such

as interrupts, dispatcher actions, I/O operations, and SVC calls) and summary data (operating system areas, control blocks, and ALCS tables).

Results

For detailed information on the different types of dumps and their contents, see the appropriate *MVS Initialization and Tuning Manual* manual.

The following example shows how you can take an SVC dump of the ALCS address space. See *MVS System Commands* manual for a description of the commands used.

User	d d,t
System	IEE853I 15.41.10 SYS1.DUMP TITLES 812 SYS1.DUMP DATA SETS AVAILABLE=010 AND FULL=000 NO DUMP DATA AVAILABLE FOR THE FOLLOWING EMPTY SYS1.DUMP DATA SETS: 00-09
User	DUMP COMM=(ALCS TEST SYSTEM)
System	*12 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
User	12jobname=ialegr01,sdata=(nosqa),end
System	IEE600I REPLY TO 12 IS;JOBNAME=IALEGR01,SDATA=(NOSQA),END *IEA911E COMPLETE DUMP ON SYS1.DUMP00 169 FOR ASID (001D)
User	d d,t
System	IEF196I IEF237I 424 ALLOCATED TO SYS00025 IEF196I IEF285I SYS1.DUMP00 KEPT IEF196I IEF285I VOL SER NOS= IASC02. IEE853I 15.58.11 SYS1.DUMP TITLES 384 SYS1.DUMP DATA SETS AVAILABLE=009 AND FULL=001 DUMP00 TITLE=ALCS TEST SYSTEM DUMP TAKEN TIME=15.50.09 DATE=12/10/90 NO DUMP DATA AVAILABLE FOR THE FOLLOWING EMPTY SYS1.DUMP DATA SETS: 01-09

SLIP command

The SLIP command controls serviceability level indication processing (SLIP), a diagnostic aid designed to intercept or trap certain system events. You can indicate what kinds of event you want trapped and what the system is to do when these events occur.

The following is an example of the kinds of events you can intercept. For a detailed description, see the *MVS System Commands*.

- Instruction fetch program event recording (PER) interruption. Use this event to trace the execution of instructions when ALCS trace does not apply. For example, you can trace the execution of a monitor installation-wide exit (see the example below).
- Storage alteration PER interruption. Use this event to produce an SVC dump or a trace to identify sources of storage damage. For example, you can identify application programs that damage the application global area.
- Abend. Use this event to take SVC dumps, or inhibit dumping, for certain ALCS abend codes. See the example below.

An event defined by the SLIP command is usually called a **SLIP trap**.

SVC dumps are written to SYS1.DUMP*xx*.

Examples of using SLIP

To produce an SVC dump when ALCS abends, use the commands:

```
SLIP SET,ID=AL01,J=jobname,C=Uxxxx,A=SVCD,AL=CU,ML=9999,END
SLIP SET,ID=AL02,J=jobname,C=S122,A=SVCD,AL=CU,ML=9999,END
```

Where:

ID=AL01
Assigns a unique 4-character ID to this trap.

J=*jobname*
This trap applies to job or started task *jobname*. Specify the ALCS job name here.

C=U*xxxx*
User completion code. The trap is activated when there is an abend with any user completion code. C=XXXX activates traps for all user abends. However you can be more specific, for example: C=U0042 activates the trap for a User 42 abend only.

C=S122
System completion code. The trap is activated when the operator cancels ALCS with dump.

A=SVCD
Directs SLIP to take an SVC dump when the trap is activated. (Not included in the command as SVCD is the default action.)

AL=CU
Address space list. Dump the current address space only.

ML=9999
Match limit. Number of times the trap occurs before it is disabled. This is used to limit the number of dumps. For example, to take only one dump of abend U0795, specify C=U0795,ML=1 (ML=1 is the default.)

To inhibit an SVC dump when an abend S0D6 occurs, use:

```
SLIP SET,ID=X0D6,C=S0D6,A=NODUMP,END
```

Where:

C=S0D6
System completion code.

A=NODUMP
Directs SLIP to inhibit dumps when a 0D6 abend occurs.

To trace instruction execution in a monitor installation-wide exit, set the following trap:

```
SLIP SET,IF,ID=USVC,J=jobname,,
     RA=(start,end),
     A=TRACE,TRDATA=(STD,REGS),
     ML=5000,PL=50,END
```

Where:

IF
Trap type is instruction fetch

ID=USVC
Trap identification

J=*jobname*
ALCS job name

RA=(*start,end*)
Range of addresses to trace. Whenever the processor executes instructions in this range, a SLIP trap occurs.

A=TRACE
Action is trace. When the trap takes place, SLIP writes a GTF trace record. (Remember to start GTF).

TRDATA=(STD,REGS)
Trace the standard information plus the contents of the general registers.

ML=5000
Disable the trap after 5000 instructions.

PL=50
Disable the trap if processing takes more than 50% processor utilization.

To find out who is storing X'00000000' at address X'0250A080', use:

```
SLIP SET,SA,ID=GBL1,J=jobname,
         RA=(025A080,025A083),DATA=(0250A080,EQ,00000000),
         A=TRACE,TRDATA=(STD,REGS),
         ML=5000,PL=50,END
```

Where:

SA
Trap type is storage alteration.

RA=(...)
Range of addresses to trace. When storage is altered in that range, a SLIP trap occurs.

DATA=(...)
You can use this operand to distinguish between "good" and "bad" alterations. SLIP ignores alterations that do not match the DATA conditions (in this case, that the storage contents are X'00000000'). This operand is optional.

To check whether the dump data sets are empty use the display command:

```
D D,T
```

Where:

D D,T
Display dump titles. Display the titles of dumps in the SYS1.DUMPxx data sets.

To clear dump data sets, use the DUMPDS commands as follows:

DD CLEAR,DSN=00
Clears SYS1.DUMP00

DD CLEAR, DSN=(00-09)
Clears SYS1.DUMP00 through SYS1.DUMP09

DD CLEAR,DSN=ALL
Clears all dump data sets

Note: These examples work only for ALCS user abends. If you enter

```
CANCEL alcsjob,DUMP
```

no dump is produced, as the AL01 trap is not activated by a system-122 abend. Enter the following SLIP command to set a trap for S122 abends:

```
SLIP SET,ID=AL02,J=jobname,C=122,AL=CU,ML=9999,END
```

or

```
SLIP SET,ID=AL02,JSPGM=DXCMON,C=122,AL=CU,ML=9999,END
```

Interactive problem control system (IPCS)

IPCS provides installations with an interactive facility for diagnosing software failures. IPCS formats and analyzes unformatted dumps and GTF traces to produce reports that you can either view at the terminal or print.

For information on how to use IPCS, see the appropriate *Interactive Problem Control System Guide* or the appropriate MVS manual.

Problem determination in ALCS

This section contains the following subsections:

- Recoup and pool error information
- System error dumps
- Format of system error dumps
- Program driver
- SLC link trace facility
- TCP/IP trace facility
- ALCS test facilities
- Information and error messages

Recoup and pool error information

Recoup (see Chapter 5, “Recoup,” on page 52) detects errors in long-term pool file records. These errors are called **pool chain errors**.

While Recoup is running, ALCS records information about Recoup progress on the ALCS diagnostic file.

When an application program gets, reads, writes, or releases a pool file record, ALCS checks the pool control information that the record contains. In this way, ALCS detects errors such as:

- A record which is read or written after release.
- A record which is released more than once.
- Missing release for a record (lost address).

These are called **pool usage errors**.

When the ALCS monitor detects a pool usage error, it records information about the error on the ALCS diagnostic file.

As with pool usage errors, ALCS records information about pool chain errors on the ALCS diagnostic file.

These include:

- Excessively long chains
- Looping chains
- Multiple references to the same chain -- except where allowed

When application programs use the ALCS release chain service, ALCS can detect errors in the chain of records. ALCS records information about these errors on the ALCS diagnostic file.

The ALCS diagnostic file processor formats and prints information about Recoup progress, pool usage errors, pool chain errors and release chain errors.

Recoup progress

During Recoup, ALCS writes Recoup progress information to the ALCS diagnostic file. The ALCS diagnostic file processor prints this information as follows:

```
hh.mm.ss RECOUP DXC8404I CMD i time_stamp  RECP
group_details
This prime group   group_read_counts
This recoup        recoup_read_counts
Errors             error_counts
```

Where:

hh.mm.ss
Time-of-day (TOD) clock time. This is the MVS local time.

i
ALCS system identifier.

time_stamp
ALCS time stamp. This is the ALCS local time.

group_details
Identifies the Recoup group:

- `Recoup descriptor=`*program_name*
  Program name of the Recoup descriptor program.
- `Group=`*group_name*
  Group name.

group_read_counts
Record read counters for this Recoup prime group:

- `total reads`...*count*
  Total number of records, fixed file, pool file, and general file.
- `pool reads`...*count*
  Number of pool file records.

recoup_read_counts
Record read counters for this Recoup:

- `total reads`...*count*
  Total number of records, fixed file, pool file, and general file.
- `pool reads`...*count*
  Number of pool file records.

error_counts
Error counters for this Recoup:

- `I/O`...*count*
  Total number of I/O errors.
- `ID`...*count*
  Total number of record ID compare errors (ID checks).
- `RCC`...*count*
  Total number of RCC compare errors (RCC checks).
- `F/A`...*count*
  Total number of file address errors (invalid F/A).
- `control-field`...*count*
  Total number of control field errors.

A possible (self-explanatory) error message is:

```
hh.mm.ss RECOUP  DXC8387I CMD i time_stamp RECP
                 Missing descriptor program - Recoup continues
                 MISSING DESCRIPTOR PROGRAM program
```

Pool usage errors

For each pool usage error, the ALCS diagnostic file processor prints the following:

```
hh.mm.ss POOL RECORD USAGE ERROR -- error_description
event_information

   DISPENSE FOR nnnn yyyy.ddd hh.mm.ss
 LAST WRITE FOR nnnn yyyy.ddd hh.mm.ss
CHAIN-CHASE FOR nnnn yyyy.ddd hh.mm.ss
    RELEASE FOR nnnn yyyy.ddd hh.mm.ss
START TIME LAST RECP yyyy.ddd hh.mm.ss
```

Where:

hh.mm.ss
Time-of-day (TOD) clock time of the error.

error_description
Type of error.

The *error_description* for each type are described in:

- “Long-term pool errors” on page 356
- “Short-term pool errors” on page 357

nnnn
The name of the application program or (for chain-chase) of the Recoup descriptor program.

yyyy.ddd hh.mm.ss
Julian date and time of the activity, using the time-of-day (TOD) clock time.

Long-term pool errors

- RELEASE OF UNDISPENSED RECORD
  The application program released a long-term pool file record that was never dispensed. ALCS sets this record as dispensed and released.
- MULTIPLE RELEASE OF RECORD
  The application program released a long-term pool file record that was already released. Note that either this release, or the previous release, can be in error. ALCS ignores this release.
- RELEASE OF LOST ADDRESS
  The application program released a long-term pool file record that was a lost address. A long-term pool file record becomes a lost address after the following sequence of events:

  1. An application program gets (dispenses) the record.
  2. No application program releases the record.
  3. Recoup does not find the record.

  This can be due to an error or omission in the Recoup descriptors. ALCS processes this release normally.
- LOST ADDRESS FOUND BY DISPENSE ROUTINE
  ALCS attempted to dispense a long-term pool file record that was a lost address. This can be due to an error or omission in the Recoup descriptors. ALCS releases the record; it does not dispense the record, unless the long-term pool dispense installation-wide exit routine alters the standard ALCS options.

- SYSTEM DISPENSE OF RECORD ON WRITING
  The application program wrote a long-term pool file record that was never dispensed. ALCS dispenses the record.
- SYSTEM SHIFT RELEASE OF RECORD ON WRITING
  The application program wrote a long-term pool file record that was already released. Note that either this write, or the previous release, can be in error. ALCS shifts the release time; that is, it resets the release time for the record to the time of the write.
- WRITE OF LOST ADDRESS RECORD
  The application program wrote a long-term pool file record that was a lost address. This can be due to an error or omission in the Recoup descriptors. ALCS writes the record in the normal way.
- READ OF UNDISPENSED RECORD
  The application program read a long-term pool file record that was never dispensed. ALCS reads the record in the normal way.
- SYSTEM DISPENSE OF RECORD ON READING DURING RECOUP
  Recoup read a long-term pool file record that was never dispensed. ALCS dispenses the record.
- READ OF PREVIOUSLY RELEASED RECORD
  The application program read a long-term pool file record that was already released. ALCS reads the record in the normal way.
- READ OF ERRONEOUSLY AVAILABLE RECORD DURING RECOUP
  Recoup read a long-term pool file record that was already released. ALCS clears the release time for the record.
- READ OF LOST ADDRESS RECORD
  The application program read a long-term pool file record that was a lost address. This can be due to an error or omission in the Recoup descriptors. ALCS reads the record in the normal way.
- READ OF LOST ADDRESS RECORD DURING RECOUP
  Recoup read a long-term pool file record that was a lost address. This can be due to an error or omission in the Recoup descriptors. The record is no longer a lost address.
- AMBIGUOUS RECORD ID REQUESTED
  The application program attempted to get (dispense) a long-term pool file record, but the record ID is ambiguous (duplicate). The application program specifies a record ID that is defined for more than one pool file type. But the application does not specify enough information (long- or short-term, size, or qualifier) to identify the type that it requires.
- UNDEFINED RECORD ID REQUESTED
  The application program attempted to get (dispense) a long-term pool file record, but either:
  - The record ID is undefined in the ALCS generation. The application program specifies a record ID that is not defined.
  - The record ID is undefined for the pool type. The application program specifies a record ID together with long- or short-term, or size information. But the record ID is not defined for any pool type that satisfies the request.
- INVALID FILE ADDRESS ON RELEASE RING
  The application program released a valid file address but there is no corresponding record on the database, so no record can be released.
- CLASS MISMATCH FOR LT POOL RELEASE
  The application program released a valid file address but the record to which it refers was filed using a different file address, that is using a different record class. A record can only be released if the file address that has been used in the release macro is the same address that was used to file the record.

Short-term pool errors

- WRITE OF TIMED OUT ST POOL RECORD
  The application program wrote a short-term pool record that was timed out.

- WRITE OF UNDISPENSED ST POOL RECORD
  The application wrote a short-term pool record that was never dispensed.
- WRITE OF PREVIOUSLY RELEASED ST POOL RECORD
  The application wrote a short-term pool record that was previously released.
- DISPENSE OF TIMED OUT ST POOL RECORD
  ALCS dispensed a short-term pool record that was a lost address.
- RELEASE OF TIMED OUT ST POOL RECORD
  The application program released a short-term pool record that was timed out. ALCS ignores this release.
- RELEASE OF UNDISPENSED ST POOL RECORD
  The application program released a short-term pool record that was never dispensed. ALCS ignores this release.
- MULTIPLE RELEASE OF ST POOL RECORD
  The application program released a short-term pool record that was already released. Note that either this release or the previous release can be in error. ALCS ignores this release.
- READ OF TIMED OUT ST POOL RECORD
  The application program read a short-term pool record that had been timed out. ALCS reads the record in the usual way.
- READ OF UNDISPENSED ST POOL RECORD
  The application program read a short-term pool record that was never dispensed. ALCS reads the record in the usual way.
- READ OF PREVIOUSLY RELEASED ST POOL RECORD
  The application program read a short-term pool record that was already released. ALCS reads the record in the normal way.

event_information
Information about the event (the monitor-request macro that the application program issued) when ALCS detected the error. This information includes the following:

- F/A=*file_address*
  File address of the record, 8 hexadecimal digits.
- PROGRAM=*program_name*
  The name of the application program that issued the monitor-request macro.
- OFFSET=*macro_offset*
  The offset of the monitor-request macro in the application program.
- MACRO=*macro_name*
  The name of the monitor-request macro.
- ID=*record_id*
  Record ID of the record, 2 characters or 4 hexadecimal digits. For write operations, this is the record ID before the write.
- RCC=*record_code_check*
  Record code check of the record, 2 hexadecimal digits. For write operations, this is the record code check before the write.
- NEW-ID=*record_id*
  For write operations, the record ID after the write.
- NEW-RCC=*record_code_check*
  For write operations, the record code check after the write.

When short-term file pool event logging is enabled, the most recent event details for the record are included as follows:

```
yyyy.ddd hh.mm.ss F/A=file_address Dispensed by ...prog
yyyy.ddd hh.mm.ss F/A=file_address Filed by .......prog
```

```
yyyy.ddd hh.mm.ss F/A=file_address Found by .......prog
yyyy.ddd hh.mm.ss F/A=file_address Released by ....prog
```

This information can be useful for analyzing errors in short-term pool handling.

Fixed-file usage errors

For each fixed-file usage error, the ALCS diagnostic file processor prints the following:

```
hh.mm.ss FIXED FILE RECORD USAGE ERROR -- error_description
event_information
```

- ACCESS OF DELETED RECORD
  The application program read or wrote a fixed-file record, which had previously been marked as deleted by loading a new DASD configuration. The *access* is a normal access to the application program. However once the deleted records are purged the application will receive an invalid file address return.

  The *event_information* is similar to that described in "Short-term pool errors" on page 357.

Pool chain errors

Pool chain errors are normally detected by Recoup (see Chapter 5, "Recoup," on page 52).

Pool chain errors can show that data is lost or damaged, but it is not possible to correct the errors automatically. Be sure to investigate and correct them as soon as possible, using, for example, the ZGAFA, ZDFIL, and ZAFIL commands. These are described in:

"ZGAFA -- Get an available pool file address" on page 201.
"ZDFIL -- Display contents of a DASD record" on page 177.
"ZAFIL -- Alter DASD record" on page 85.

For each pool chain error, the ALCS diagnostic file processor prints:

```
hh.mm.ss POOL CHAIN ERROR -- error_description
CHAIN-FROM record_details CHAIN-TO record_details
```

or, if the error occurs for a record in a prime group, the ALCS diagnostic file processor prints:

```
hh.mm.ss RECORD READ ERROR -- error_description
prime_group_record_details
```

Where:

hh.mm.ss
Time-of-day (TOD) clock time when Recoup detects the error.

error_description
Type of error, one of:

- INVALID F/A
  Recoup cannot read the chain-to record because the file address is invalid.
- I/O ERROR
  Recoup cannot read the chain-to or prime group record because of an I/O error.
- CONTROL ERROR
  Recoup cannot read the chain-to record because:
  - There is an error in the Recoup descriptor, or
  - There are fields in the record that show where (in the record) there is a file address. The field contents are invalid, so Recoup cannot locate the file address.

- ID CHECK -- ID IS *record_id*
  The record ID in the chain-to or prime group record is not the expected record ID. *record_id* is the record ID in the chain-to or prime group record; 2 characters or 4 hexadecimal digits.
- RCC CHECK -- RCC IS *record_code_check*
  The RCC in the chain-to or prime group record is not the expected RCC.

  record-code-check is the record code check in the chain-to or prime group record; 2 hexadecimal digits.

CHAIN-FROM *record_details*
Information about the chain-from record:

- F/A=*file_address*
  File address of the chain-from record, 8 hexadecimal digits.
- ID=*record_id*
  Record ID of the chain-from record, 2 characters or 4 hexadecimal digits.
- RCC=*record_code_check*
  RCC of the chain-from record, 2 hexadecimal digits.
- CTL=*control_byte*
  Control byte of the chain-from record, 2 hexadecimal digits. The control byte is byte 3 of the record; that is, the byte that follows the record code check.

CHAIN-TO*record_details* or *prime_group_record_details*
Information about the chain-to or prime group record:

- F/A=*file_address*
  File address of the chain-to or prime group record; 8 hexadecimal digits.
- EXPECTED-ID=*record_id*
  Expected record ID of the chain-to or prime group record; 2 characters or 4 hexadecimal digits.
- EXPECTED-RCC=*record_code_check*
  Expected RCC of the chain-to or prime group record; 2 hexadecimal digits.
- CTL=*control_byte*
  Control byte of the chain-to or prime group record; 2 hexadecimal digits.

Release chain pool errors

When an application issues the RLCHA monitor-request macro, the program ALCS releases a chain of records. In the process of releasing the records, ALCS checks for various errors in the chain of records. If an error is found, information about the error is written to the diagnostic file and no further records are released. The ALCS diagnostic file processor formats and prints the information recorded on the diagnostic file for these errors.

For each error detected, the ALCS diagnostic file processor prints:

```
hh.mm.ss.RLCH ff message FAE-fae HOC-hoc
PIC-pic ID-id RCC-rcc PROG-program
```

Where:

message
The type of error. This is one of the following:

- INVALID MESSAGE NUMBER
  This error should not occur; if it does, inform your IBM programming support representative.
- INVALID FILE ADDRESS
  The file address *fae* is invalid.

- RELEASE OF A NON-POOL TYPE RECORD
  The file address *fae* is not the file address of a long- or short-term pool file record.
- RELEASE OF A NON-DISPENSED LT RECORD
  The file address *fae* is the file address of a long-term pool file record, but this pool file record has not been dispensed.
- RELEASE OF A RELEASED LT RECORD
  The file address *fae* is the file address of a long-term pool file record that has already been released.
- LT RECORD CHAINED FROM A ST RECORD
  The file address *fae* is the file address of a long-term pool file record that is chained from a short-term pool file record whose file address is *pic*. As short-term pool file records have a life of only minutes at the most, this type of chaining is considered to be an error.
- CIRCULAR CHAIN IN ST CHAIN
  The file address *fae* is the file address of a short-term pool file record whose forward chain contains the file address of a short-term pool file record that has been released earlier in the chain.
- TOO MANY ST RECORDS IN CHAIN
  There are more than 255 records in a chain of short-term pool file records.
- ID COMPARE CHECK
  The record whose file address is *fae* has a record ID that is different from the record ID passed in the RLCHA parameter list.
- RCC COMPARE CHECK
  The record whose file address is *fae* has an RCC that is different from the RCC passed to the RLCHA parameter list.

FAE-*fae*
File address in error. File address of the record at which the error was detected. It is 8 hexadecimal digits.

HOC-*hoc*
Head of chain. File address of the first record in the chain. It is 8 hexadecimal digits.

PIC-*pic*
Previous in chain. File address of the record previous to the record at which the error was detected. It is 8 hexadecimal digits. If the record in error is the first record in the chain, this contains '********'.

ID-*id*
The record ID in the RLCHA parameter list. It is either 2 alphanumeric characters or 4 hexadecimal characters.

RCC-*rcc*
The RCC field in the RLCHA parameter list. It is 2 hexadecimal characters.

PROG-*program*
The name of the program which issued the RLCHA monitor-request macro.

System error dumps

ALCS produces system error dumps to help with problem determination. ALCS does not print system error dumps. Instead, it writes each dump to the ALCS diagnostic file. Use the ALCS diagnostic file processor to print the dump. See “Running the ALCS diagnostic file processor” on page 26 for further information.

ALCS normally (but not always) produces a system error dump when one of the following occurs:

Control error
The online monitor program detects an error condition that requires analysis. The dump is called a control (CTL) dump.

Operational error
An application program detects an error condition that requires analysis; the application issues SYSRA, SERRC, or snapc to request a system error dump. The dump is called an operational (OPR) dump.

Manual dump
The ZDUMP command, described in "ZDUMP -- Request a system error dump" on page 200, requests a system error dump. The dump is called a manual dump. It is a type of control dump.

Dump sequence number and system error number

Each system error dump has a dump sequence number and a system error code.

The dump sequence number is a 6-digit decimal number that increases by 1 for each dump.

The system error code is a 6-digit hexadecimal number that indicates the type of error condition. System error codes that start with 000 are reserved for ALCS to use. The *ALCS Messages and Codes* lists them. Application programs can use other system error codes.

System error message

When ALCS produces a system error dump, it normally (but not always) prints a message at RO CRAS. This is called the system error dump message, or the system error message. It includes the dump sequence number and the system error code. It also identifies the first ALCS diagnostic file that contains the system error dump.

System error options

ALCS supports user-specified options that specify, for example:

- Whether or not ALCS produces a system error dump in certain situations.
- What storage areas ALCS includes in system error dumps.
- Whether or not ALCS prints a system error message.

These are called the system error options.

Specify system error options as ALCS generation parameters (see *ALCS Installation and Customization* for an explanation of the SCTGEN macro). Use the ALCS commands ZDSER to display and ZASER to alter the system error options. This is described in "ZDSER -- Display system error options" on page 196.

Regardless of the setting of the system error options:

- ALCS suppresses or truncates system error dumps if the ALCS diagnostic file is not available.
- ALCS can suppress the system error message if the ALCS load is too high.

Catastrophic system errors

In most cases, ALCS continues processing after it produces a system error dump. Depending on the type of error, ALCS sometimes cancels (exits) the entry that is active when the error occurs. However, some errors are so severe that ALCS cannot continue. In this case, ALCS abnormally terminates (abends). These are called **catastrophic system errors**.

If a catastrophic system error occurs, ALCS attempts to produce a system error dump. Whether or not it is able to produce a system error dump, ALCS then abnormally terminates. If the ALCS job or procedure includes a suitable data definition (DD) job control statement, MVS produces an abend dump. "Initiating ALCS" on page 5 describes how to run the ALCS monitor.

Format of system error dumps

This section describes the format of ALCS system error dumps. The ALCS diagnostic file processor prints system error dumps in several sections. These sections are described below. However, the ALCS diagnostic file processor does not always print all the sections. The sections that it prints depend on, for example:

- *The type of error.* For example, control dumps can contain different information from operational dumps. Usually they contain more information.

- *Conditions at the time ALCS detects the error.* For example, if there is an active entry when ALCS detects the error, the dump includes information about the active entry (the contents of the ECB and so on). If there is no active entry, the dump does not include this information.
- *System error options in force at the time ALCS detects the error.* These depend on the ALCS generation (see *ALCS Installation and Customization*). The ZASER command, described in "ZASER -- Alter system error options" on page 94, can change the system error options.
- *ALCS diagnostic file processor control statements.* These control statements can request that the ALCS diagnostic file processor does not print some dump sections. "Diagnostic file processor control statements" on page 27 describes how to use them.

Standard system error dump format

The ALCS diagnostic file processor uses a standard format to print most sections of a system error dump. This standard format shows the contents of up to 32 bytes on each print line. Figure 12 on page 363 shows an example of this standard format.

```
02EEFD60  000 U W 0B00 0005C200      008 0C000000 17817000 ...
02EEFD80  020 13050000 178B6800      028 13050000 17A06800 ...
02EEFDA0  040 130A0000 17A45804      048 13090000 17A55800 ...
********
```

Figure 12. Standard system error dump format (with offsets)

The first (leftmost) column is the storage address (8 hexadecimal digits). It is the address of the first byte on the line. The ALCS diagnostic file processor does not print lines if the entire 32 bytes of storage contain binary zeros. Instead it prints eight asterisks (********) in the address column. This indicates that ALCS has suppressed one or more lines of zeros.

The rest of the line consists of up to four groups of columns. Each group contains three columns that show the displacement and contents of 8 bytes of storage (that is, the displacement from the start of this storage area (3 or 4 hexadecimal digits) followed by the contents of the 8 bytes). The contents are in hexadecimal or character notation. The ALCS diagnostic file processor uses character notation if the storage area can contain character data and if two or more consecutive bytes contain printable characters, as specified in the TRANSLATE control statement. See "Diagnostic file processor control statements" on page 27.

The ALCS diagnostic file processor does not print offsets greater than hexadecimal X'7FFF'. Instead, it prints blanks in the offset columns.

Figure 12 on page 363 shows the contents of an area of storage that starts at address hexadecimal X'2EEFD60'. At displacement 0, 2 bytes contain hexadecimal X'E4E6' (EBCDIC characters UW). The next 2 bytes contain 0B00. The byte at displacement 48 (address 2EEFDA8) contains 13, and so on. Bytes from address 2EEFDC0 onwards contain binary zeros.

Note: The byte at displacement 6 contains hexadecimal X'C2' (the EBCDIC character B). The ALCS diagnostic file processor uses hexadecimal notation for this, not character notation, because the byte is not part of a string of two or more consecutive printable characters.

For some storage areas, the ALCS diagnostic file processor prints character "tags" instead of hexadecimal offsets. Figure 13 on page 364 shows an example of this. It shows the first few lines of an ECB dump. The ECB starts at address hexadecimal X'2EED4A0'. The first two fullwords are CE1CHW and CE1BAD. Both are reserved fields. Next is the ECB work area, EBW000, and so on.

```
02EED4A0  CHW 00000000 0003ABD0 BAD
02EED4A8 W000 30R O M  001D0030     W008 00000000 70010000 ...
02EED4C8 W032 008D002C 00050000     W040 00400003 00000000 ...
```

Figure 13. Standard system error dump format (with tags)

Notice that the first line of the ECB dump contains only 8 bytes (not 32 bytes). The ALCS diagnostic file processor sometimes prints storage contents in this way, so that it can start a group of related fields on a new line.

System error dump header

The ALCS diagnostic file processor prints a header at the start of each system error dump. This header is similar to the system error message that ALCS prints at RO CRAS when it produces a dump. Figure 14 on page 364 shows examples of the system error dump header.

```
hh.mm.ss SE-000101 OPR-F11CC2 PROG-XERA OFFSET-104
VOLUME=volser DSNAME=dsname ALCS - yyyy.ddd hh.mm LOCAL, yyyy.ddd hh.mm GMT
                            MVS  - yyyy.ddd hh.mm LOCAL, yyyy.ddd hh.mm GMT
hh.mm.ss SE-NODUMP OPR-F11CC2 PROG-XERA OFFSET-104
VOLUME=volser DSNAME=dsname ALCS - yyyy.ddd hh.mm LOCAL, yyyy.ddd hh.mm GMT
                            MVS  - yyyy.ddd hh.mm LOCAL, yyyy.ddd hh.mm GMT
hh.mm.ss SE-000102 CTL-000000 PSW-077D1000 800237E6         MSG='ZDUMP'
VOLUME=volser DSNAME=dsname ALCS - yyyy.ddd hh.mm LOCAL, yyyy.ddd hh.mm GMT
                            MVS  - yyyy.ddd hh.mm LOCAL, yyyy.ddd hh.mm GMT
```

Figure 14. Examples of system error dump header

The system error dump header contains the following information (different system error dump headers can contain different information, depending on the error):

ALCS - *yyyy.ddd hh.mm* LOCAL, *yyyy.ddd hh.mm* GMT
The ALCS year, day of year, and time (local and GMT) at the time of the dump.

MVS - *yyyy.ddd hh.mm* LOCAL, *yyyy.ddd hh.mm* GMT
The MVS year, day of year, and time (local and GMT) at the time of the dump.

SE-*number*
Dump sequence number, a 6-digit decimal number. SE-NODUMP means that there is no dump for the error; usually this is because the error duplicates a previous dump.

CTL-*code* or OPR-*code*
System error code, a 6-digit hexadecimal number. CTL indicates that the error is a control system error. That is, an error that the online monitor detects. OPR indicates that the error is an operational system error. That is, an error that an application program detects. The *ALCS Messages and Codes* lists the system error codes that ALCS uses.

PROG-*name* or PROG-*name1*/*name2*
Name of the application program or ALCS monitor CSECT (*name*) executing at the time of the error. The dump header contains this only if ALCS can determine that the error is in a particular application program or ALCS monitor CSECT. If a TPFDF program is executing at the time of the error, this is in the format *name1*/*name2* This provides the name of the TPFDF program (*name1*) executing at the time of error and the application program (*name2*) which invoked the TPFDF program.

OFF-*listing_address*
Offset of error within the ALCS monitor CSECT, a hexadecimal number. This address corresponds to the address (LOC) in the assembler listing of the ALCS monitor CSECT. The dump header only contains this if ALCS can determine that the error is in a particular monitor CSECT.

OFFSET-*listing_address*
Offset of error within the application program, a hexadecimal number. This address corresponds to the address (LOC) in the assembler listing of the application program. The dump header only contains this if ALCS can determine that the error is in a particular application program.

AT-*address*
Address of error in memory, a hexadecimal number. The dump header contains this only if ALCS can determine that a particular application program was executing at the time of the error, but the failing instruction is outside the program.

PSW-*psw*
Corrected program status word (PSW) at the time of the error. "Corrected" means that the instruction address in the PSW is adjusted to point to the failing instruction, not to the next sequential instruction. The dump header contains this if ALCS cannot determine that the error is in a particular application program.

CRN-*crn*
The CRN of the originating communication resource. The dump header contains this only if ALCS can determine that the error is in an entry that originates from a particular communication resource.

CRI-*cri*
The CRI of the originating communication resource. The dump header contains this only if ALCS can determine that the error is in an entry that has an associated CRI, but the CRI does not match any known communication resource.

MSG='*text*'
Explanatory message associated with the error. The dump header contains this only if there is an explanatory message.

VOLUME-*volser*
Volume serial of ALCS diagnostic file data set that contains the system error dump. (Note that the system error dump can overflow to other MVS data sets.)

DSNAME-*dsname*
Data set that contains the system error dump. (Note that the system error dump can overflow to other MVS data sets.)

If there is no dump for the error (SE-NODUMP), the system error dump header does not include the ALCS diagnostic file processor data set name and volume serial number.

If the system error is a program interruption that is not a SERRC, the system error dump header includes an extra line showing the type of program interruption.

If the system error is an MVS abend that is not a program interruption, the system error dump header includes an extra line showing the system completion code.

General and floating-point registers

This dump shows the contents of the general and floating-point registers at the time of the error (that is, when the online monitor detects the error, or when the application program issues SYSRA or SERRC). The dump also shows the corrected program status word (PSW) at the time of the error. "Corrected" means that the instruction address in the PSW is adjusted to point to the failing instruction[4], not to the next sequential instruction.

The register contents and the PSW are in hexadecimal notation. Figure 15 on page 366 shows an example.

[4] An "imprecise interrupt" can occur, which does not point accurately to the failing instruction, but this is extremely rare.

```
GENERAL PURPOSE REGISTERS
        RAC 00001313 7F705970 RG1   RGA FF5FDEAC 7F70D000 ...
        RAP 7F5FDDA0 02F414A0 REB   RLA 8003C5EC 0003BF20 ...
        PSW 077D0000 82F41548
```

Figure 15. Example of a general register dump

Area addressed by program status word

This dump shows the contents of 128 or 4096 bytes of storage that include the storage that the PSW addresses, if that storage is accessible to ALCS. Figure 16 on page 366 shows an example. In the example, the PSW is hexadecimal X'077D000082F41540'. This PSW instruction address points to the character string 'SE' (hexadecimal X'E2C5') (a SERRC monitor-request macro).

```
AREA ADDRESSED BY PSW
02F41518      T M 0000 00000000          X Q 0000 000B3240 ...
02F45138      X Z 0068 00003440          S E 61F1 131307F2 ...
********
```

Figure 16. Example of an area addressed by PSW dump

Areas addressed by general registers

This dump shows the contents of 128 or 4096 bytes of storage that include the storage addressed by each of the general registers at the time of the error, if that storage is accessible to ALCS. For example:

```
AREA ADDRESSED BY RG1
7F705790      00000000 00000000          X W 64A0 C V S N  ...
********

AREA ADDRESSED BY RGA
7F5FDEA8      4D2090A8 58E09130          5820E014 1BFF41FF ...
7F5FDEC8      81F04AE0 83DE47F0          81160CAA 00503000 ...
⋮
```

Figure 17. Example of areas addressed by general registers dump

Storage unit block list descriptor

This dump shows the contents of the storage unit block list descriptor for the entry active at the time of the error. Figure 18 on page 366 shows an example.

```
BLOCK LIST DESCRIPTOR -- PRIME SU -- TIME 12.27.28.2
00078560  PCH 00077D20 02F41000 PBA  PPI 0003C580 C0000000 ...
00078580  FOP 00000000 00000000      LOP 00000000 40000000 ...
```

⋮

Figure 18. Example of a storage unit block list descriptor dump

Tags identify fields in the storage unit block list descriptor. They relate to labels that the LS0LS macro generates in the LS0LS DSECT. For example, PCH is the tag for the field LS0PCH, FOP is the tag for field LS1FOP, and so on.

If the entry is using more than one storage unit, there is a storage unit block list descriptor for each storage unit.

hh.mm.ss.t (hours, minutes, seconds, and tenths of a second) is the time (TOD clock) that ALCS stores in the descriptor when it creates the ECB. For entries that process VTAM input messages, ALCS creates the ECB ready to receive the message; later (when the message arrives) ALCS replaces the TOD clock in the descriptor with the time that the message arrives.

If an entry is using more than one storage unit, the first appears with the heading:

```
     BLOCK LIST DESCRIPTOR -- PRIME SU -- TIME hh.mm.ss.t
```

the rest appear with the heading:

```
     BLOCK LIST DESCRIPTOR -- OVERFLOW SU -- TIME hh.mm.ss.t
```

with the same tags. The time is the TOD clock that ALCS stores in the descriptor when it dispenses the storage unit.

Entry control block descriptor

This dump shows the contents of the ECB descriptor for the entry active at the time of the error. Figure 19 on page 367 shows an example.

```
ENTRY CONTROL BLOCK DESCRIPTOR
02F40000  BDA 00078560 00000000  PIA STA 00000000 00000000 ...
02F40020 ACEE 00000000 00000000      FEX 00000000 80000000 ...
```

⋮

Figure 19. Example of an entry control block descriptor dump

Tags identify fields in the ECB descriptor. They relate to the CE3 labels that the DXCECBD macro generates in the DXCECBD DSECT. For example, BDA is the tag for the field CE3BDA, and so on.

If the entry is using more than one storage unit, there is an ECB descriptor only for the first storage unit. The ECB descriptor is positioned just before the first storage unit.

Data event control block (DECB) descriptor

This dump shows the contents of the DECB descriptor for the entry active at the time of the error. Figure 20 on page 367 shows an example.

```
DECB DESCRIPTOR
0F71E000 DESC 0F705768 00000000           00000000 0032002E ...
0F71E020  020 DECBDECB 00010000       028 00000000 00000000 ...
```

⋮

Figure 20. Example of a DECB descriptor dump

Tags identify fields in the DECB descriptor. They relate to the CE5 labels that the DXCECBD macro generates in the DXCECBD DSECT. For example, DESC is the tag for the field CE5DESC, and so on.

There can be more than one DECB descriptor for the entry. A DECB descriptor contains the DECB header and the system areas for several DECBs. Each DECB descriptor is positioned just before the storage unit containing the associated DECB application areas.

Data event control block (DECB) application areas

This dump shows the contents of DECB application areas for the entry active at the time of the error. Figure 21 on page 368 shows an example.

```
DECB APPLICATION AREAS
0F71F020 CNAM C1D5D5F0 F1404040          40404040 40404040 ...
0F71F040 CFRW 00000000 00000000          030D58F3 00000000 ...
0F71F060      00000000 00000000          00000000 00000000 ...
```

⋮

Figure 21. Example of a DECB application areas dump

Tags identify fields in the DECB application areas. They relate to labels that the IDECB macro generates in the IDECB DSECT. For example, CNAM is the tag for the field IDECNAM, and so on.

Entry control block prefix

This dump shows the contents of the ECB prefix for the entry active at the time of the error. Figure 22 on page 368 shows an example.

```
ENTRY CONTROL BLOCK PREFIX
02F410A0  PFX 00000000 00000000          00000000 00000000 ...
02F410C0      00000000 00000000          00000000 00000000 ...
```

⋮

Figure 22. Example of an entry control block prefix dump

Tags identify fields in the ECB prefix. They relate to labels that the EB0EB macro generates in the EB0EB DSECT. For example, PFX is the tag for the field CE2PFX and so on.

Entry macro trace block

The entry macro trace block is formatted to show the monitor-request macros that the entry issues before the error. The monitor-request macros are in time sequence, the most recent last. Figure 23 on page 368 shows an example.

```
ENTRY MACRO TRACE DATA
PROG  MACRO  ADDR  REPEAT   PROG  MACRO  ADDR  REPEAT ...
XERA  GETCC  0034           XERA  FACEC  004E         ...
XERA  FILNC  00EE           XERA  WAITC  00F4         ...
```

⋮

Figure 23. Example of an entry macro trace block dump

The macro trace block dump is in four groups of columns. Each group contains information about one monitor-request macro under the following headings:

PROG
Name of the program that issues the monitor-request macro.

MACRO
Name of the monitor-request macro.

ADDR
Hexadecimal offset (listing address) of the monitor-request macro in the program.

REPEAT
Repeat count if the program issues the monitor-request macro repeatedly.

In the example shown in Figure 23 on page 368, program XERA issues the GETCC monitor-request macro at displacement hexadecimal X'034'. It then issues FACEC (ENTRC FACE) hexadecimal X'04E', and so on. Note that the macro request type, not the macro name, appears in the dump. For most macros, the macro request type and the macro name are the same.

Entry control block

This dump shows the contents of the ECB for the entry active at the time of the error. Figure 24 on page 369 shows an example.

```
ENTRY CONTROL BLOCK -- CREATED BY CRETC -- LAST MACRO ...
02F415A0  CHW 00000000 0003C580 BAD
02F415A8 W000       00 02F435FE     W008 0001001E 00000000 ...
02F415C8 W032 00000190 01970197     W040 00000000 02F4255C ...
```

⋮

Figure 24. Example of an entry control block dump

Tags identify fields in the ECB. They relate to labels that the EB0EB macro generates in the EB0EB DSECT. For example, CHW is the tag for the field CE1CHW, W000 is the tag for field EBW000, and so on.

The contents of the register save area in the ECB (dump tags RDA, RDB, RAC, RG1, ...) can be useful. They show the register contents at the time that the entry issued the last monitor-request macro before the error was detected.

Communication table item

This dump shows the contents of the communication table item for the originating communication resource. This is for the CRI in field CE3CRI in the ECB descriptor for the entry active at the time of the error. Figure 25 on page 369 shows an example.

```
COMMUNICATION TABLE ITEM -- CRI - 0200ED
7F259500  CRB D7E8C5E2 C1D9C5F3          CRI 000200ED 00000000 ...
7F259500  SW1 20C20012 00000000 SW2      TOD AB40DF64 F13E6904 ...
7F259538  038 E8000000 00000000          040 00000000 00000000 ...
7F259558  058 00000000 00500780          060 20080000 00000000 ...
7F259578  078 00000000 00000000          080 00000000 18500B00 ...
********
```

⋮

Figure 25. Example of an entry control block dump

Tags identify fields in the communication table item. They relate to labels that the C00RE macro generates in the CO0RE DSECT. For example CRN is the tag for the field RECOCRN, CRI is the tag for the field RECOCRI, and so on.

Blocks associated with entry

This dump shows the contents of storage blocks associated with the entry active at the time of the error. These include:

- Attached storage blocks
- Automatic (ALASC) storage blocks
- Detached block control table
- Detached (DETAC) storage blocks
- TPFDF stack blocks
- Detached TPFDF storage blocks
- HLL control blocks
- Local save stack blocks

Figure 26 on page 370 shows an example of an attached storage block.

```
STORAGE BLOCK -- LEVEL 3
02F43150  000 X T 8480 C B X N       008 00000000 00000000 ...
********
```

⋮

Figure 26. Example of an attached storage block dump

The heading shows the level where the block is attached.

Programs associated with entry

This dump shows the application programs associated with the entry that was active at the time of the error.

The dump shows these programs in order of execution. For example, if program PRG1 calls (ENTRC) program PRG2 and program PRG2 then calls (ENTRC) program PRG3, they appear in the dump in the order PRG1, PRG2, PRG3. A program that issues ENTNC or ENTDC is no longer associated with the entry; it does not appear in the dump.

Figure 27 on page 370 shows an example of an application program.

```
PROGRAM XERA00
7F5FDD88  LNG 00000400 0 0 0040 VSN  SHR FFFFFFFF FFFFFFFF ...
7F5FDDA8  020 95019135 4780801C      028 S E 61F1 1F900CAA ...
```

⋮

Figure 27. Example of an application program dump

The heading shows the program name and the version number. Figure 27 on page 370 shows program XERA version 00.

The first line has tags that identify fields in the program header, as follows:

LNG
Length (bytes) of application program, low-order halfword.

VSN
Program version number, 2 characters.

SHR
Shared global resource indicators, 32 bits.

XCL
Exclusive global resource indicators, 32 bits.

BSE
Program base (general register 8 (RAP) points to this field).

NAM
Program name, 4 characters.

User storage list items

These are the storage areas (if any) that the program requested with the SLIST parameter of the SERRC macro. They appear in standard system error dump format, with offsets.

MVS system diagnostic work area

This dump shows the contents of the MVS system diagnostic work area (SDWA) at the time of the error.

Figure 28 on page 371 shows an example.

```
OS SYSTEM DIAGNOSTIC WORK AREA
BASE AREA
0098B938 PARM 00000000 040C1000 ABCC CTL1 FF750001 00000000 ...
0098B958 GR02 FF5FDEAC 7F70D000 GR03 GR04 7F705E28 02F44B5A ...
```

⋮

Figure 28. Example of a system diagnostic work area dump

See the appropriate *MVS Data Areas* for a description of the format and contents of the SDWA. Tags relate to the field descriptions in that document. For example, PARM is the tag for the field SDWAPARM, and so on.

If general register 10 (RLA) points to an online monitor CSECT at the time of the error, ALCS saves the CSECT name in SDWACSCT (tag CSCT).

Remember that the SYSRA and SERRC monitor-request macros provoke a program interruption, specification exception. MVS records this as system completion code 0C6 in SDWAABCC (tag ABCC).

System macro trace block

This dump shows the system macro trace block formatted to show the monitor-request macros that all entries issue before the error. The monitor-request macros are in time sequence, with the most recent last. Figure 29 on page 372 shows an example.

```
SYSTEM MACRO TRACE DATA
ECB       PROG  MACRO  ADDR   ECB       PROG  MACRO  ADDR  ...
02E514A0  CGTD  GTFCC  0032   02E514A0  CVIA  SENDA  01C8  ...
02E594A0  SLCI  CXFRC  0E02   02E594A0  NUC   ENTDC  1D7E  ...
```

⋮

Figure 29. Example of a system macro trace block dump

See also "System macro trace block" on page 371.

The system macro trace block dump is in four groups of four columns each. Each group contains information about one monitor-request macro under the following headings:

ECB
Address of the ECB of the entry that issues the monitor-request macro.

PROG
Name of the program that issues the monitor-request macro.

MACRO
Name of the monitor-request macro.

ADDR
Hexadecimal offset (listing address) of the monitor-request macro in the program.

In the example in Figure 29 on page 372, the entry with ECB at address hexadecimal X'2E514A0' issues GTFCC at displacement hexadecimal X'032' in program CGTD. It then issues SENDA (SENDC Dx,A) at X'1C8' in CVIA, and so on.

Later, the online monitor SLC input routine creates a new entry. The macro trace block shows this as CXFRC (create new entry) in SLCI (the online monitor SLC input routine). The monitor then passes control to an application program to process the new entry. The macro trace block shows this as ENTDC in NUC (the online monitor nucleus).

Note: The macro request type, not the macro name, appears in the dump. For most macros, the macro request type and the macro name are the same. Only authorized monitor-request macros appear in the system macro trace.

Monitor keypoint record

This dump shows the contents of the monitor keypoint record (CTKB). Figure 30 on page 372 shows an example.

```
MONITOR KEYPOINT - CTKB
7F7313B0  BID C K 0260 V F B    PRG      00000002 FFC40000  ...
7F7313D0      00000000 000003B7 CLS      40404040 40000000  ...
7F7313F0      00000000 00000000 FPC  TMR A4940610 A4940610  ...
```

⋮

Figure 30. Example of a monitor keypoint record (CTKB) dump

Tags identify fields in CTKB. They relate to labels that the CK9KC macro generates in the CK9KC DSECT. For example, BID is the tag for the field CK9BID, and so on.

Resource hold table

This dump shows the contents of the resource hold table. Figure 31 on page 373 shows an example.

```
RESOURCE HOLD TABLE
0007E650  NBR 00000033 00000000 LKW   TRC 00000000 00000000  ...
********
0007E850  FCH 00000000 00000000       LCH 0007E850 00000000  ...
0007E870  TYP 00020088 00008000 ERR   MSK 00020000 003D0000  ...
********
```

Figure 31. Example of a resource hold table dump

Tags identify fields in the resource hold table. They relate to labels that the CH0CH macro generates in the CH0CH DSECT. For example, NBR is the tag for the field CH0NBR, and so on.

CRET table

This dump shows the contents of CRET table entries that are active (in use) at the time of the error. When an application program issues CRETC, ALCS saves information in the CRET table. It uses this information to create the new entry at the correct time. Figure 32 on page 373 shows an example.

```
ACTIVE CRET LIST ITEMS
0007B908  CHN 0007B8E0 00000389 TIM  ACT 00000000 X L M 4  ...
0007B8E0  CHN 0007B840 0000038B TIM  ACT 02024600 C B Q E  ...
0007B840  CHN 00000000 00000E14 TIM  ACT 00000000 X E R B  ...
```

Figure 32. Example of a CRET table dump

Tags identify fields in the CRET table. They relate to labels that the CR0ET macro generates in the CR0ET DSECT. For example, CHN is the tag for the field CR0CHN, and so on.

Macro trace control area

This dump shows the contents of the macro trace control area. This area contains control blocks that the ALCS trace facility uses. The control area consists of:

- Header
- File mode trace entry
- Conversational mode trace entries

Macro trace control areas that are all zeros do not appear in the dump.

Figure 33 on page 374 shows an example.

```
MACRO TRACE CONTROL AREA HEADER
00079700  LKW 00000000 01000000      NBR 00000001 0000000A  ...
00079720 CNV0 00000000 00079D48      028 00000000 0007A038  ...
⋮

MACRO TRACE CONTROL -- FILE MODE TRACE ENTRY
0004E758  LKW 00000000 00000000          00000000 00000000  ...
0004E778  DSP 80880000 00000000      MDE 00000000 00000000  ...
⋮

MACRO TRACE CONTROL -- CONVERSATIONAL MODE TRACE ENTRY
00079A58  LKW 00000001 C0A00000       STK 000784A0 000200BD ...
0006BB88      00000000 00000000           00000000 00000000 ...
********
00079AB8  GRP 00000000 00000000 NBM       00000000 00000000 ...
********
00079AF8      00000000 00000000       NBP 00000000 00000000 ...
********
00079B38      00000000 00000000           00000000 00000000 ...
00079B58  ADP 00000000 00000000 ADD   ADP 00000000 00000000 ...
********
:
```

Figure 33. Example of a macro trace control area dump

Tags identify fields in the resource macro trace control area. They relate to labels that the GT0CA macro generates in the GT0CA DSECT. For example, LKW in the header is the tag for the field GT0LKW, LKW in the file mode trace entry is the tag for the field GT1LKW, and so on.

Program control tables

This dump shows the contents of the tables that ALCS uses to control application programs. For example, ALCS uses these tables to find the storage address of an application program. The tables are:

- Module load table
- Program hash table
- Program control table

Figure 34 on page 374 shows an example.

```
MODULE LOAD TABLE
7F6FD008  000 00000000 00000000      008 00000000 00000000  ...
********
7F6FD048  040 D X C A  A 0 0 0       048 00000000 00000000  ...
7F6FD068  060 00000000 00000000      068 00000000 28007000  ...
⋮

PROGRAM HASH TABLE
7F6F2008  000 P Q M 2  00000000      008 7F657FC8 00000000  ...
********
7F6F2088  080 X M S 2  00000000      088 7F5EB1AC 00000000  ...
⋮

PROGRAM CONTROL TABLE
7F6E5008  000 7F5D95D4 00000000      008 7F6E0008 0000CFF8  ...
********
7F6E5088  080 00000000 00000000      088 00000000 7F6E50B0  ...
⋮
```

Figure 34. Example of a program control tables dump

Pool file control tables

This dump shows the contents of the tables that ALCS uses to control the dispensing and releasing of pool file records. These tables are:

- Pool file control tables
- Pool file directory records

Figure 35 on page 375 shows an example.

```
POOL FILE CONTROL TABLES
7E36D008  LKW 00000000 7F6BFE58 KPA  LMS 00000000 016001C8  ...
7E36D028  KUC 00000000 00000000 SKP SLI2 03600000 0000C080  ...
⋮

POOL FILE DIRECTORY RECORD
7F40D428  BID F P D 00 00000000 PRG      00000000 1B55203F  ...
7F40D448  020 FFFFFF00 00000000      028 0001FFFF FFFFFFFF  ...
⋮
```

Figure 35. Example of a pool file control tables dump

Tags identify fields in the pool file control tables. They relate to labels that the FC0FC macro generates in the FC0FC DSECT. For example, LKW is the tag for the field FC0LKW, and so on.

Tags identify fields in the pool file directory records. They relate to labels that the FP0DR macro generates in the FP0DR DSECT. For example, BID is the tag for the field FP0BID, and so on.

Control and data areas

This dump shows the contents of control/data areas used by various ALCS facilities:

- APPC
- CPU loop TCBs
- Monitor exits
- MQ
- OCTM
- PDU
- SQL
- TCP/IP
- WAS

Figure 36 on page 376 shows an example.

```
APPC CONTROL/DATA AREA (DXCAPPCA)
7F2C3000  TAG A P P C  A R E A        STS 0A400000 0012A000  ...
7F2C3020 SDAT A L C S  M 1 4040      STOK 010000CC 001E0005  ...
7F2C3040  SSN A L C S  M 0 0 0        SUF A L C S  M 2 4040  ...
...

CPU LOOP TCBS CONTROL/DATA AREA (TC0TB)
1080C920 EVCB 800D3C58 00A8CB70 ADDR SAVE 000D3C00 A0000000  ...
1080C940 WAIT 00000000 00000000 BUSY PTBL 00000000           ...
...

MONITOR EXITS CONTROL/DATA AREA (IE0TB)
7F69C000  HDR E X I T  L I S T        CNT 00410000 00000000  ...
7F69C020      U S R S  V C 4040           00000000 00000000  ...
7F69C040  NAM U S R T  I M 1 40       ADR 00000000 00000000  ...
...

MQ CONTROL/DATA AREA (DXCMQIA)
7F2A3000  TAG M Q M 40 A R E A       STAT C0000000 90F8AE2C  ...
7F2A3020  PUT 90F8AEF0 90F8AEF8 PUT1  INQ 90F8AF00 00000000  ...
7F2A3040 MNAM C S Q 1  40404040           40404040 40404040  ...
...

OCTM CONTROL/DATA AREA (DXCTMCT)
7C291000  TAG O C T M  00C72000 SIZE LOK1 00000000 00000000  ...
7C291020  GNL 7CCE5528 7CCE9408 GNT   GN1 7CC91518 7CCB9520  ...
7C291040  CIU 7CCE3527 00001FFF       OIU 7CD007F0 00002000 ...
...

PDU CONTROL/DATA AREA (DXCPDUA)
7F2BC7F4  TAG P D U 40 A R E A        STS 00000000 00000000 ...
7F2BC814      00000000 00000000           00000000 00000000  ...
7F2BC834      00000000 00000000           00000000 00000000  ...
...

SQL CONTROL/DATA AREA (DXCSQLA)
7F333000  TAG S Q L 40 A R E A       STAT 80000000 D B 2 A   ...
7F333020 CABC 00000000 00000000 CECB CSSN D B 2 A  80A84070  ...
7F333040 LIST 7F33302C 7F333030           7F333034 7F333038  ...
...

TCP/IP CONTROL/DATA AREA (DXCSOCKA)
7E081000  TAG T C P A  R E A 40       STS 00800000 7E08E790  ...
7E081020  C08 90FA6780 90FA7E34 IU1   IU2 80147178 00000000  ...
7E081040  IDS 40404040 40404040           I A L A  R E D 5   ...
...
WAS CONTROL/DATA AREA (DXCWASA)
7ED2F000  TAG W A S A  R E A 40      SIZE 00000248 00000000  ...
7ED2F020 ITCB 00000000 7EK 2 48 FTCB INTB 00000000 00000001  ...
7ED2F040 TOD1 00000000 00000000      1REG 995E4B58 995E4AE8  ...
...
```

Figure 36. Example of a dump of control/data areas for APPC, CPU loop TCBs, Monitor exits, MQ, OCTM, PDU, SQL, TCP/IP, WAS

ALCS entry dispatcher work lists

This dump shows the contents of the ALCS entry dispatcher work list headers. Figure 37 on page 377 shows an example.

```
ENTRY DISPATCHER WORK LIST -- IOCB LIST
00076008  TOP 00000000 00000000 CDS   BOT 00076008 00000000

ENTRY DISPATCHER WORK LIST -- READY LIST
00076038  TOP 00000000 00000000 CDS   BOT 00076038 00000000

ENTRY DISPATCHER WORK LIST -- DELAY LIST
00076050  TOP 00000000 00000000 CDS   BOT 00076050 00000000
⋮
```

Figure 37. Example of an ALCS entry dispatcher work lists dump

Tags identify fields in the ALCS entry dispatcher work list headers as follows:

TOP
Address of first item on work list

CDS
Not used

BOT
Address of last item on work list

Block list descriptors

This dump shows the contents of the block list descriptors. ALCS uses the block list descriptors to control the dispensing and release of I/O control blocks (IOCBs) and storage units.

Figure 38 on page 377 shows an example.

```
BLOCK LIST DESCRIPTOR -- IOCBS
00076B00  ADR 7F714000 00000079 NBR  FLG 00000000 00077B10  ...
00076B20  PBS 00000200 00000080 TOT  MSK FFFFFFE0 00000004  ...
00076B30  030 00077B10 7F714000      038 00000000 80000000  ...
⋮

BLOCK LIST DESCRIPTOR -- STORAGE UNITS
00077B30  ADR 02E01000 0000002D NBR  FLG 00000000 000787A0  ...
00077B50  PBS 00007000 00000032 TOT  MSK FFFFFFC0 00000009  ...
00077B60  030 000785E0 02E01000      038 0003DCC2 40000000  ...
⋮
```

Figure 38. Example of a block list descriptors dump

Tags identify fields in the block list descriptors. They relate to labels that the LS0LS macro generates in the LS0LS DSECT. For example, ADR is the tag for the field LS0ADR, and so on.

I/O control blocks

This dump shows the contents of IOCBs. Figure 39 on page 377 shows an example.

```
DASD IOCB -- IN USE
7F721E00  CHN 00000000 00059D5A EXT  EVC 40000000 01080000  ...
7F721E20  ADR 7F567000 00310800 BTY  SAV 00000000 00000000  ...
********
⋮
```

Figure 39. Example of an I/O control block dump

The heading shows the IOCB type, and whether the IOCB is in use or available.

Tags identify fields in the IOCB. They relate to labels that the IO0CB macro generates in the IO0CB DSECT. For example, CHN is the tag for the field IO0CHN, and so on.

Monitor interface area

This dump shows the contents of the online monitor interface area, with tags. Note especially that the monitor CSECT base addresses are tagged. This is the first part of the ALCS online monitor nucleus CSECT. Figure 40 on page 378 shows an example.

```
MONITOR INTERFACE AREA
0005D2C0      D X C N  U C               A Q 2 9  4 7 8    ...
0005D2E0 ASTI 0005D300 0005D304 ACLS ANEB 0005D308 00000000 ...
0005D300  STI 09D01260 09D00634 CLS   NEB 09D001E8 09D001F8 ...
0005D320 SAVS 0009D000 000C2800 SAVE SRBS 000C3000 09D01208 ...
0005D340      00000000 00000000       CDS 7F6BF000 7F521200 ...
0005D360 CDS2 00000000 00000000           00000000 00000000 ...
0005D380 INTC 0004FFB8 00051F60 INTD INTS 00054820 00053A70 ...
0005D3A0 INTE 00052F38 00000000           00000000 00093688 ...
0005D3C0      00000000 00000000           00000000 00000000 ...
0005D3E0  DMP 00037DC8 00039CD0 EMSG DMPC 00038DB8 00038FA0 ...
0005D400      00000000 000353E8 DCR   GTF 0004A6D0 00049370 ...
0005D420      00000000 00000000           00000000 0006B590 ...
0005D440 SEQP 0006AFC8 00000000           00000000 00096EA8 ...
0005D460  VFC 00095450 00097890 VFP   VFH 00095CB0 00095FB0 ...
0005D480  DAV 00034648 000338B8 DAT   DAS 00032B20 0003F940 ...
0005D4A0      00000000 00047838 GFS   GFC 000428C0 00043DA0 ...
0005D4C0  PDU 000618C8 00000000           00000000 00057BB0 ...
0005D4E0      00000000 00078620 SND   OPZ 0005F3A0 0001FCE8 ...
0005D500 COMP 00020BC0 0002F2B0 COM3 COLF 00014598 00097F00 ...
⋮
```

Figure 40. Example of a monitor interface area dump

Monitor work areas

This dump shows the contents of the online monitor work areas. Figure 41 on page 378 shows an example.

```
MONITOR WORK AREA   1 -- NUC
00068800  PLI 00068800 00000000 HSA  LSA 00072000 8003CBB2  ...
00068820  RGB 00076068 00000000 RGC  RGD 0003E0E8 000128F8  ...
⋮

MONITOR WORK AREA   1 -- TIM
00068C00  PLI 00000000 00068800 HSA  LSA 00069800 02EF94A0  ...
00068C20  RGB 00076488 00000000 RGC  RGD 8FE51E01 000787E0  ...
⋮
```

Figure 41. Example of a monitor work areas dump

The heading shows the task that owns the work area and the function that uses the section of the work area. Monitor work area 0 belongs to the main ALCS task, monitor work area 1 belongs to the first CPU loop task, monitor work area 2 to the second CPU loop task, and so on. NUC indicates that the nucleus CSECT (DXCNUC) uses the section of the work area, TIM that the timer routines (DXCTIM) use the section, and so on.

Tags identify fields in the monitor work area. They relate to labels that the MN0SV macro generates in the MN0SV DSECT. For example, PLI is the tag for the field MN0PLI, and so on.

Application global area

This dump shows the contents of the application global area. The dump lists the various parts of the application global area in the following order:

- Directory 0, area 1
- Directory 0, area 2
- Directory 0, area 3
- Directory 1, area 1
- Directory 1, area 2

and so on. Figure 42 on page 379 shows an example.

```
APPLICATION GLOBAL AREA -- DIRECTORY 0 -- AREA 1
7f70D000      00000000 00000000          00000000 000000000  ...
********
7F70D040 GBLB 7F70D1C0 00041C40     GBLC 7F70D5E0 00051C40   ...
⋮

APPLICATION GLOBAL AREA -- DIRECTORY 0 -- AREA 2
7F705000  000 A H 5020 C B X N       008 00000000 000000000  ...
⋮
```

Figure 42. Example of an application global area dump

Tags identify fields in the application global area directories. The GL*n*BA macros generate these tags. GL0BA generates the tags for directory 0, and so on.

Entry storage

This dump shows the contents of all the ALCS storage units. If a storage unit contains an ECB, a formatted ECB dump follows the storage unit dump. Figure 43 on page 379 shows an example.

```
******** ENTRY STORAGE FOLLOWS

BLOCK LIST DESCRIPTOR -- PRIME SU -- TIME 12.29.57.0
00077B60  PCH 000785E0 02E01000 PBA  PPI 0003DCC2 40000000  ...
⋮

ENTRY CONTROL BLOCK DESCRIPTOR
02E00000  BDA 00077B60 00000000          00000000 00000000  ...
⋮

ENTRY CONTROL BLOCK PREFIX
02E010A0  PFX 00077B60 00000000          00000000 00000000  ...
⋮

ENTRY MACRO TRACE DATA
PROG  MACRO  ADDR  REPEAT   PROG  MACRO  ADDR  REPEAT  ...
TIM   CXFRC  0DC8           NUC   ENTDC  1DF6          ...
⋮

ENTRY CONTROL BLOCK -- CREATED BY ALCS MONITOR -- LAST  ...
02E014A0  CHW 00000000 0003DCC2 BAD
⋮
```

Figure 43. Example of an entry storage dump

Entry summary

A summary of all the ALCS storage units that contain ECBs. Each line of the summary gives information about one entry. The information is in columns, and a heading line identifies the columns as follows:

DESC
Address of the ECB descriptor.

CHAIN
ECB descriptor chain. That is, the address of the next ECB descriptor in the chain. For example, if an ECB is on the ready list, this chainword contains the address of the descriptor of the next ECB on the ready list.

DEADLOCK
ECB deadlock-detector chain. If the ECB is waiting for record hold, resource hold, or sequential file assign, this chainword contains the address of the descriptor of the ECB that this ECB is waiting for.

ECB
ECB address.

CRI
Communication resource identifier (CRI) of the resource that originated the transaction that this entry is processing.

REC HOLD
Count of record addresses that this entry is holding.

RES HOLD
Count of resources that this entry is holding.

SEQ FILE
Count of sequential files assigned to this entry.

LAST MACRO
Information about the last monitor-request macro that this entry executed. That is, the program name, macro name, and listing address of the macro within the program.

STATUS
Status of the entry. IN USE means that the storage unit is not available for use by other entries. MSG WAIT means that ALCS is waiting for a VTAM RECEIVE to complete. When the RECEIVE completes, the entry processes the message that arrives.

VFA control areas

This dump shows the contents of the tables that ALCS uses to control VFA buffers. These tables are:

- Control area, common section
- Control area, size-dependent data
- Prime record locator table
- Overflow record locator table

Figure 44 on page 381 shows an example.

```
VFA CONTROL AREA -- COMMON SECTION
7F731618 RLPA 7F5CD000 000001CD RLPC RLOF 7F5CED30 FFFFFDCE  ...
7F731638   L1 7F7316E0 7F731760 L2     L3 7F7317E0 7F731860  ...
********

VFA CONTROL AREA -- SIZE DEPENDENT DATA
7F731660 LOCK 000000FF 00000000 NBRB BLEN 00000000 00110000  ...
7F731680  PRF 00000000 00000000 PRL    S1 00000000 00000000  ...
7F7316A0  HSN 00000000 00000000       HST 00000000 00000000  ...
7F7316C0 HSNR 00000000 00000000 HSRP HSLG 00000000 00000000  ...
********

VFA FILE ADDRESS LOOKUP TABLE
7F5139A8      7F50BC28 7F160520           02240140 00020001  ...
7F5139C8      7F50B9B8 7F160520           00480140 00020001  ...
⋮

VFA RECORD LOCATOR TABLE - PRIME
7F7CD000      00000000 00000000           00000000 000000FF  ...
7F7CD020      00762D40 7F5C6640           00000000 000000FF  ...
⋮

VFA RECORD LOCATOR TABLE -- OVERFLOW
7F5CECD0      00073040 7F5C75C0           00000000 7F5CE910  ...
7F5CECF0      00273040 7F5C7900           00000000 7F5CD580  ...
⋮
```

Figure 44. Example of a VFA control areas dump

Tags identify fields in the VFA control area common section and size-dependent data. They relate to labels that the VC0CA macro generates in the VC0CA DSECT. For example, RLPA is the tag for the field VC0RLPA, LOCK is the tag for the field VC0LOCK, and so on.

The VL0RL macro generates the VL0RL DSECT. It describes the record locator table format.

VFA buffer headers

This dump shows the contents of the VFA buffer headers. Figure 45 on page 381 shows an example.

```
VFA BUFFER HEADERS
7F5C5000      00000000 00000000          00000004 00000000  ...
7F5C5020      00000000 7F5B2000          00000000 00000000  ...
7F5C5040      7F5CE560 7F5CE560          02020004 09200080  ...
⋮
```

Figure 45. Example of a VFA buffer headers dump

The VH0BH macro generates the VH0BH DSECT. It describes the buffer header format.

VFA buffers

This dump shows the contents of the VFA buffers.

SLC link and channel keypoints

This dump shows the contents of the SLC link and channel keypoints. Figure 46 on page 382 shows an example.

```
SLC LINK KEYPOINT
7F45C000  HDR 00000000 00000000          00000000 0000F400  ...
7F45C020  SOF 03020020 00FF00F0 XMT  KCH 10802000 0003001F  ...
⋮

SLC CHANNEL KEYPOINT
7F45CBD8  HDR 00000000 00000000          00000000 0000F500  ...
7F45CBF8  NSR 00000000 00000000 RRT  EXC 00000000 00000000  ...
⋮
```

Figure 46. Example of an SLC link and channel keypoint dump

Tags identify fields in the link and channel keypoints. They relate to labels that the LK4KC and LK5KC macros generate in the LK4KC and LK5KC DSECTs. For example, HDR in the link keypoint is the tag for the field LK4HDR, and so on.

System error dump areas

This dump shows information specific to the system error code.

- System area 1 contains a VFA buffer header.
- System area 2 contains a block descriptor, ECB descriptor, ECB prefix, and ECB (without any attached or detached blocks).

See the description of the system error code for more details about the content of system error dump areas.

User defined areas

This dump shows the contents of storage areas that the user-written dump exit routine requests. See *ALCS Installation and Customization* for a description of this installation-wide exit routine.

Program driver

About this task

It is sometimes necessary to write an ALCS application program to perform a one-off function, for example:

- To test a user-written macro service routine
- To rebuild a chain of pool file records that is damaged

To simplify writing these one-off programs, ALCS provides a program driver facility, the ZDRIV command, which activates any specified program.

To use the program driver facility:

Procedure

1. Assign a unique name to the application program.
2. Assemble the program and link-edit it into a load module.
3. Use the ZPCTL command to load it into the system.
4. Use the ZDRIV command to activate the program. Optionally enter a character string to pass to the program. This character string appears to the called program as an input message. That is, on entry to the program, the character string is in a storage block attached on level 0, and in the IMSG input message format described in *ALCS Application Programming Reference - Assembler*.

   If no character string is given, an empty IMSG is attached on level 0.

Results

See “ZDRIV -- Activate application program” on page 193.

ZDRIV can also activate any existing application program, for example:

- To activate a specially written staging program that sets up entry conditions and enters an existing application program.
- To activate an existing application program directly if no special entry conditions are required.

SLC link trace facility

The SLC link trace facility monitors activity on the SLC network. Use it during testing or production operation. The SLC link trace facility records data each time a block is sent or received on an SLC channel. Use the ZLKTR command to activate and deactivate the link trace facility, and to specify the scope of the trace details). For example:

- Trace selected SLC links; that is, all the channels on the links.
- Trace all types of blocks, or one or more selected types of blocks only. For example:
  - Type B message blocks
  - Type A message blocks
  - Network control blocks
  - Link control blocks

SLC link trace can write the trace information to any or all of the following:

A printer terminal
The terminal must be either a CRAS printer or a printer associated with a terminal.

Note: Trace to a terminal imposes a higher load on ALCS than either trace to SLC trace block or trace to the ALCS diagnostic file, so use it with caution.

SLC link trace block
Use the ZLKTR command to view the current contents of the SLC link trace block to a display terminal. See “ZLKTR -- Control SLC link trace” on page 208 for more details of this command.

The ALCS diagnostic file
Run the ALCS diagnostic file processor to print the output. Use ALCS diagnostic file processor options to select how to print SLC link trace information for example:

- Do not print SLC link trace information.
- Print all SLC link trace information.
- Print SLC link trace information from selected time intervals.

For each LCB and message block the ALCS diagnostic file processor prints:

```
hh.mm.ss.t SLC type crn KCNn information
```

The ALCS diagnostic file processor can also print the following status messages for the SLC link trace facility:

```
hh.mm.ss.t SLC STARTING
hh.mm.ss.t SLC ENDING
hh.mm.ss.t SLC INVALID OP CODE - lcb_data
hh.mm.ss.t SLC RESUMING AFTER SHUTDOWN - nnnn LOST ITEMS
```

Where:

hh.mm.ss.t
Time-of-day (TOD) clock time.

type
One of:

R
Block received on SLC channel.

W
Block sent on SLC channel.

crn
CRN of the link.

KCN*n*
Where *n* is the number of the channel within the link (from 1 to 7).

information
Information about the block. This information includes:

LCB=*lcb_type*
LCB type.

ATSN=*atsn*
The acknowledge transmission sequence number (ATSN).

TSN=*tsn*
The transmission sequence number (TSN).

LEN=*size*
Block size (number of bytes).

EIB=*error_byte*
Error index byte. For an explanation of the EIB, refer to *IBM 3705 Emulation Program Generation and Logic Manual for LICRA Line Control*

DATA=*data*
First 4 to 63 bytes of the block in hexadecimal. Use the ZLKTR command to select the number of bytes to be traced.

MBI=*type_label - block_number*
The message type (Type A or Type B) followed by the message label and the message block number.

AMBI=*type_label*
The message type (Type A or Type B) followed by the acknowledge message label.

LCI=*message_details*
Four fields, as follows:

PD or ND
Message is (PD) or is not (ND) a possible duplicate message or possible duplicate block.

HL or NH
Message is (HL) or is not (NH) for a high-level network.

BP or NP
Message is (BP) or is not (NP) protected.

LB or NL
Message is (LB) or is not (NL) the last block of the message.

HLN=*hen/hex*
Entry (*hen*) and exit (*hex*) addresses of the high-level network.

ACI=*code*
Message translate code.

lcb_data
First 12 bytes of the block, in hexadecimal.

nnnn
Number of trace entries lost because of link trace shutdown.

TCP/IP trace facility

The TCP/IP trace facility monitors activity on the TCP/IP network. Use it during testing or production operation. The TCP/IP trace facility records data each time a data block is sent or received on a TCP/IP communication connection. Use the ZACOM command to activate and deactivate the TCP/IP trace facility for each TCP/IP communication resource. ALCS writes TCP/IP trace information to the following:

- System TCP/IP trace block
  ALCS always writes TCP/IP trace information to the system TCP/IP trace block. Use the ZCTCP command to view the current contents of the system TCP/IP trace block on a display terminal.
- Diagnostic file
  Use the ZCTCP command to start and stop writing TCP/IP trace information to the ALCS diagnostic file. Run the ALCS diagnostic file processor to print the output.
- Web browser
  Use the ALCS Web server to display TCP/IP trace information at a remote Web browser. To do this, you must create a directory entry in the ALCS hierarchical file system (HFS) for ECB-controlled program CTCW, then provide a corresponding uniform resource locator (URL) to the Web browser.

  For example, create the following HFS directory entry:

  ```
  mkprg tcpiptrace.cgi ctcw
  ```

  and provide the following URL to your Web browser:

  ```
  http://alcswebserver.com/tcpiptrace.cgi
  ```

  See "ALCS World Wide Web Server User Guide" for more information about the ALCS Web server and the ALCS HFS.

For each data block, ALCS shows:

```
TCP/IP type hh.mm.ss.iii CRN-crn LEN-size
data
⋮
```

Where:

type
One of:

IN
Data block received on TCP/IP connection.

OUT
Data block sent on TCP/IP connection.

hh.mm.ss.iii
Time-of-day (TOD) clock time.

crn
CRN of the TCP/IP communication resource.

size
Data block size (number of bytes).

data
First 372 bytes of the data block in hexadecimal. Each output line shows up to 36 bytes of data.

ALCS test facilities

ALCS provides:

- The dynamic program load facility for testing individual programs or groups of programs.
- The system test vehicle (STV) for testing the system; that is, testing all the applications together, using a simulated network.

These are described in the following sections.

Dynamic program load facility

The ALCS dynamic program load facility is a tool for testing modified or new application programs from a terminal, without affecting the rest of the system. ALCS program management can load single or multiple copies of a program as a test program. Program management manages the association of test programs with test terminals. ALCS communication support sets a flag in the communication tables to indicate that a terminal is in test mode. When an entry is from a terminal in test mode, the communication routine flags the ECB to indicate that it is a test entry. ALCS program management selects test copies of programs for these entries.

Terminal in test mode

Use the ZPCTL command to load a test module for a terminal. A test module belongs to a particular terminal (either the terminal specified on the ZPCTL command or, if no terminal is specified, the ZPCTL input terminal). See “ZPCTL -- Load/unload programs and installation-wide monitor exits” on page 244 for details.

Test entries

ALCS treats all input from terminals in test mode as test input. ALCS communication support flags the ECB to indicate that it is a test entry. Program management resolves program calls for test entries by selecting test copies of programs where applicable.

System test vehicle

The ALCS system test vehicle (STV) is a tool for testing ALCS configurations and application programs before starting production use. Use it also to test the system after changing the configuration or application programs.

STV simulates a communication network that can consist of ALC, IBM 3270, and WTTY terminals.

STV reads input messages from a test unit data set. STV enters these messages into ALCS as if they come from a real terminal. STV allows simulated input only from terminals that are specified as test terminals in the ALCS generation.

STV intercepts data that is sent to a test terminal, and writes it to the ALCS diagnostic file. Use the ALCS diagnostic file processor to process and format STV data.

Preparing for system test

To obtain the maximum benefit from system test using STV, prepare for the test as follows:

1. Prepare the basic test plan.

   Decide what major areas of the system to test.

   Note:

   STV does not provide an appropriate environment for testing ALCS commands. If it is necessary to test these functions, prepare a separate plan to test them "hands on".
2. Design the test network configuration.

Decide what terminals are needed to enter test messages. Note that to test the system's capability to handle many input messages simultaneously, it is necessary to include several of each type of terminal.

3. Generate the ALCS test system, including the test network.
4. Initialize the database as required for testing purposes.
5. Specify the test input messages together with their expected responses.

   Where possible, avoid test input messages that depend on the initial contents of the database.
6. Use the ALCS system test compiler (STC) to prepare test unit data sets.

   When preparing the input messages and expected responses, group the messages on separate test unit data sets in such a way that, if there are errors, the correction can be tested by processing only one data set rather than restarting from the first data set. See *ALCS Installation and Customization* for a full explanation of STC.

   Note: STC may display error responses from the PL/I high level language environment. See "Bibliography" on page 446 for a list of associated documentation.

Running STV

Use the ZTEST command from a ***Prime CRAS authorized*** terminal to start and control STV. See "ZTEST -- Control system test vehicle" on page 303 for more information about this command.

STV output listing

STV intercepts output messages to test terminals. It writes them, together with copies of the input messages from the test unit data set, to the ALCS diagnostic file. Run the ALCS diagnostic file processor to format and print this data.

The output listing can include:

- Start and end of test indicators
- Input and output messages

Start and end of test indicators: The following messages indicate the state of STV.

```
TEST STARTED
TEST PAUSED
TEST RESTARTED
TEST CANCELLED
TEST STOPPED
```

In each case, the system test compiler RUNID control statement is also printed. See *ALCS Installation and Customization* for a full explanation of the system test compiler RUNID control statement.

Input and output messages:

Messages are printed as they appear at a terminal. Vertical bar (|) characters indicate the display or paper margins as follows:

Display or printer
Right margin position

WTTY display
70

ALC
64

IBM 3270 display
80

IBM 3270 printer
No right margin

Where applicable, an underscore (_) character indicates the cursor position.

To the left of the first line of each message there is a heading that indicates:

- Input message or output message.
- WTTY or non-WTTY terminal.
- Message sequence number. (STV maintains separate sequences of numbers for input and output messages, and for WTTY and non-WTTY messages.)
- Terminal CRI.

For non-WTTY messages, the end-of-message character prints as:

#EOM
End-of-message complete (X'4E')

#EOI
End-of-message incomplete (X'6D')

#EOU
End-of-message unsolicited (X'5F')

#EOP
End-of-message push button (X'7E')

#ETX
End-of-text (X'03')

STV simulates hardware answerbacks (positive acknowledgments) from printer terminals.

Multi-block messages on display terminals are printed with the start of text in each block starting at the beginning of a line. With real terminals this does not occur because there is no automatic new line at the end of a block. Some printer terminals perform a new line at the end of each block, so the ALCS diagnostic file processor correctly simulates this type of terminal.

Leading and trailing letter-shift (#LSC) characters on WTTY messages are indicated as follows:

```
*** nn LEADING LSCS
*** nn TRAILING LSCS
```

If the first character of a WTTY message line is a case shift character, the ALCS diagnostic file processor does not print it. Instead, it prints the tag "LSC" or "FSC", as appropriate, in the left-hand margin.

Sometimes the ALCS diagnostic file processor cannot print messages as they would appear at the terminal. This is usually because of an error in the message. In such cases, ALCS diagnostic file processor prints a hexadecimal dump of the message block, with the following heading:

```
UNABLE TO FORMAT MESSAGE - reason
```

where *reason* can be:

UNKNOWN LINE TYPE
The ALCS diagnostic file processor cannot determine the type of terminal.

LINE(S) TOO LONG
One or more lines of the message exceed 79 characters (non-WTTY message) or 69 characters (WTTY message).

INVALID END OF MESSAGE
An ALC message does not end with a valid end-of-message character (#EOM, #EOI, #EOU, #EOP, or #ETX).

UNPRINTABLE CHARACTER(S)
There are one or more unprintable characters in the message text.

TOO MANY LSCS
There are more than 99 leading or trailing letter-shift characters on a WTTY message. This is probably, but not necessarily, an error.

LF MISSING AFTER CR
A carriage return character (#CAR) is not immediately followed by a line feed character (#LFEED) in a WTTY message. Note that the sequence #LFEED #CAR is not correct.

LSC/FSC NOT AFTER LF
A case shift character does not immediately follow a line feed character in a WTTY message. Although this is not an error, the ALCS diagnostic file processor cannot indicate the position of a case shift character unless it is the first character of a line. The message is printed in hexadecimal.

If STV cannot process an input message from a test unit data set, the message is printed in hexadecimal, followed by:

```
STV UNABLE TO PROCESS MESSAGE - reason
```

where *reason* can be:

INVALID CRI
This is issued for one of the following reasons:

- The CRI does not exist.
- The resource is a type that STV does not support.
- The resource cannot logically input a message to ALCS (for example, a simplex out WTTY line).

RESOURCE IS NOT AN STV RESOURCE
The originating terminal is not a test terminal. See *ALCS Installation and Customization* for details of how to define test terminals.

OUTPUT MESSAGE LOST - INSUFFICIENT STORAGE
STV cannot obtain storage to record the output message.

Information and error messages

ALCS writes some information and error messages to the ALCS diagnostic file. The ALCS diagnostic file processor prints these messages as follows:

hh.mm.ss event_description

Where:

hh.mm.ss
Time-of-day (TOD) clock time of the event.

event_description
Type of event, described in the following section.

TCP/IP messages

CRN-*crn* TCPIP MAXIMUM CONCURRENT SERVER CONNECTIONS
The maximum number of concurrent connections has been reached for the TCPIP communication resource with CRN *crn*.

CRN-crn TCPIP CONNECTION CLOSED BY INSTALLATION-WIDE EXIT
The installation-wide monitor exit USRTCP2 closed the connection for the TCPIP communication resource with CRN *crn*.

Performance monitoring and control

Performance monitoring and control for real-time systems is an essential part of installation management.

Three functions in ALCS: data collection, the statistical reports generator (SRG), and the performance monitor, are designed to assist programming and non-programming staff in such tasks as:

- Capacity planning

- Verifying response time criteria
- Identifying the cause of performance problems
- Monitoring the use and impact of a new application

MVS and other subsystem facilities

In addition to data collection and SRG, and the performance monitor, you also have all the MVS and other subsystem facilities for monitoring and controlling the performance of your system. These are fully described in the appropriate MVS and subsystem manuals.

Use MVS Resource Measurement Facility (RMF) in conjunction with SRG. RMF reports on the overall system performance. SRG reports on what is happening in the ALCS region.

Alternatively, you can use the IBM program Service Level Reporter (SLR) to analyze data collection records on the ALCS diagnostic file, and produce reports tailored to your requirements. You can find the format of the records in *ALCS Application Programming Reference - Assembler*.

Data collection

Data collection is an online ALCS function that can optionally collect the following classes of performance data:

- DASD I/O operations
- VFA
- ALCS waits
- Input and output messages
- SLC communication
- X.25 communication
- Application programs
- Entries

ALCS writes the data collection output to the ALCS data collection file or, if not defined in the sequential file generation, to the ALCS diagnostic file.

Use the ZDCLR command (see "ZDCLR -- Control data collection" on page 134) to start and stop data collection and to select the classes of performance data to collect. Alternatively, use the DCLOPTS parameter of the SCTGEN generation macro to start data collection automatically as part of ALCS startup (see the *ALCS Installation and Customization* for details.).

Data collection entries data

The minimum data that you can collect (without stopping data collection completely) is the entries data. For each entry, ALCS (and optionally, the application) records data in a storage area associated with the ECB. For example, the storage area contains counters for DASD I/O operations; ALCS maintains this count. When each entry exits, data collection writes this information to the sequential file.

There are two types of entry data collection data, as follows:

System data

During the life of an entry, the ALCS monitor accumulates statistics in the entry system data collection area.

Application data

During the life of an entry, application programs can store and update information in the entry application data collection area. This is done using the DCLAC macro. This is described in *ALCS Application Programming Reference - Assembler*.

Application programs can only use the entry application data collection area if you have defined it in the system configuration table. See *ALCS Installation and Customization* for information on how to code the DCLECBU parameter on the SCTGEN macro.

Data collection performance overhead

Data collection increases the processing load on the system, particularly when collecting all classes of performance data. If the processing load reduces the available storage units to below a preset level, ALCS delays processing new messages; this invalidates the data collected. To check that this does not happen, monitor the available storage units against the preset level (activity control variable). See "ZAACV -- Alter activity control variables" on page 67 for a description of activity control variables (the ACV parameters). Use the ZSTAT command to display available storage units and the ZDACV command to display the activity control variables.

To produce reliable statistics, run data collection regularly at peak times, and for long enough to include a representative message mix (say 30 minutes). Regular data collection over short periods is preferable to occasional longer periods. The amount of data output to the ALCS data collection file can be a limiting factor. The program collector produces the greatest output, and you may wish to consider running this collector less frequently.

Use the ZDCLR command (described in "ZDCLR -- Control data collection" on page 134) to start and stop data collection and to select the classes of performance data to collect.

When data collection ends, ALCS continues to write other diagnostic data to the ALCS data collection file data set, which will be closed automatically when it is full. If you want to process the ALCS data collection file at once, use the ZSSEQ command to close it. Whether you close the data set or wait for it to close automatically, SRG will extract the data collection material from any the other data that is on the file.

Performance monitor

The ALCS performance monitor is an online tool written to overcome some of the drawbacks of using data collection. The performance monitor collects statistics in memory and saves summarized information to ALCS pool records.

The performance monitor has low overhead and collects system data as well as data by application. The data is collected continuously and summarized in memory. Historical data is kept on file.

The performance monitor accumulates data in **application categories**; you can choose up to 152 different categories to suit your installation. An application category can represent an ALCS application (such as reservations) or an action code, terminal type, origin city, or anything that can be used to identify an entry. In this way, you can measure consumption data according to various different criteria.

You can add and delete application categories using the ZPERF ADD and ZPERF DELETE commands. Entries identify themselves to the performance monitor using the APIDC macro which is described in *ALCS Application Programming Reference - Assembler*. APIDC can specify up to three different application categories to be used for collecting performance data for a particular entry.

For entries that do not identify themselves using APIDC, the performance monitor collects performance data using a default application category.

Additionally, all entries - whether identified to the performance monitor or not - are collected within the **system total** category.

Identifying entries to be collected in each performance monitor application category is based on the user's logic. This identification may be based on action code, terminal id, or any other entry parameters. You may find it convenient to use a service program to set the appropriate performance monitor application categories as soon as an entry is initialized. The service program would use rules designed by the user.

Scope and limitations

ALCS runs in a single MVS address space. Within this address space, work is executed under control of several MVS tasks. The most prominent of these tasks are the CPU loop task(s) under which all applications, ECB macro services, IOCB routines, and several system processes execute.

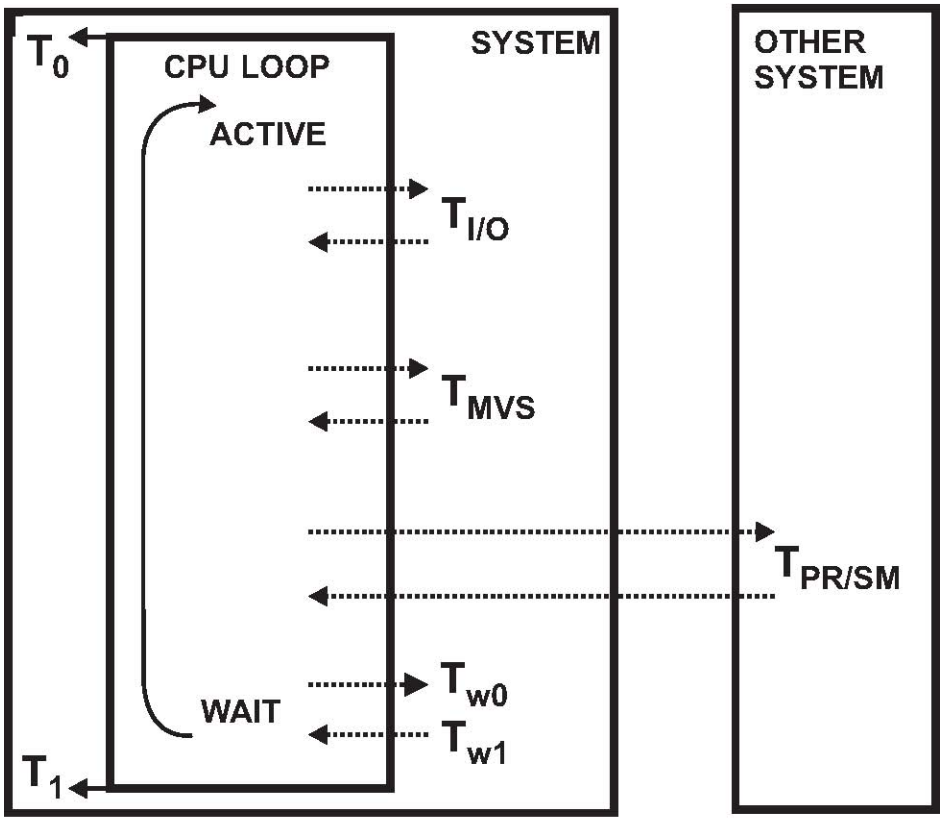
It is the CPU loop task(s) that the ALCS performance monitor measures in some detail. The performance monitor also accrues CPU usage information for the other MVS tasks used by ALCS, which can be displayed with the ZPERF TASK command.

The ALCS performance monitor uses exponential smoothing to obtain the total values shown in the performance monitor displays. A change in system activity will result in these totals not adding up to 100%.

In summary, the figures displayed by the ALCS performance monitor are indicative; they may not be precise. However, the data provided is a valuable resource in monitoring the system.

Historical data collected by the ALCS performance monitor is averaged for a monthly period. Adding an application to the performance monitor during a month will result in no average being available for the month in which the application is added. For this reason it is best to add applications at the end of a month.

CPU Collection



Cycle Time = $T_1 - T_0$

Cycle Time = Σ(Active Time) + Σ(Wait Time)

Wait Time = $\Sigma(T_{w1} - T_{w0})$

Process Time = Σ(Cycle Time) - Σ(Wait Time)

Interruptions = $\Sigma T_{I/O} + \Sigma T_{MVS} + \Sigma T_{PR/SM}$

CPU Time = Active Time - Interruptions

How CPU time is measured

Within the MVS unit-of-work control blocks, the TCB and SRB, is a CPUTIME field. This field is updated by the MVS TIMEUSED macro and gives a precise figure for the CPU used by the unit of work when the field is updated. Time for which the task is not using the processor due to interruptions for PR/SM or I/O or other system interrupts is not included.

The ALCS performance monitor uses the TIMEUSED macro to record the CPU used within in each performance monitor cycle. The ALCS performance monitor apportions this CPU time used to individual entries based on the TOD values in the entries. The figures derived will give a good indication of CPU usage but not a precise value.

CPU usage

CPU time is derived from data gathered using the MVS TIMEUSED macro. This differs from the **cycle time** in that the MVS TIMEUSED macro takes into account time for which the address space did not have control. Control can be lost without ALCS being aware of it by interruptions.

$$\text{Cycle Time} = \text{TOD Value } T_1 - \text{TOD Value } T_0$$

$$\text{CPU Time} = \text{MVS TIMEUSED } T_1 - \text{MVS TIMEUSED } T_0$$

$$\%\ \text{CPU} = \frac{\text{CPU TIME}}{\text{CYCLE TIME}} \times \frac{\text{Number of TCBs}}{\text{Number of Processors}}$$

ECB life time

ECB life time is calculated by subtracting the time-of-day (TOD) clock at ECB creation from the TOD clock at ECB end.

$$\text{ECB Life Time} = \text{TOD}_{\text{ECB END}} - \text{TOD}_{\text{ECB CREATION}}$$

ECB life time can be split into three components: application time, system time, and wait time.

Application time is the time spent within the ECB-controlled application programs. System time is time spent by ALCS on behalf of the application to process macro service calls and other asynchronous activities. Wait time is the time when the ECB is not running; the ECB may be waiting for an external event such as I/O to complete, or it may be ready to execute and waiting on a queue.

ECB application time

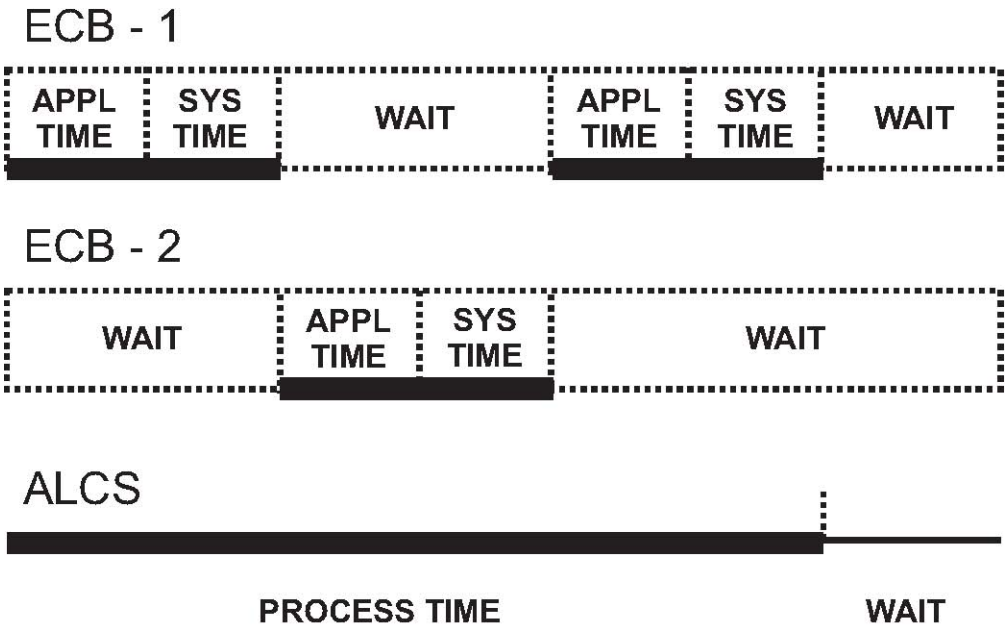
The application time is measured by recording the time-of-day (TOD) clock before returning to the application (GENEXT) and also when the application issues a macro service call (GENENT).

$$\text{ECB Application Time} = \Sigma(\text{TOD}_{\text{GENENT}} - \text{TOD}_{\text{GENEXT}})$$

ECB system time unit

The ALCS performance monitor does not measure the time spent executing macro calls directly. Many macros lose control at various points in their processing. Rather than intercept all these points to deduce the system time and wait time, the simplification is made that all macros require the same processing.

The processing time used by all entries is derived. The processing time within a cycle is the cycle time minus the wait time. The processing time is made up application and system time. System time for a cycle is the processing time minus the sum of the application time for the cycle.



$$\text{Process Time} = \Sigma(\text{Cycle Time}) - \Sigma(\text{Wait Time})$$

$$\text{Process Time} = \Sigma(\text{Application Time}) + \Sigma(\text{System Time})$$

$$\text{System Time Unit} = \frac{\Sigma(\text{System Time})}{\text{number authorized macros}}$$

Many macros in ALCS result in no loss of control. Typically, these macros are ones that do not need to gain authorization. Authorization is the ALCS process, conditionally required at GENENT, to update the program key mask so that the system can update tables key storage. The ALCS performance monitor only counts authorized macros. Unauthorized macros tend to be of very short path length and do not lose control; the ALCS performance monitor is not aware of them and treats them as part of the application program.

Collection Cycle

Each CPU loop task has a performance monitor table associated with it. Periodically, data collected in this table is collated into a system-wide performance monitor table. This period is called the **collection cycle**.

The ALCS performance monitor cycle is 1.6 seconds.

Every 50 milliseconds, ALCS sequentially flags one of the CPU loop tasks to call the performance monitor summarization routine. This allows for up to 32 CPU loop tasks to be processed in the collection cycle. And spreading the collection process calls for each task over the cycle minimizes the chance that serialization of the process is required.

Once a cycle is complete, the CPU time and cycle time are measured. The amount of variance of the cycle time above 1.6 seconds is an indicator of the amount of external interference.

Collection Cycle Smoothing

The summarization process takes data from the task performance monitor table and adds it to the system table. This is done by taking 15 times the value in the system table, adding the data from the task table, dividing the sum by 16 and saving the result in the system table.

The ZPERF command displays the values stored in the system table.

Once the data from the task table has been summarized to the system table, the data in the task table is cleared.

Application Identification

Performance data is recorded for all entries in the system. By default, all entry data is recorded in a single category.

Data can be recorded into specific categories. These categories are called **applications**[5] in the performance monitor.

An entry is identified as part of a particular performance monitor application using a field in the ECB descriptor called the **application identifier**. This field contains an index into the performance monitor tables, where consumption data is gathered and summarized.

The application identifier is placed in the ECB descriptor by the application program, using the APIDC monitor-request macro. Created entries get the application identifier copied from their creating ECB. APIDC may be used at any time not only to initialize the application identifier but also to change it or clear it.

Note that a performance monitor application does not necessarily represent a particular ALCS application or application group. It could represent the terminal type or origin city, or anything that can be used to identify an entry. The application must be pre-defined to the performance monitor to be valid. An entry without a valid application identifier will have its consumption data collected using the default category in the performance monitor tables.

APIDC converts the performance monitor application name to a numeric index and stores it in the ECB descriptor. This index is used to directly access the application item in the performance monitor tables during the collection.

There are three **levels** of application identification, allowing an entry to be identified in up to 3 ways (perhaps by origin, application suite, and action code).

Performance monitor application name SYST is used to record all the entries. DFT1, DFT2, DFT3 are the names of the default applications for each level - the entry will be recorded in these applications if it does not use APIDC to assign a specific application identification for the level. And you can use the ZPERF FORCE command to cause an entry to be recorded in the PERF application.

History File

The ALCS performance monitor stores historical data on the ALCS real-time database in **history file** records. It stores data for the current year and up to five previous years. You can use installation-wide exit program AFP1 to change the number of previous years that are retained.

Data is saved from the system-wide performance monitor table (PM0TB) and from the system load information table (SL0IT). The system load information table contains data that is displayed with the ZSTAT command.

The mean values for each 1-hour interval are recorded and saved on file. This data can be kept for up to 5 years.

Data is collated every 5 seconds from the PM0TB and SL0IT into an interim table that is filed. This interim data is summed and divided by the number of interim recordings in an interval (1 hour) to derive the mean values that are recorded in the history file records.

The history file is stored on the ALCS real-time database. The master index, in keypoint record #KPTRI(15), contains an item for each month/year that points to an L1 pool record that has an index for 31 day items and an average day item. Each of these items in turn points to an L1 pool record that contains an index for 24 hour items and an average hour item. Each of these items in turn points to an L3 pool record that contains the performance monitor system information for the hour. This L3 record points, via the forward chain, to the performance monitor application records for the hour. There can be between 1 and 4 application records depending on the number of user-defined performance monitor applications.

[5] The performance monitor application names do not have to match the application names defined by ALCS communication generation COMDEF macros. You may find it convenient to use the same names or you may prefer to use a different set of names for the performance monitor applications.

Each application record can hold data for 39 user-defined performance monitor applications.

The total number of performance monitor applications (including the pre-defined applications SYST, DFT1, DFT2, and DFT3) must not exceed 156.

System Performance Indicators

A SPI is an indicative number calculated by the performance monitor. There is no ideal value for these numbers and we do not suggest you attempt to tune them to any value. What they do is allow for you to determine if the system is operating in the same way day to day. Any SPI should remain within a narrow range for the normal running of the system. A dramatic change to a SPI implies that the ALCS system and/or applications have changed and now behave differently.

SPI Application

The Application SPI is an indicator of the amount of CPU required by the entries. It takes the CPU percentage used and divides it by the mean number of ECBs per second for the period recorded by the performance monitor.

The value is:

$$\frac{\%\ \text{CPU}}{\text{number of ECBs per second}}$$

SPI CPU

The CPU SPI is an indicator of the interference, number of interruptions, within the system. The value will fluctuate with changes in the system environment such as changes in LPAR priority, task priority, or I/O activity.

The value is:

$$\frac{(100 - (\%\ \text{WAIT} + \%\ \text{CPU}))}{\%\ \text{CPU}}$$

SPI I/O

The I/O SPI is a measure of ALCS I/O performance when under load. It is showing the gradient of the curve of I/O queue time over I/O rate. When the system is busy and functioning well the value will not fluctuate much but at periods of low transaction rates or if the DASD subsystem becomes overloaded then this figure will vary.

The value is:

$$\frac{\text{I/O Queue Time}}{\text{I/O Rate}}$$

Note that this is a linear relationship but does not pass through origin, I/O Queue time does not tend to zero as the rate does so the value is not constant with load.

System I/O system performance indicator

The System I/O performance indicator shows what percentage of ALCS I/O is not directly driven by the entries but by the system to perform background tasks and housekeeping.

The value is:

$$\frac{100 \times (\text{Total I/O} - \text{Sum of (Total ECB Reads + Total ECB Writes)})}{\text{Sum of (Total ECB Reads + Total ECB Writes)}}$$

Message system performance indicator

The message SPI shows the input message rate per ECB.

VFA system performance indicator

The VFA SPI takes the ECB collected figures for finds, reads, files and writes to indicate the efficiency of VFA.

The value is:

$$\frac{100 \times ((\text{Total ECB Reads} - \text{Total ECB VFA Reads}) + (\text{Total ECB Writes} - \text{Total ECB VFA Writes}))}{\text{Sum of (Total ECB Reads + Total ECB Writes)}}$$

ALCS Throttle

The ALCS throttle THRTC macro is a service to control resources used by "batch-like" applications.

During the processing of an entry control is transferred by ALCS to various applications, which can call the THRTC service. The THRTC service make use of a throttle table, which contains values (the normal and restricted NLOOP, WAIT1, WAIT2, and TIME values) and thresholds (CPU, IOQ, USR) for each throttle application. Those application names, values, and thresholds are based on rules designed by you (the user) and you can display, add, delete, and update them with help of the ZCTHR command.

`THRTC WAIT,`**`NAME`**=*nnnn* can help you to avoid performance degradation. The first time you call the `THRTC WAIT,`**`NAME`**=*nnnn* service for an entry, THRTC will copy values such as NLOOP, WAIT1, and WAIT2 defined in the throttle table for application *nnnn* to your ECB.

When you call `THRTC WAIT,`**`NAME`**=*nnnn* again, then you will wait. Normally you will wait WAIT1 seconds or milliseconds (as defined in the throttle table and copied to your ECB), but after calling `THRTC WAIT,`**`NAME`**=*nnnn* NLOOP times you will wait WAIT1 + WAIT2 seconds or milliseconds (as defined in the throttle table and copied to your ECB) but after calling `THRTC WAIT,`**`NAME`**=*nnnn* NLOOP times you will wait WAIT1 + WAIT2 seconds or milliseconds. Note that NLOOP with value zero implies that you always will wait WAIT1 + WAIT2 seconds or milliseconds.

When adding or updating the application entry in the ALCS throttle with the ZCTHR command, use the TIME parameter to specify the unit of the wait time as either seconds or milliseconds. The defined wait time unit applies both to WAIT1 and WAIT2 waits.

If one of the non-zeroes thresholds defined in the throttle table is exceeded, then the "restricted" NLOOP, WAIT1, WAIT2, TIME values are used. Otherwise the "normal" values are used (note that a threshold with value zero implies that there is NO threshold). The CPU utilization (a percentage) and IOQ (Average I/O queue time in milliseconds) thresholds are continuously compared with values obtained by the performance monitor. The USR threshold is compared with a value obtained by the throttle. In order to obtain this value the throttle will invoke the installation-wide ECB-controlled exit ATH1 (if that exit is loaded).

`THRTC LOAD,`**`NAME`**=*nnnn* will return the NLOOP value (either the normal or restricted value) as a response.

For additional information refer to “ZCTHR -- Control and display the ALCS Throttle” on page 116.

Improving application performance

Even small improvements to frequently used application programs can have a significant effect on system performance. Two areas that can repay examination are:

- **Storage serialization.** In a multiprocessor system, you should minimize storage serialization in order to avoid degrading system performance. The `BEGIN` macro parameters, XCL= and SHR=, control the serialization of the global area in ALCS. If the XCL= and SHR= parameters are omitted, all global area access by application programs is serialized, as described in *ALCS Installation and Customization*.

  It can be a significant effort to examine every application program and code the `BEGIN` macro parameters. A more productive method can be to use the report on "Program Usage Mix by Called Program". Starting with the most frequently called program, check to see whether it updates the global area. If not, code `XCL=NONE`. Check also to see if it references the global area. If not, code `SHR=NONE`.
- **Program calls.** Reducing the number of program calls improves the performance of an application. The report on "Program Usage Mix by Called Program" can show programs that only one program calls. This is often because the called program is an overflow to the calling program. Where possible, merge these programs to reduce the number of program calls. Programs that run under ALCS can be up to 4KB in 24-bit mode and up to 32KB in 31-bit mode.

This report can also show other program call abnormalities. For example, a frequently called program with a high percentage of the calls from an infrequently called program. You can often improve performance by moving or copying a routine from one of the programs to the other.

Message mix

The "Medium Speed Message Mix by Action Code" report shows how the end users use the system. It can indicate the percentage of messages for a particular application (for example, to assess the use of a new application).

The report shows only the first action code in a message. For example, an airline agent can enter the following:

- Name
- Received from
- Phone number
- Ticketing details

as four separate messages or as one message. If the agent enters it as one message, the "end item" key separates each part of the message. In this case the report shows the action code for the name message only. The absence of action codes for the received from, phone number, and ticketing messages indicates that airline agents are using the reservation system efficiently.

Note: The ECB processing summary (see “ECB processing summary” on page 37) shows all the FINDs, FILEs, ENTERs, and MILLISECS LIFE for the four message parts against the first action code (in the above example, the name action code).

Record access mix by record type

This report identifies the most frequently accessed record types. Consider changing the VFA option for the record types frequently written to DASD to delayed file or time-initiated file to improve the VFA hit rate for DASD writes.

Record access mix by record ID

This report can highlight abnormal conditions; for example, an unexpectedly high access level for a record type may indicate bad application design.

You can also use this report to identify good candidates for short-term (rather than long-term) pool.

Record ID 0000 represents pool file records that ALCS reads to check that they are available before dispensing them.

Applying corrective service

You will receive corrective service from IBM as the product is enhanced. We strongly recommend that all corrective service is applied to your system as soon as possible. You may use the ISPF panels to assist in the application of corrective service changes.

Using ISPF panels to apply corrective service to ALCS

About this task

```
  File  Options  Help
------------------------------------------------------------------------
                          ALCS primary menu
Command ===> ____________________________________________________________

Choose one of the following actions, then press ENTER


Actions:
__  1.  Installation
    2.  Generation and database creation
    3.  Maintenance
    4.  Operations
    5.  Application development

ALCS system . . SMPE      IBM-supplied SMP/E system definitions
```

Figure 47. ISPF panel: ALCS Primary Menu

Select option 3 to display the Maintenance panel which is shown in Figure 48 on page 399.

```
                              Maintenance
Command ===> ____________________________________________________________

Choose one of the following actions, then press ENTER


Actions:
__  1.  Receive corrective service (online)
    2.  Apply corrective service (batch)
    3.  Accept corrective service (batch)
    4.  Restore corrective service (batch)
    5.  Reject corrective service (online)
    6.  List details of corrective service (online)
    7.  Browse service log
```

Figure 48. ISPF panel: Maintenance

Select the appropriate option to take you to the panel that you wish to use.

The standard method of manually applying all SYSMODs that affect ALCS is the SMP/E "RECEIVE/ACCEPT BYPASS (Apply Check)/System Generation/JCLIN" method. This applies to IBM supplied corrective service (APAR fixes) and IBM supplied preventive service (PTFs), as well as USERMODs. To apply one or more SYSMODs:

Procedure

1. RECEIVE the SYSMOD.
2. APPLY the SYSMOD to a test ALCS system. It is advisable to test any modification to ALCS in a test environment before implementing it on the production system. To set up a test environment, define a test ALCS system to SMP/E in a separate target zone, and maintain a separate complete set of target libraries (and optionally distribution libraries).
3. Use the test system to test the modifications introduced by these SYSMODs. To resolve any outstanding problems:

a) RESTORE, and if necessary REJECT the USERMOD, correct the changes and then RECEIVE and APPLY again.

b) Construct, RECEIVE, and APPLY additional USERMODs. This can include replacement USERMODs for those regressed by the application of IBM service.

   USERMODs are change decks. If a USERMOD has been APPLYed to an IBM component, it must be RESTORed and (and, if necessary) REJECTed before the IBM service can be APPLYed. New USERMODs must now be built and APPLYed taking into account any changes made by the IBM corrective service.

c) APPLY additional IBM corrective or preventive service.

4. When the changes are complete and error free, APPLY the SYSMOD to the production ALCS target zone and target libraries.

Results

For a complete description of SMP/E commands, and details of USERMOD construction and naming, see:

- *System Modification Program Extended (SMP/E) User's Guide*
- *System Modification Program Extended (SMP/E) Reference*.

Appendix A. Command synonyms

Table 24 on page 401 lists TPF, ALCS/VSE, and ALCS/MVS/XA commands that have ALCS Version 2 equivalents.

Table 24. TPF, ALCS/VSE, ALCS/MVS/XA commands and their ALCS V2 equivalent

TPF	ALCS/VSE 1.4	ALCS/MVS/XA	ALCS V2	ALCS V2 Synonym
--	--	ZBTAP	ZACOM	ZBTAP
--	--	ZCLNK	ZACOM	ZCLNK
--	--	--	ZAKEY	--
--	ZHALT	ZASYS HALT	ZASYS HALT	ZHALT
--	--	--	ZCMQI	--
--	--	--	ZCMSP	--
--	--	--	ZCSQL	--
--	--	--	ZCTCP	--
--	ZDATA	ZDATA	ZDATA	ZRORG
--	--	--	ZDKEY	--
--	--	--	ZDPDU	--
--	--	--	ZHELP	--
--	--	--	ZLOGN	--
--	--	--	ZLOGF	--
--	--	--	ZMAIL	--
--	--	--	ZOCTM	--
--	--	--	ZRELO	--
--	--	--	ZRETR	--
LOG*x*	--	ZROUT	ZROUT	ZLOGI
Z *xyy*	--	--	ZSCRL	--
ZACLV	ZAACV	ZAACV	ZAACV	ZAACP
ZACOR, ZADCA	ZACOR	ZACOR	ZACOR	ZASTG
ZACRS	ZACRS, ZACRT	ZACOM	ZACOM	--
ZAFIL	ZAFIL	ZAFIL	ZAFIL	--
ZAMOD	--	ZDASD	ZDASD	--
ZAPGM	ZAPRG	ZAPRG	ZAPRG	ZAPGM
ZAREC	ZAFIL	ZAFIL	ZAFIL	--
ZASER	ZASER	ZASER	ZASER	--
ZATIM	ZATIM	ZATIM	ZATIM	--
ZCYCL	ZASYS	ZASYS	ZASYS	ZCYCL
ZDADD	--	ZDFIL,ZDASD	ZDFIL,ZDASD	--
ZDCLV	ZDACV	ZDACV	ZDACV	ZDACP

Table 24. TPF, ALCS/VSE, ALCS/MVS/XA commands and their ALCS V2 equivalent (continued)

TPF	ALCS/VSE 1.4	ALCS/MVS/XA	ALCS V2	ALCS V2 Synonym
ZDCRS	ZDCRS, ZDCRT	ZDCOM	ZDCOM	--
ZDCRS	ZDMTA	ZDCOM	ZDCOM	--
ZDDAT	ZDTIM	ZDTIM	ZDTIM	--
ZDDCA	ZDCOR	ZDCOR	ZDCOR	ZDSTG
ZDFIL	ZDFIL	ZDFIL	ZDFIL	--
ZDFPC	ZDFPC	ZDFPC	ZPOOL	ZDFPC, ZDPFC
ZDMFS	--	--	ZDASD	--
ZDMOD	--	--	ZDASD	--
ZDMSG	--	--	ZDASD	--
ZDPGM	ZDPRG	ZDPRG	ZDPRG	ZDPGM
ZDREC	ZDFIL	ZDFIL	ZDFIL	--
ZDSER	ZDSER	ZDSER	ZDSER	--
ZDSTB	ZDTAP	ZDSEQ	ZDSEQ	ZDTAP
ZDSYS	ZDSYS	ZDSYS	ZDSYS	--
ZDTAP	ZDTAP	ZDSEQ	ZDSEQ	ZDTAP
ZDTIM	ZDTIM	ZDTIM	ZDTIM	--
ZDTLB	--	ZDSEQ	ZDSEQ	ZDTAP
ZDUMP	ZDUMP	ZDUMP	ZDUMP	--
ZDVSN	--	ZDASD	ZDASD	--
ZECBL	--	--	ZDECB, ZPURG	ZDMAP, ZKILL
ZFDNT	--	--	ZDASD	--
ZFMNT	--	--	ZDASD	--
ZFRST	ZRSTR	ZRSTR	ZRSTR	--
ZGAFA, ZGAFI	ZGAFA	ZGAFA	ZGAFA	--
ZGFSP	ZSGFS	ZPOOL	ZPOOL	ZSGFS
ZIUMP	--	--	ZSNDU	--
ZLDLS	ZLDLS	ZLDLS	ZDCOM	ZLDLS
ZLIDL	ZLCLS	ZLCLS	ZACOM	ZLCLS
ZLKST	ZLKST	ZLKST	ZLKST	ZLKST
ZLKTF	ZLKTF	ZLKTR	ZLKTR	--
ZLKTN	ZLKTN	ZLKTR	ZLKTR	--
ZLREP, ZLVAL	ZLOPN	ZLOPN	ZACOM	ZLOPN
ZLSTA	ZLSTA	ZACOM	ZACOM	ZLSTA
ZLSTP	ZLSTP	ZACOM	ZACOM	ZLSTP
ZLTST	ZLTST	ZLTST	ZLTST	--
ZMCHR, ZMCPY	-	ZDASD	ZDASD	--
ZMEAS	ZDCLR	ZDCLR	ZDCLR	--

Table 24. TPF, ALCS/VSE, ALCS/MVS/XA commands and their ALCS V2 equivalent (continued)

TPF	ALCS/VSE 1.4	ALCS/MVS/XA	ALCS V2	ALCS V2 Synonym
ZNDLU	--	ZDCOM	ZDCOM	--
ZNPRG, ZNRPT	--	ZCPRT	ZACOM	ZCPRT
ZOLDR	--	ZPCTL	ZPCTL	--
ZPLMT	ZPLMT	ZCPRT	ZACOM	ZCPRT
ZPTAP	ZDTAP	ZDSEQ	ZDSEQ	ZDTAP
ZPUMP	--	--	ZSNDU	--
ZRCRS	ZRCRS	ZRCRS	ZRCRS	--
ZRECP	ZRECP	ZRECP	ZRECP	--
ZRLMT	ZRLMT	ZCPRT	ZACOM	ZCPRT
ZROUT	--	ZACOM	ZACOM	--
ZSLDR	--	ZDATA	ZDATA	--
ZSNDU	ZSNDU	ZSNDU	ZSNDU	--
ZSTAT	ZSTAT	ZSTAT	ZSTAT	--
ZSTOP	ZTRAC	ZTRAC	ZTRAC	--
ZSTVS	ZTEST	ZTEST	ZTEST	--
ZTERM	ZDMTA	ZDCOM	ZDCOM	--
ZTEST	ZDRIV	ZDRIV	ZDRIV	--
ZTLBL	ZTLBL	ZASEQ	ZASEQ	--
ZTOFF	ZTCLS	ZCSEQ	ZCSEQ	ZTCLS
ZTPSW	ZTPSW	ZSSEQ	ZSSEQ	ZTPSW
ZTRAC	ZTRAC	ZTRAC	ZTRAC	--
ZTSTB	ZDTAP	ZDSEQ	ZDSEQ	ZDTAP

Appendix B. Acronyms and abbreviations

The following acronyms and abbreviations are used in books of the ALCS Version 2 library. Not all are necessarily present in this book.

AAA
agent assembly area

ACB
VTAM access method control block

ACF
Advanced Communications Function

ACF/NCP
Advanced Communications Function for the Network Control Program, usually referred to simply as "NCP"

ACF/VTAM
Advanced Communications Function for the Virtual Telecommunication Access Method, usually referred to simply as "VTAM"

ACK
positive acknowledgment (SLC LCB)

ACP
Airline Control Program

AID
IBM 3270 attention identifier

AIX®
Advanced Interactive eXecutive

ALC
airlines line control

ALCI
Airlines Line Control Interconnection

ALCS/MVS/XA
Airline Control System/MVS/XA

ALCS/VSE
Airline Control System/Virtual Storage Extended

ALCS V2
Airline Control System Version 2

AML
acknowledge message label (SLC LCB)

AMS
access method services

AMSG
AMSG application message format

APAR
authorized program analysis report

APF
authorized program facility

API
application program interface

APPC
advanced program-to-program communications

ARINC
Aeronautical Radio Incorporated

ASCU
agent set control unit (SITA), a synonym for "terminal control unit"

AT&T
American Telephone and Telegraph Co.

ATA
Air Transport Association of America

ATSN
acknowledge transmission sequence number (SLC)

BATAP
Type B application-to-application program

BSC
binary synchronous communication

C
C programming language

CAF
Call Attach Facility

CCW
channel command word

CDPI
clearly differentiated programming interface

CEC
central electronic complex

CEUS
communication end-user system

CI
VSAM control interval

CICS®
Customer Information Control System

CLIST
command list

CMC
communication management configuration

CML
clear message label (synonym for AML)

COBOL
COmmon Business Oriented Language

CPI-C
Common Programming Interface - Communications

CPU
central processing unit

CRAS
computer room agent set

CRI
communication resource identifier

CRN
communication resource name

CSA
common service area

CSECT
control section

CSID
cross system identifier

CSW
channel status word

CTKB
Keypoint record B

CTL
control system error

CUA
Common User Access

DASD
direct access storage device

DBCS
double-byte character set

DBRM
DB2 database request module

DB2
IBM DB2 for z/OS (refers to DB2)

DCB
data set control block

DECB
ALCS data event control block

DF
delayed file record

DFDSS
Data Facility Data Set Services

DFHSM
Data Facility Hierarchical Storage Manager

DFP
Data Facility Product

DFSMS
Data Facility Storage Management Subsystem

DFT
distributed function terminal

DIX
delete item index

DRIL
data record information library

DSI
direct subsystem interface

DSECT
dummy control section

DTP
ALCS diagnostic file processor

EBCDIC
extended binary-coded decimal interchange code

ECB
ALCS entry control block

EIB
error index byte

EID
event identifier

EJB
Enterprise Java Bean

ENQ
enquiry (SLC LCB)

EOF
end of file

EOM
end of message

EOI
end of message incomplete

EOP
end of message pushbutton

EOU
end of message unsolicited

EP
Emulation Program

EP/VS
Emulation Program/VS

ETX
end of text

EvCB
MVS event control block

EXCP
Execute Channel Program

FACE
file address compute

FIFO
first-in-first-out

FI
file immediate record

FM
function management

FMH
function management header

GB
gigabyte (1 073 741 824 bytes)

GDS
general data set

GFS
get file storage (called pool file storage in ALCS)

GMT
Greenwich Mean Time

GTF
generalized trace facility (MVS)

GUPI
general-use programming interface

HEN
high-level network entry address

HEX
high-level network exit address

HFS
Hierarchical File System

HLASM
High Level Assembler

HLL
high-level language

HLN
high-level network

HLS
high-level system (for example, SITA)

HTTP
Hypertext Transfer Protocol

IA
interchange address

IASC
International Air Transport Solution Centre

IATA
International Air Transport Association

IATA5
ATA/IATA transmission code 5

IATA7
ATA/IATA transmission code 7

ICF
integrated catalog facility

ICSF
Integrated Cryptographic Service Facility

ID
identifier

ILB
idle (SLC LCB)

IMA
BATAP acknowledgement

IMS
Information Management System

IMSG
IMSG input message format

I/O
input/output

IOCB
I/O control block

IP
Internet Protocol

IPARS
International Programmed Airlines Reservation System

IPCS
Interactive Problem Control System

IPL
initial program load

ISA
initial storage allocation

ISC
intersystem communication

ISO/ANSI
International Standards Organization/American National Standards Institute

ISPF
Interactive System Productivity Facility

ISPF/PDF
Interactive System Productivity Facility/Program Development Facility

ITA2
International Telegraph Alphabet number 2

JCL
job control language

JES
job entry subsystem

JNDI
Java Naming and Directory Interface

JSON
JavaScript Object Notation

KB
kilobyte (1024 bytes)

KCN
link channel number (SLC)

KSDS
VSAM key-sequenced data set

LAN
local area network

LCB
link control block (SLC)

LDB
link data block (SLC)

LDI
local DXCREI index

LEID
logical end-point identifier

LE
Language Environment®

LICRA
Link Control - Airline

LMT
long message transmitter

LN
line number (ALCS/VSE and TPF terminology)

LN/ARID
line number and adjusted resource identifier (ALCS/VSE terminology)

LSET
Load set

LSI
link status identifier (SLC)

LU
logical unit

LU 6.2
Logical Unit 6.2

MATIP
Mapping of airline traffic over IP

MB
megabyte (1 048 576 bytes)

MBI
message block indicator (SLC)

MCHR
module/cylinder/head/record

MESW
message switching

MNOTE
message note

MQI
Message Queueing Interface

MQM
Message Queue Manager

MSNF
Multisystem Networking Facility

MVS
Multiple Virtual Storage (refers to MVS) (refers to both MVS/XA and MVS/ESA, and also to OS/390® and z/OS)

MVS/DFP
Multiple Virtual Storage/Data Facility Product

MVS/ESA
Multiple Virtual Storage/Enterprise System Architecture

MVS/XA
Multiple Virtual Storage/Extended Architecture

NAB
next available byte

NAK
negative acknowledgment (SLC LCB)

NCB
network control block (SLC)

NCP
Network Control Program (refers to ACF/NCP)

NCP/VS
Network Control Program/Virtual Storage.

NEF
Network Extension Facility

NEF2
Network Extension Facility 2

NPDA
Network Problem Determination Application

NPSI
Network Control Program packet switching interface

NTO
Network Terminal Option

OCR
one component report

OCTM
online communication table maintenance

OLA
optimized local adapters

OMSG
OMSG output message format

OPR
operational system error

OSID
other-system identification

OS/2
IBM Operating System/2®

PARS
Programmed Airlines Reservation System

PDF
parallel data field (refers to NCP)

PDM
possible duplicate message

PDS
partitioned data set

PDSE
partitioned data set extended

PDU
pool directory update

PER
program event recording

PFDR
pool file directory record

PL/I
programming language one

PLM
purge long message (name of ALCS/VSE and TPF general tape)

PLU
primary logical unit

PNL
passenger name list

PNR
passenger name record

PP
IBM program product

PPI
program-to-program interface

PPMSG
program-to-program message format

PPT
program properties table

PR
permanently resident record

PRC
prime computer room agent set

PRDT
physical record (block) descriptor table

PRPQ
programming request for price quotation

PR/SM
Processor Resource/Systems Manager

PS
VTAM presentation services

PSPI
product sensitive programming interface

PSW
program status word

PTF
program temporary fix

PTT
Post Telephone and Telegraph Administration

PU
physical unit

PVC
permanent virtual circuit

QSAM
queued sequential access method

RACF®
resource access control facility

RB
request block

RBA
relative byte address

RCC
record code check

RCPL
routing control parameter list

RCR
resource control record

RCS
regional control center

RDB
Relational Database

RDBM
Relational Database Manager

REI
resource entry index

RLT
record locator table

RMF
Resource Measurement Facility

RO CRAS
receive-only computer room agent set

RON
record ordinal number

RPL
VTAM request parameter list

RPQ
request for price quotation

RSM
resume (SLC LCB)

RTM
recovery and termination management

RU
request unit

SAA
Systems Application Architecture®

SAL
system allocator list (TPF terminology)

SAM
sequential access method

SDLC
Synchronous Data Link Control

SDMF
standard data and message file

SDSF
System Display and Search Facility

SDWA
system diagnostic work area

SI
DBCS shift in

SITA
Société Internationale de Télécommunications Aéronautiques

SLC
ATA/IATA synchronous link control

SLIP
serviceability level indication processing

SLN
symbolic line number

SLR
Service Level Reporter

SLU
secondary logical unit

SMP/E
System Modification Program Extended

SNA
Systems Network Architecture

SO
DBCS shift out

SON
system ordinal number

SQA
system queue area

SQL
Structured Query Language

SQLCA
SQL Communication Area

SQLDA
SQL Descriptor Area

SRB
service request block

SRG
statistical report generator

SRM
System Resource Manager

STC
system test compiler

STP
stop (SLC LCB)

STV
system test vehicle

SWB
service work block

SYN
character synchronization character

TA
terminal address

TAS
time available supervisor

TCB
task control block

TCID
terminal circuit identity

TCP/IP
Transmission Control Protocol / Internet Protocol

TI
time-initiated record

TOD
time of day

TPF
Transaction Processing Facility

TPF/APPC
Transaction Processing Facility/Advanced Program to Program Communications

TPF/DBR
Transaction Processing Facility/Data Base Reorganization

TPFDF
TPF Database Facility

TPF/MVS
Transaction Processing Facility/MVS (alternative name for ALCS V2)

TP_ID
transaction program identifier

TSI
transmission status indicator

TSN
transmission sequence number

TSO
time-sharing option

TSO/E
Time Sharing Option Extensions

TUT
test unit tape (sequential file)

UCB
unit control block

UCTF
Universal Communications Test Facility

VFA
virtual file access

VIPA
virtual IP address

VM
virtual machine

VM/CMS
virtual machine/conversational monitor system

VS
virtual storage

VSAM
virtual storage access method

VSE
Virtual Storage Extended

VSE/AF
Virtual Storage Extended/Advanced Function

VSE/VSAM
Virtual Storage Extended/Virtual Storage Access Method

VTAM
Virtual Telecommunications Access Method (refers to

VTOC
volume table of contents

WAS
WebSphere Application Server (refers to WebSphere)

WSF
Write Structured Field

WTTY
World Trade Teletypewriter

XMSG
XMSG message switching message format

XREF
ALCS cross referencing facility

Glossary

Notes:

1. Acronyms and abbreviations are listed separately from this Glossary. See Appendix B, “Acronyms and abbreviations,” on page 404.
2. For an explanation of any term not defined here, see the IBM *Dictionary of Computing*.

A

AAA hold
See terminal hold.

abnormal end of task (abend)
Termination of a task before its completion because of an error condition that cannot be resolved by recovery facilities while the task is executing.

access method services (AMS)
A utility program that defines VSAM data sets (or files) and allocates space for them, converts indexed sequential data sets to key-sequenced data sets with indexes, modifies data set attributes in the catalog, facilitates data set portability between operating systems, creates backup copies of data sets and indexes, helps make inaccessible data sets accessible, and lists data set records and catalog entries.

activity control variable
A parameter that ALCS uses to control its workload. The system programmer defines activity control variables in the ALCS system configuration table generation.

Advanced Communications Function for the Network Control Program (ACF/NCP)
An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability.

Advanced Program-to-Program Communications (APPC)
A set of inter-program communication services that support cooperative transaction processing in an SNA network. APPC is the implementation, on a given system, of SNA's logical unit type 6.2 (LU 6.2). See APPC component and APPC transaction scheduler.

Aeronautical Radio Incorporated (ARINC)
An organization which provides communication facilities for use within the airline industry.

agent assembly area (AAA)
A fixed-file record used by IPARS applications. One AAA record is associated with each terminal and holds data that needs to be kept beyond the life of an entry. For example, to collect information from more than one message.

agent set
Synonym for communication terminal.

agent set control unit (ASCU)
Synonym for terminal interchange.

Airline Control Program (ACP)
An earlier version of the IBM licensed program Transaction Processing Facility (TPF).

Airline Control System (ALCS)
A transaction processing platform providing high performance, capacity, and availability, that runs specialized (typically airline) transaction processing applications.

Airline Control System/Multiple Virtual Storage/Extended Architecture (ALCS/MVS/XA)
An ALCS release designed to run under an MVS/XA operating system.

Airline Control System Version 2 (ALCS V2)
An ALCS release designed to run under a z/OS operating system.

Airline Control System/Virtual Storage Extended (ALCS/VSE)
An ALCS release designed to run under a VSE/AF operating system.

airlines line control (ALC)
A communication protocol particularly used by airlines.

Airlines Line Control Interconnection (ALCI)
A feature of Network Control Program (NCP) that allows it to manage ALC networks in conjunction with a request for price quotation (RPQ) scanner for the IBM 3745 communication controller.

Airline X.25 (AX.25)
A discipline conforming to the ATA/IATA AX.25 specification in the ATA/IATA publication *ATA/IATA Interline Communications Manual*, ATA/IATA document DOC.GEN 1840. AX.25 is based on X.25 and is intended for connecting airline computer systems to SITA or ARINC networks.

ALCS command
A command addressed to the ALCS system. All ALCS commands start with the letter Z (they are also called "Z messages") and are 5 characters long.

These commands allow the operator to monitor and control ALCS. Many of them can only be entered from CRAS terminals. ALCS commands are called "functional messages" in TPF.

ALCS data collection file
A series of sequential data sets to which ALCS writes performance-related data for subsequent processing by the statistical report generator or other utility program. See also data collection and statistical report generator.

ALCS diagnostic file
A series of sequential data sets to which the ALCS monitor writes all types of diagnostic data for subsequent processing by the diagnostic file processor.

ALCS diagnostic file processor
An offline utility, often called the "post processor", that reads the ALCS diagnostic file and formats and prints the dump, trace, and system test vehicle (STV) data that it contains.

ALCS entry dispatcher
The ALCS online monitor's main work scheduler. Often called the "CPU loop".

ALCS offline program
An ALCS program that runs as a separate MVS job (not under the control of the ALCS online monitor).

ALCS online monitor
The part of ALCS that performs the services for the ECB-controlled programs and controls their actions.

ALCS trace facility
An online facility that monitors the execution of application programs. When it meets a selected monitor-request macro, it interrupts processing and sends selected data to an ALCS display terminal, to the ALCS diagnostic file, or to the system macro trace block. See also instruction step.

The ALCS trace facility also controls tracing to the MVS generalized trace facility (GTF), for selected VTAM communication activity, and controls tracing of input and output messages to a (wrap around) online trace area for selected communication resources.

ALCS update log file
A series of sequential data sets in which the ALCS monitor records changes to the real-time database.

ALCS user file
A series of sequential data sets to which you may write all types of diagnostic data for subsequent processing by an offline processor. You write the data from an installation-wide monitor exit using the callable service UWSEQ.

allocatable pool
The ALCS record class that includes all records on the real-time database. Within this class, there is one record type for each DASD record size.

The allocatable pool class is special in that ALCS itself can dispense allocatable pool records and use them for other real-time database record classes. For example, all fixed-file records are also allocatable pool records (they have a special status of "in use for fixed file").

When ALCS is using type 2 long-term pool dispense, ALCS satisfies requests for long-term pool by dispensing available allocatable pool records.

See DASD record, real-time database, record class, and record type.

alternate CRAS
A computer room agent set (CRAS) that is not Prime CRAS or receive only CRAS. See computer room agent set, Prime CRAS, and receive only CRAS.

alternate CRAS printer
A CRAS printer that is not receive only CRAS. See CRAS printer and receive only CRAS.

answerback
A positive acknowledgement (ACK) from an ALC printer.

APPC component
The component of MVS that is responsible for extending LU 6.2 and SAA CPI Communications services to applications running in any MVS address space. Includes APPC conversations and scheduling services.

APPC transaction scheduler
A program such as ALCS that is responsible for scheduling incoming work requests from cooperative transaction programs.

application plan
See DB2 application plan.

application
A group of associated application programs that carry out a specific function.

application global area
An area of storage in the ALCS address space containing application data that any entry can access.

The application global area is subdivided into keypointable and nonkeypointable records. Keypointable records are written to the database after an update; nonkeypointable records either never change, or are reinitialized when ALCS restarts.

C programs refer to global records and global fields within the application global area.

application program
A program that runs under the control of ALCS. See also ECB-controlled program.

application program load module
In ALCS, a load module that contains one or more application programs.

application queue
In message queuing with ALCS, any queue on which application programs put and get messages using MQI calls.

assign
Allocate a general sequential file to an entry. The TOPNC monitor-request macro (or equivalent C function) opens and allocates a general sequential file. The TASNC monitor-request macro (or equivalent C function) allocates a general sequential file that is already open but not assigned to an entry (it is reserved).

associated resource
Some ALCS commands generate output to a printer (for example, ZDCOM prints information about a communication resource). For this type of command the printed output goes to the associated resource; that is, to a printer associated with the originating display. There is also a response to the originating display that includes information identifying the associated resource.

asynchronous trace
One mode of operation of the ALCS trace facility. Asynchronous trace is a conversational trace facility to interactively trace entries that do not originate from a specific terminal.

automatic storage block
A storage block that is attached to an entry, but is not attached at a storage level. An assembler program can use the ALASC monitor-request macro to obtain an automatic storage block and BACKC monitor-request macro to release it. C programs cannot obtain automatic storage blocks.

B

backward chain
The fourth fullword of a record stored on the ALCS database, part of the record header. See chaining of records.

When standard backward chaining is used, this field contains the file address of the previous record in the chain, except that the first record contains the file address of the last record in the chain. (If there is only one record, the backward chain field contains zeros.)

balanced path
A path where no single component (channel, DASD director or control unit, head of string, and internal path to the DASD device) is utilized beyond the limits appropriate to the required performance.

bar
In the MVS 64-bit address space, a virtual line called the bar marks the 2-gigabyte address. The bar separates storage below the 2-gigabyte address, called **below the bar**, from storage above the 2-gigabyte address, called **above the bar**.

BATAP
Type B application-to-application program

Binary Synchronous Communication (BSC)
A form of telecommunication line control that uses a standard set of transmission control characters and control character sequences, for binary synchronous transmission of binary-coded data between stations.

bind
See DB2 bind

BIND
In SNA, a request to activate a session between two logical units (LUs). The BIND request is sent from a primary LU to a secondary LU. The secondary LU uses the BIND parameters to help determine whether it will respond positively or negatively to the BIND request.

binder
The program that replaces the linkage editor and batch loader programs that were provided with earlier versions of MVS.

BIND image
In SNA, the set of fields in a BIND request that contain the session parameters.

block
See storage block.

C

catastrophic
A type of system error that results in the termination of ALCS.

chain-chase
See Recoup.

chaining of records
One record can contain the file address of another (usually a pool-file record). The addressed record is said to be chained from the previous record. Chains of records can contain many pool-file records. See forward chain and backward chain.

class
See record class.

clearly differentiated programming interfaces (CDPI)
A set of guidelines for developing and documenting product interfaces so that there is clear differentiation between interfaces intended for general programming use (GUPIs) and those intended for other specialized tasks.

close
Close a sequential file data set (MVS CLOSE macro) and deallocate it from ALCS. For general sequential files this is a function of the TCLSC monitor-request macro (or equivalent C function). ALCS automatically closes other sequential files at end-of-job.

command
See ALCS command.

command list (CLIST)
A sequential list of commands, control statements, or both, that is assigned a name. When the name is invoked the commands in the list are executed.

commit
An operation that terminates a unit of recovery. Data that was changed is now consistent.

common entry point (CEP)
A function in the Transaction Processing Facility Database Facility (TPFDF) product that provides common processing for all TPFDF macro calls issued by ALCS application programs. It also provides trace facilities for TPFDF macro calls.

Common Programming Interface - Communications (CPI-C)
The communication element of IBM Systems Application Architecture (SAA). CPI-C provides a programming interface that allows program-to-program communication using the IBM SNA logical unit 6.2.

Common User Access
Guidelines for the dialog between a user and a workstation or terminal.

communication management configuration (CMC)
A technique for configuring a network that allows for the consolidation of many network management functions for the entire network in a single host processor.

communication resource
A communication network component that has been defined to ALCS. These include each terminal on the network and other network components that ALCS controls directly (for example, SLC links). Resources can include, for example:

SNA LUs (including LU 6.1 links)
ALC terminals
SLC and WTTY links
Applications.

communication resource identifier (CRI)
A 3-byte field that uniquely identifies an ALCS communication resource. It is equivalent to the LN/IA/TA in TPF and the LN/ARID in ALCS/VSE. ALCS generates a CRI for each resource.

communication resource name (CRN)
A 1- to 8-character name that uniquely identifies an ALCS communication resource. For SNA LUs, it is the LU name. The system programmer defines the CRN for each resource in the ALCS communication generation.

communication resource ordinal
A unique number that ALCS associates with each communication resource. An installation can use the communication resource ordinal as a record ordinal for a particular fixed-file record type. This uniquely associates each communication resource with a single record.

For example, IPARS defines a fixed-file record type (#WAARI) for AAA records. Each communication resource has its own AAA record - the #WAARI record ordinal is the communication resource ordinal. See also record ordinal and agent assembly area.

compiler
A program that translates instructions written in a high level programming language into machine language.

computer room agent set (CRAS)
An ALCS terminal that is authorized for the entry of restricted ALCS commands.

Prime CRAS is the primary terminal that controls the ALCS system. Receive Only CRAS (RO CRAS) is a designated printer or NetView operator identifier to which certain messages about system function and progress are sent.

configuration data set
(1) A data set that contains configuration data for ALCS. See also configuration-dependent table .

(2) The ALCS record class that includes all records on the configuration data set. There is only one record type for this class. See record class and record type.

configuration-dependent table
A table, constructed by the ALCS generation process, which contains configuration-dependent data. Configuration-dependent tables are constructed as conventional MVS load modules. In ALCS V2, there are separate configuration-dependent tables for:

System data
DASD data
Sequential file data
Communication data
Application program data.

See also configuration data set.

control byte
The fourth byte of a record stored on the ALCS database, part of the record header. ALCS ignores this byte; some applications, however, make use of it.

control interval (CI)
A fixed-length area of direct access storage in which VSAM stores records. The control interval is the unit of information that VSAM transmits to or from direct access storage.

control transfer
The process that the ALCS online monitor uses to create a new entry and to transfer control to an ECB-controlled program.

conversation_ID:
An 8-byte identifier, used in Get_Conversation calls, that uniquely identifies a conversation. APPC/MVS returns a conversation_ID on the CMINIT, ATBALLOC, and ATBGETC calls; a conversation_ID is required as input on subsequent APPC/MVS calls.

CPU loop
See ALCS entry dispatcher.

CRAS printer
A computer room agent set (CRAS) that is a printer terminal. See computer room agent set.

CRAS display
A computer room agent set (CRAS) that is a display terminal. See computer room agent set.

CRAS fallback
The automatic process that occurs when the Prime CRAS or receive only CRAS becomes unusable by which an alternate CRAS becomes Prime CRAS or receive only CRAS. See also Prime CRAS, receive only CRAS, and alternate CRAS.

create service
An ALCS service that enables an ALCS application program to create new entries for asynchronous processing. The new ECBs compete for system resources and, once created, are not dependent or connected in any way with the creating ECB.

cycling the system
The ALCS system can be run in one of four different system states. Altering the system state is called cycling the system. See SLC link for another use of the term "cycling".

D

DASD record
A record stored on a direct access storage device (DASD). ALCS allows the same range of sizes for DASD records as it allows for storage blocks, except no size L0 DASD records exist.

data collection
An online function that collects data about selected activity in the system and sends it to the ALCS data collection file, if there is one, or to the ALCS diagnostic file. See also statistical report generator.

database request module (DBRM)
A data set member created by the DB2 precompiler that contains information about SQL statements. DBRMs are used in the DB2 bind process. See DB2 bind.

data-collection area
An ECB area used by the ALCS online monitor for accumulating statistics about an entry.

data event control block (DECB)
An ALCS control block, that may be acquired dynamically by an entry to provide a storage level and data level in addition to the 16 ECB levels. It is part of entry storage.

The ALCS DECB is independent of the MVS control block with the same name.

Data Facility Storage Management Subsystem (DFSMS)
An MVS operating environment that helps automate and centralize the management of storage. It provides the storage administrator with control over data class, management class, storage group, and automatic class selection routine definitions.

Data Facility Sort (DFSORT)
An MVS utility that manages sorting and merging of data.

data file
A sequential data set, created by the system test compiler (STC) or by the `ZDATA DUMP` command, that contains data to be loaded on to the real-time database. (An ALCS command `ZDATA LOAD` can be used to load data from a data file to the real-time database.) A data file created by STC is also called a "pilot" or "pilot tape".

data level
An area in the ECB or a DECB used to hold the file address, and other information about a record. See ECB level and DECB level.

data record information library (DRIL)
A data set used by the system test compiler (STC) to record the formats of data records on the real-time system. DRIL is used when creating data files.

DB2 application plan
The control structure produced during the bind process and used by DB2 to process SQL statements encountered during program execution. See DB2 bind.

DB2 bind
The process by which the output from the DB2 precompiler is converted to a usable control structure called a package or an application plan. During the process, access paths to the data are selected and some authorization checking is performed.

DB2 Call Attach Facility (CAF)
An interface between DB2 and batch address spaces. CAF allows ALCS to access DB2.

DB2 for z/OS
An IBM licensed program that provides relational database services.

DB2 host variable
In an application program, an application variable referenced by embedded SQL statements.

DB2 package
Also called application package. An object containing a set of SQL statements that have been bound statically and that are available for processing. See DB2 bind.

DB2 package list
An ordered list of package names that may be used to extend an application plan.

DECB level
When an application program, running under ALCS, reads a record from a file, it must "own" a storage block in which to put the record. The address of the storage block may be held in an area of a DECB called a storage level.

Similarly, there is an area in a DECB used for holding the 8-byte file address, record ID, and record code check (RCC) of a record being used by an entry. This is a data level.

The storage level and data level in a DECB, used together, are called a DECB level.

See also ECB level.

diagnostic file
See ALCS diagnostic file.

dispatching priority
A number assigned to tasks, used to determine the order in which they use the processing unit in a multitasking situation.

dispense (a pool-file record)
To allocate a long-term or short-term pool-file record to a particular entry. ALCS performs this action when requested by an application program. See release a pool-file record.

double-byte character set
A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets.

Because each character requires 2 bytes, entering, displaying, and printing DBCS characters requires hardware and supporting software that are DBCS-capable.

duplex
A communication link on which data can be sent and received at the same time. Synonymous with full duplex. Communication in only one direction at a time is called "half-duplex". Contrast with simplex transmission.

duplex database
Synonym for duplicated database.

duplicated database
A database where each data set is a mirrored pair. In ALCS, you can achieve this using either ALCS facilities or DASD controller facilities (such as the IBM 3990 dual copy facility). See mirrored pair.

dynamic program linkage
Program linkage where the connection between the calling and called program is established during the execution of the calling program. In ALCS dynamic program linkage, the connection is established by the ALCS ENTER/BACK services. Contrast with static program linkage.

dynamic SQL
SQL statements that are prepared and executed within an application program while the program is executing. In dynamic SQL, the SQL source is contained in host language variables rather than being coded into the application program. The SQL statement can change several times during the application program's execution. Contrast with embedded SQL.

E

ECB-controlled program
A program that runs under the control of an entry control block (ECB). These programs can be application programs or programs that are part of ALCS, for example the ALCS programs that process operator commands (Z messages). ECB-controlled programs are known as E-type programs in TPF.

ECB level
When an application program, running under ALCS, reads a record from file, it must "own" a storage block in which to put the record. The address of the storage block may be held in an area of the ECB called a storage level.

There are 16 storage levels in the ECB. A storage block with its address in slot zero in the ECB is said to be attached on level zero.

Similarly, there are 16 areas in the ECB that may be used for holding the 4-byte file addresses, record ID, and record code check (RCC) of records being used by an entry. These are the 16 data levels.

Storage levels and data levels, used together, are called ECB levels.

See also DECB level.

embedded SQL
Also called static SQL. SQL statements that are embedded within an application program and are prepared during the program preparation process before the program is executed. After it is prepared, the statement itself does not change (although values of host variables specified within the statement can change). Contrast with dynamic SQL.

Emulation Program/Virtual Storage (EP/VS)
A component of NCP/VS that ALCS V2 uses to access SLC networks.

ENTER/BACK
The general term for the application program linkage mechanism provided by ALCS.

entry
The basic work scheduling unit of ALCS. An entry is represented by its associated entry control block (ECB). It exists either until a program that is processing that entry issues an EXITC monitor-request macro (or equivalent C function), or until it is purged from the system. An entry is created for each input message, as well as for certain purposes unrelated to transactions. One transaction can therefore generate several entries.

entry control block (ECB)
A control block that represents a single entry during its life in the system.

entry dispatcher
See ALCS entry dispatcher.

entry macro trace block
There is a macro trace block for each entry. Each time an entry executes a monitor-request macro (or a corresponding C function), ALCS records information in the macro trace block for the entry.

This information includes the macro request code, the name of the program that issued the macro, and the displacement in the program. The ALCS diagnostic file processor formats and prints these macro trace blocks in ALCS system error dumps.

See also system macro trace block.

entry storage
The storage associated with an entry. It includes the ECB for the entry, storage blocks that are attached to the ECB or DECBs, storage blocks that are detached from the ECB or DECBs, automatic storage blocks, and DECBs. It also includes heap storage (for high-level language or assembler language programs) and stack storage (for high-level language programs).

equate
Informal term for an assignment instruction in assembler languages.

error index byte (EIB)
See SLC error index byte.

extended buffer
A storage area above 2 GB used for large messages.

extended message format
For input and output messages, a message format which includes a 4-byte field for the message length.

Execute Channel Program (EXCP)
An MVS macro used by ALCS V2 to interface to I/O subsystems for SLC support.

F

fetch access
Access which only involves reading (not writing). Compare with store access.

file address
4-byte (8 hexadecimal digits) value or 8-byte value in 4x4 format (low order 4-bytes contain a 4-byte file address, high order 4 bytes contain hexadecimal zeros) that uniquely identifies an ALCS record on DASD. `FIND/FILE` services use the file address when reading or writing DASD records. See fixed file and pool file.

file address compute routine (FACE)
An ALCS routine, called by a monitor-request macro (or equivalent C function) that calculates the file address of a fixed-file record. The application program provides the FACE routine with the fixed-file record type and the record ordinal number. FACE returns the 4-byte file address.

There is also an FAC8C monitor-request macro (or equivalent C function), that will return an 8-byte file address in 4x4 format.

FIND/FILE
The general term for the DASD I/O services that ALCS provides.

fixed file
An ALCS record class - one of the classes that reside on the real-time database. All fixed-file records are also allocatable pool records (they have a special status of "in use for fixed file").

Within this class there are two record types reserved for use by ALCS itself (#KPTRI and #CPRCR). There can also be installation-defined fixed-file record types.

Each fixed-file record type is analogous to a relative file. Applications access fixed-file records by specifying the fixed-file record type and the record ordinal number. Note however that fixed-file records are not physically organized as relative files (logically adjacent records are not necessarily physically adjacent).

See real-time database, record class, and record type. See also system fixed file. Contrast with pool file.

fixed-file record
One of the two major types of record in the real-time database (the other is a pool-file record). When the number of records of a particular kind will not vary, the system programmer can define a fixed file record type for these records. ALCS application programs accessing fixed-file records use the ENTRC monitor-request macro to invoke the 4-byte file address compute routine (FACE or FACS) or use the FAC8C monitor-request macro to compute an 8-byte file address. The equivalent C functions are `face` or `facs` or `tpf_fac8c`.

fixed-file record type
(Known in TPF as FACE ID.) The symbol, by convention starting with a hash sign (#) [6] which identifies a particular group of fixed-file records. It is called the fixed-file record type symbol. The equated value of this symbol (called the fixed-file record type value) also identifies the fixed-file record type.

forward chain
The third fullword of a record stored on the ALCS database (part of the record header). When standard forward chaining is used, this field contains the file address of the next record in the chain, except that the last (or only) record contains binary zeros.

full-duplex
Deprecated term for duplex.

[6] This character might appear differently on your equipment. It is the character represented by hexadecimal 7B.

functional message
See ALCS command.

G

general data set (GDS)
The same as a general file, but accessed by different macros or C functions in ALCS programs.

general file
(1) A DASD data set (VSAM cluster) that is used to communicate data between offline utility programs and the online system. General files are not part of the real-time database.

(2) The ALCS record class that includes all records on the general files and general data sets. Each general file and general data set is a separate record type within this class. See record class and record type.

general file record
A record on a general file.

generalized trace facility (GTF)
An MVS trace facility. See also ALCS trace facility.

general sequential file
A class of sequential data set that is for input or output. ALCS application programs must have exclusive access to a general sequential file before they can read or write to it. See also real-time sequential file.

general tape
TPF term for a general sequential file.

general-use programming interface (GUPI)
An interface intended for general use in customer-written applications.

get file storage (GFS)
The general term for the pool file dispense mechanisms that ALCS provides.

global area
See application global area.

global resource serialization
The process of controlling access of entries to a global resource so as to protect the integrity of the resource.

H

half-duplex
A communication link that allows transmission in one direction at a time. Contrast with duplex.

halt
(1) The ALCS state when it is terminated.

(2) The action of terminating ALCS.

heap
An area of storage that a compiler uses to satisfy requests for storage from a high-level language (for example, `calloc` or `malloc` C functions). ALCS provides separate heaps for each entry (if needed). The heap is part of entry storage. Assembler language programs may also obtain or release heap storage using the CALOC, MALOC, RALOC, and FREEC monitor-request macros.

High Level Assembler (HLASM)
A functional replacement for Assembler H Version 2. HLASM contains new facilities for improving programmer productivity and simplifying assembler language program development and maintenance.

high-level language (HLL)
A programming language such as C or COBOL.

high-level language (HLL) storage unit
Alternative name for a type 2 storage unit. See storage unit.

high-level network (HLN)
A network that provides transmission services between transaction processing systems (for example, ALCS) and terminals. Strictly, the term "high-level network" applies to a network that connects to transaction processing systems using SLC. But in ALCS publications, this term is also used for a network that connects by using AX.25 or MATIP.

high-level network designator (HLD)
The entry or exit point of a block in a high-level network. For SLC networks, it is the SLC address of a switching center that is part of a high-level network. It comprises two bytes in the 7-bit transmission code used by SLC.

HLN entry address (HEN)
The high-level designator of the switching center where a block enters a high-level network.

HLN exit address (HEX)
The high-level designator of the switching center where a block leaves a high-level network.

hold
A facility that allows multiple entries to share data, and to serialize access to the data. The data can be a database record, or any named data resource. This facility can be used to serialize conflicting processes. See also record hold and resource hold.

host variable
See DB2 host variable

HTTP enabler
Part of the z/OS Client Web Enablement Toolkit which provides RESTful services enabling a z/OS application HTTP client to access Web services.

I

information block
See SLC link data block.

initial storage allocation (ISA)
An area of storage acquired at initial entry to a high-level language program. ALCS provides a separate ISA for each entry (if required). The ISA is part of entry storage.

initiation queue
In message queuing, a local queue on which the queue manager puts trigger messages. You can define an initiation queue to ALCS, in order to start an ALCS application automatically when a trigger message is put on the queue. See trigger message.

input/output control block (IOCB)
A control block that represents an ALCS internal "task". For example, ALCS uses an IOCB to process a DASD I/O request.

input queue
In message queuing with ALCS, you can define a local queue to ALCS in order to start an ALCS application automatically when a message is put on that queue. ALCS expects messages on the input queue to be in PPMSG message format. See PPMSG.

installation-wide exit
The means specifically described in an IBM software product's documentation by which an IBM software product may be modified by a customer's system programmers to change or extend the functions of the IBM software product. Such modifications consist of exit routines written to replace an existing module of an IBM software product, or to add one or more modules or subroutines to an IBM software product for the purpose of modifying (including extending) the functions of the IBM software product. Contrast with user exit.

instruction step
One mode of operation of the ALCS trace facility. Instruction step is a conversational trace facility that stops the traced application program before the execution of each processor instruction.

Integrated Cryptographic Service Facility (ICSF)
A facility on z/OS that provides data encryption and decryption services.

Interactive System Productivity Facility (ISPF)
An IBM licensed program that serves as a full-screen editor and dialog manager. ISPF provides a means of generating standard screen panels and interactive dialog between the application programmer and terminal user.

interchange address (IA)
In ALC, the 1-byte address of a terminal interchange. Different terminal interchanges connected to the same ALC link have different interchange addresses. Different terminal interchanges connected to different ALC links can have the same interchange address. See also terminal interchange

International Programmed Airlines Reservation System (IPARS)
A set of applications for airline use. The principal functions are reservations and message switching.

IPARS for ALCS
The ALCS shipment includes IPARS as a sample application, and installation verification aid for ALCS.

J

JSON Parser
Part of the z/OS Client Web Enablement Toolkit which provides a generic, native z/OS JavaScript Object Notation (JSON) parser for z/OS applications

K

KCN
Abbreviation for an SLC channel number. See SLC channel.

keypointable
See application global area.

keypoint B (CTKB)
A record that contains dynamic system information that ALCS writes to DASD when it is updated so that ALCS can restart from its latest status.

L

Language Environment
A common run-time environment and common run-time services for z/OS high level language compilers.

level
See ECB level.

line number (LN)
(1) In ALC, the 1-byte address of an ALC link. Different links connected to the same communication controller have different line numbers. Different links connected to different communication controllers can have the same line number.

(2) Synonym for symbolic line number.

Link Control -- Airline (LICRA)
The name of a programming request for price quotation (PRPQ) to the IBM 3705 Emulation Program (EP/VS). This modifies EP/VS to support SLC networks.

link control block (LCB)
See SLC link control block.

link data block (LDB)
See SLC link data block.

link trace
See SLC link trace.

local DXCREI index (LDI)
The first byte of a communication resource indicator (CRI).

local queue
In message queuing, a queue that belongs to the local queue manager. A local queue can contain a list of messages waiting to be processed. Contrast with remote queue.

lock
A serialization mechanism whereby a resource is restricted for use by the holder of the lock. See also hold.

log
See ALCS update log.

logging
The process of writing copies of altered database records to a sequential file. This is the method used to provide an up-to-date copy of the database should the system fail and the database have to be restored. The database records are logged to the ALCS update log file.

logical end-point identifier (LEID)
In NEF2 and ALCI environments, a 3-byte identifier assigned to an ALC terminal.

logical unit type 6.2 (LU 6.2)
The SNA logical unit type that supports general communication between programs in a distributed processing environment; the SNA logical unit type on which Common Programming Interface - Communications (CPI-C) is built.

log in
TPF term for establishing routing between a terminal and an application.

log on
Establish a session between an SNA terminal and an application such as ALCS. See also routing.

logon mode
In VTAM, a set of predefined session parameters that can be sent in a BIND request. When a set is defined, a logon mode name is associated with the set.

logon mode table
In VTAM, a table containing several predefined session parameter sets, each with its own logon mode name.

long message transmitter (LMT)
A part of the IPARS application that is responsible for blocking and queuing printer messages for output. Also called XLMT.

long-term pool
An ALCS record class - one of the classes that reside on the real-time database. Within this class, there is one record type for each DASD record size. All long-term pool-file records are also allocatable pool records. ALCS application programs can use long-term pool records for long-lived or high-integrity data. See pool file, real-time database, record class, and record type.

L0, L1, L2, L3, ..., L8
Assembler symbols (and defined values in C) for the storage block sizes and record sizes that ALCS supports. See DASD record and storage block size.

M

macro trace block
See entry macro trace block and system macro trace block.

Mapping of Airline Traffic over IP (MATIP)
A protocol for transporting traditional airline messages over an IP (Internet Protocol) network. Internet RFC (Request for Comments) number 2351 describes the MATIP protocol.

MBI exhaustion
The condition of an SLC link when a sender cannot transmit another message because all 7 SLC message labels are already "in use"; that is, the sender must wait for acknowledgement of a message

so that it can reuse the corresponding message label. See also SLC link, SLC message label, and SLC message block indicator.

message
For terminals with an Enter key, an input message is the data that is sent to the host when the Enter key is hit. A response message is the data that is returned to the terminal. WTTY messages have special "start/end of message" character sequences. One or more input and output message pairs make up a transaction.

message block indicator
See SLC message block indicator.

message label
See SLC message label.

Message Queue Interface (MQI)
The programming interface provided by the IBM WebSphere MQ message queue managers. This programming interface allows application programs to access message queuing services.

message queue manager
See queue manager.

message queuing
A programming technique in which each program within an application communicates with the other programs by putting messages on queues. This enables asynchronous communication between processes that may not be simultaneously active, or for which no data link is active. The message queuing service can assure subsequent delivery to the target application.

message switching
An application that routes messages by receiving, storing, and forwarding complete messages. IPARS for ALCS includes a message switching application for messages that conform to ATA/IATA industry standards for interline communication *ATA/IATA Interline Communications Manual,* DOC.GEN/1840.

mirrored pair
Two units that contain the same data and are referred to by the system as one entity.

monitor-request macro
Assembler language macro provided with ALCS, corresponding to TPF "SVC-type" or "control program" macros. Application programs use these macros to request services from the online monitor.

MQ Bridge
The ALCS MQ Bridge allows application programs to send and receive messages using WebSphere MQ for z/OS queues, without the need to code MQ calls in those programs. The MQ Bridge installation-wide monitor exits USRMQB0, USRMQB1, USRMQB2, and USRMQB3 allow you to customize the behaviour of the MQ Bridge to suit your applications.

MQSeries
A previous name for WebSphere MQ.

multibyte character
A mixture of single-byte characters from a single-byte character set and double-byte characters from a double-byte character set.

multiblock message
In SLC, a message that is transmitted in more than one link data block. See link data block.

Multiple Virtual Storage/Data Facility Product (MVS/DFP)
An MVS licensed program that isolates applications from storage devices, storage management, and storage device hierarchy management.

Multisystem Networking Facility (MSNF)
An optional feature of VTAM that permits these access methods, together with NCP, to control a multiple-domain network.

N

namelist
In message queuing, a namelist is an object that contains a list of other objects.

native file address
For migration purposes ALCS allows two or more file addresses to refer to the same database or general file record. The file address that ALCS uses internally is called the native file address.

NCP Packet Switching Interface (NPSI)
An IBM licensed program that allows communication with X.25 lines.

NetView
A family of IBM licensed programs for the control of communication networks.

NetView operator identifier (NetView operator ID)
A 1- to 8-character name that identifies a NetView operator.

NetView program
An IBM licensed program used to monitor a network, manage it, and diagnose network problems.

NetView resource
A NetView operator ID which identifies one of the following:

- A NetView operator logged on to a terminal.
- A NetView operator ID automation task. One of these tasks is used by ALCS to route RO CRAS messages to the NetView Status Monitor Log (STATMON).

network control block (NCB)
A special type of message, used for communication between a transaction processing system and a high-level network (HLN). For example, an HLN can use an NCB to transmit information about the network to a transaction processing system.

For a network that connects using SLC, an NCB is an SLC link data block (LDB). Indicators in the LDB differentiate NCBs from other messages.

For a network that connects using AX.25, NCBs are transmitted across a dedicated permanent virtual circuit (PVC).

Network Control Program (NCP)
An IBM licensed program resident in an IBM 37xx Communication Controller that controls attached lines and terminals, performs error recovery, and routes data through the network.

Network Control Program Packet Switching Interface (NPSI)
An IBM licensed program that provides a bridge between X.25 and SNA.

Network Control Program/Virtual Storage (NCP/VS)
An IBM licensed program. ALCS V2 uses the EP/VS component of NCP/VS to access SLC networks.

Network Extension Facility (NEF)
The name of a programming request for price quotation (PRPQ P09021) that allows management of ALC networks by NCP; now largely superseded by ALCI.

Network Terminal Option (NTO)
An IBM licensed program that converts start-stop terminal device communication protocols and commands into SNA and VTAM communication protocols and commands. ALCS uses NTO to support World Trade Teletypewriter (WTTY).

O

object
In message queuing, objects define the attributes of queue managers, queues, process definitions, and namelists.

offline
A function or process that runs independently of the ALCS online monitor. For example, the ALCS diagnostic file processor is an offline function. See also ALCS offline program.

online
A function or process that is part of the ALCS online monitor, or runs under its control. For example, all ALCS commands are online functions. See also ALCS online monitor.

open
Allocate a sequential file data set to ALCS and open it (MVS OPEN macro). For general sequential files this is a function of the TOPNC monitor-request macro (or equivalent C function). ALCS automatically opens other sequential files during restart.

optimized local adapters (OLA) for WebSphere Application Server for z/OS (WAS)
Built-in, high-speed, bi-directional adapters for calls between WebSphere Application Server for z/OS and ALCS in another address space on the same z/OS image. OLA allows ALCS customers to support an efficient integration of newer Java-based applications with ALCS-based applications. A set of callable services can be used by ALCS assembler or C/C++ programs for exchanging data with applications running in WebSphere Application Server for z/OS. For more information on the callable services (with names of the form BBOA1*xxx*) see the IBM Information Center for WebSphere Application Server - Network Deployment (z/OS) and search for BBOA1. You can use the USRWAS1 installation-wide monitor to verify the caller's authority and to identify input and output messages.

operator command
See ALCS command. Can also refer to non-ALCS commands, for example, MVS or VTAM commands.

ordinal
See communication resource ordinal and record ordinal.

P

package
See DB2 package

package list
See DB2 package list

padded ALC
A transmission code that adds one or more bits to the 6-bit airline line control (ALC) transmission code so that each ALC character occupies one character position in a protocol that uses 7- or 8-bit transmission codes. See also airlines line control.

padded SABRE
Synonym for padded ALC.

passenger name record (PNR)
A type of record commonly used in reservation systems. It contains all the recorded information about an individual passenger.

path
The set of components providing a connection between a processor complex and an I/O device. For example, the path for an IBM 3390 DASD volume might include the channel, ESCON Director, 3990 Storage Path, 3390 Device Adapter, and 3390 internal connection. The specific components used in a particular path are dynamic and may change from one I/O request to the next. See balanced path.

pathlength
The number of machine instructions needed to process a message from the time it is received until the reponse is sent to the communication facilities.

performance monitor
An online function that collects performance data and stores it in records on the ALCS real-time database. It can produce online performance reports based on current data and historical data.

pilot
See data file.

pool directory update (PDU)
A facility of TPF that recovers long-term pool file addresses without running Recoup . PDU identifies and makes available all long-term pool-file records that have been released.

pool file
Short-term pool, long-term pool, and allocatable pool. Within each pool file class, there is one record type for each record size; for example, short-term pool includes the record type L1STPOOL (size L1 short-term pool records).

Each pool-file record type contains some records that are in-use and some that are available. There is a dispense function that selects an available record, changes its status to in-use, and returns the file address. Also, there is a release function that takes the file address of an in-use pool-file record and changes the record status to available.

To use a pool-file record, a program must:

1. Request the dispense function. This returns the file address of a record. Note that the record contents are, at this stage, unpredictable.
2. Write the initial record contents, using the file address returned by step “1” on page 433.
3. Save the file address returned by step “1” on page 433.
4. Read and write the record to access and update the information as required. These reads and writes use the file address saved in step “3” on page 433.

   When the information in the record is no longer required, a program must:
5. Delete (clear to zeros) the saved copy of the file address (see step “3” on page 433).
6. Request the release function.

See also record class. Contrast with fixed file.

pool file directory record (PFDR)
The ALCS pool file management routine keeps a directory for each size (L1, L2, ...L8) of short-term pool file records and long-term pool-file records. It keeps these directories in pool file directory records.

pool-file record
ALCS application programs access pool-file records with file addresses similar to those for fixed-file records. To obtain a pool-file record, an application program uses a monitor-request macro (or equivalent C function) that specifies a 2-byte record ID or a pool-file record type.

When the data in a pool-file record is no longer required, the application uses a monitor-request macro (or equivalent C function) to release the record for reuse. See pool file.

pool-file record identifier (record ID)
The record ID of a pool-file record. On get file requests (using the GETFC monitor-request macro or equivalent C function) the program specifies the pool-file record ID. This identifies whether the pool-file record is a short-term or long-term pool-file record and also determines the record size (L1, L2, ...L8). (Coding the 2-byte record IDs, and the corresponding pool-file record sizes and types, is part of the ALCS generation procedure.) See also record ID qualifier.

pool-file record type
Each collection of short-term and long-term pool-file records of a particular record size (identified by the symbols L1, L2, ..., L8) is a different record type. Each pool-file record type has a different name. For short-term pool-file records, this is L*n*STPOOL, where L*n* is the record size symbol. For long-term pool-file records the name is L*n*LTPOOL.

post processor
See ALCS diagnostic file processor.

PPMSG
ALCS program-to-program message format, used by the ALCS message router to send and receive messages on a message routing path to another system. In PPMSG message format, the routing control parameter list (RCPL) precedes the message text.

primary action code
The first character of any input message. The primary action code Z is reserved for ALCS commands. See secondary action code.

Prime CRAS
The primary display terminal, or NetView ID, that controls the ALCS system. See also computer room agent set (CRAS).

process definition object
In message queuing, an object that contains the definition of a message queuing application. For example, a queue manager uses the definition when it works with trigger messages.

product sensitive programming interface (PSPI)
An interface intended for use in customer-written programs for specialized purpose only, such as diagnosing, modifying, monitoring, repairing, tailoring or tuning of ALCS. Programs using this interface may need to be changed in order to run with new product releases or versions, or as a result of service.

program linkage
Mechanism for passing control between separate portions of the application program. See dynamic program linkage and static program linkage.

program nesting level
One of 32 ECB areas used by the ENTER/BACK mechanism for saving return control data.

program-to-program interface
In NetView, a facility that allows user programs to send data to, or receive data from, other user programs. It also allows system and application programs to send alerts to the NetView hardware monitor.

P.1024
A SITA implementation of SLC. See SLC.

P.1124
A SITA implementation of SLC. See SLC.

P.1024A
The SITA implementation of airline line control (ALC).

Q

queue manager
A system program that provides queuing services to applications. It provides an application programming interface so that programs can access messages on the queues that the queue manager owns. WebSphere MQ for z/OS is an example of a queue manager.

R

real-time database
The database to which ALCS must have permanent read and write access. As an ALCS generation option, the real-time database can be duplicated in order to minimize the effects of a DASD failure.

real-time sequential file
A sequential data set used only for output. ALCS application programs can write to any real-time sequential file without requiring exclusive access to the data set. See also general sequential file.

real-time tape
TPF term for a real-time sequential file.

receive only (RO)
The function of a communication terminal that can receive but not send data. An example is a printer that does not have a keyboard.

receive only CRAS
A printer terminal (or NetView operator ID) that ALCS uses to direct status messages. Commonly known as RO CRAS.

record
A set of data treated as a unit.

record class
The first (highest) level categorization of ALCS DASD records. ALCS defines the following record classes:

Allocatable pool
Application fixed file
Configuration data set
General file
Long-term pool
Short-term pool
System fixed file.

See also record type and record ordinal.

record code check (RCC)
The third byte of any record stored in the ALCS database. It is part of the record header.

The RCC field is intended to help detect the incorrect chaining of records which have the same record ID. This is particularly useful for passenger name records (PNRs), of which there are often hundreds of thousands. A mismatch in RCC values shows that the chain is broken, probably as a result of an application program releasing a record too soon. (A false match cannot be excluded, but the RCC should give early warning of a chaining problem.)

record header
A standard format for the first 16 bytes of a record stored on the ALCS database. It contains the following fields:

Record ID
Record code check
Control byte
Application program name
Forward chain
Backward chain.

Not all records contain forward chains and backward chains. Some applications extend the record header by including extra fields. TPFDF uses an extended record header.

record hold
A type of hold that applies to DASD records. Applications that update records can use record hold to prevent simultaneous updates. See also resource hold.

record identifier (record ID)
The first two bytes of a record stored on the ALCS database, part of the record header.

The record ID should always be used to indicate the nature of the data in the record. For example, airlines reservations applications conventionally store passenger name records (PNRs) as long-term pool-file records with a record ID of 'PR'.

When application programs read such records, they can (optionally) request ALCS to check that the record ID matches that which the application program expects.

When application programs request ALCS to dispense pool file records, ALCS uses the record ID to select an appropriate long-term or short-term pool-file record of the requested record size (L1, L2,...,L8). See also record ID qualifier.

record ID qualifier
A number 0 through 9 that differentiates between record types that have the same record ID.

For compatibility with previous implementations of the record ID qualifier, ALCS also accepts the character qualifiers P and O. P (primary) is equivalent to 0, and O (overflow) is equivalent to 1.

record ordinal
The relative record number within a record type. See record class and record type.

record size
See DASD record.

record type
The second level categorization of ALCS DASD records. Within any one record class, the records are categorized into one or more record types. See also record type number, record type symbol, record class and record ordinal.

record type number
A number that identifies a record type.

record type symbol
The character string that identifies a fixed-file record type (#*xxxxx*), a long-term pool-file record type (LsLTPOOL), a short-term pool-file record type (LsSTPOOL), or a general file (GF-*nnn*). The value of the record type symbol is the record type number.

Recoup
A real-time database validation routine which runs online in the ALCS system. (Note that, while the Recoup routines of TPF consist of a number of phases, some online and some offline, the ALCS Recoup is a single online phase that runs, without operator intervention, in any system state.)

Recoup reads selected fixed-file records in the database, and then follows up all chains of pool-file records in the database, noting that these records are in use and giving a warning of any that have been corrupted or released. It then updates the pool file directory records (PFDRs) to show the status of all records.

The ALCS pool file dispense procedure identifies records not in a chain (and so apparently available for reuse) that have not been released.

recoup descriptors
These describe the structure of the entire real-time database.

reentrant
The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by two or more tasks. All ALCS application programs must be reentrant.

relational database
A database that is in accordance with the relational model of data. The database is perceived as a set of tables, relationships are represented by values in tables, and data is retrieved by specifying a result table that can be derived from one or more base tables.

release (a pool-file record)
To make available a long-term or short-term pool-file record so that it can be subsequently dispensed. An application program requests the release action. See dispense a pool-file record.

release file storage (RFS)
The general term for the pool-file release mechanisms that ALCS provides.

remote queue
In message queuing, a queue that belongs to a remote queue manager. Programs can put messages on remote queues, but they cannot get messages from remote queues. Contrast with local queue.

remote terminal trace
One mode of operation of the ALCS trace facility. Remote terminal trace is a conversational trace facility to interactively trace entries from a terminal other than your own.

REpresentational State Transfer (REST)
An architecture which defines how data is represented to a client in a format (using pis) that is convenient for that client to access Web services.
Common message protocols used for this purpose are HTTP, JSON and XML. Applications using the REST pis are said to be RESTful applications.

reservations
An online application which is used to keep track of seat inventories, flight schedules, and other related information. The reservation system is designed to maintain up-to-date data and to respond within seconds or less to inquiries from ticket agents at locations remote from the computing system.

IPARS for ALCS includes a sample reservations application for airlines.

reserve
Unassign a general sequential file from an entry but leave the file open, so that another (or the same) entry can assign it. Application programs can use the TRSVC monitor-request macro (or equivalent C function) to perform this action.

resource
Any facility of a computing system or operating system required by a job or task, and including main storage, input/output devices, processing unit, data sets, and control or processing programs. See also communication resource.

resource entry index (REI)
The second and third bytes of a communication resource identifier (CRI).

resource hold
A type of hold that can apply to any type of resource. Applications can define resources according to their requirements, and identify them to ALCS using a unique name. See also record hold.

RO CRAS
See receive only CRAS.

rollback
An operation that reverses all the changes made during the current unit of recovery. After the operation is complete, a new unit of recovery begins.

routing
The connection between a communication resource connected to ALCS (typically a terminal on an SNA or non-SNA network) and an application (running under ALCS or another system). Also sometimes called "logging in", but this must be distinguished from logging on, which establishes the SNA connection (session) between the terminal and ALCS.

routing control parameter list (RCPL)
A set of information about the origin, destination, and characteristics of a message. With each input message, ALCS provides an RCPL in the ECB. An output message that is sent using the ROUTC (routc) service also has an RCPL associated with it.

S

scroll
To move a display image vertically or horizontally to view data that otherwise cannot be observed within the boundaries of the display screen.

secondary action code
The second character of an ALCS command. (ALCS commands are made up of 5 characters: Z followed by a secondary action code.) See primary action code.

sequential file
A file in which records are processed in the order in which they are entered and stored in the file. See general sequential file and real-time sequential file.

serialization
A service that prevents parallel or interleaved execution of two or more processes by forcing the processes to execute serially.

For example, two programs can read the same data item, apply different updates, and then write the data item. Serialization ensures that the first program to start the process (read the item) completes the process (writes the updated item) before the second program can start the process - the second program applies its update to the data item which already contains the first update. Without serialization, both programs can start the process (read the item) before either completes the process (writes the updated item) - the second write destroys the first update. See also assign, lock, and hold.

Serviceability Level Indicator Processing (SLIP)
An MVS operator command which acts as a problem determination aid.

short-term pool
An ALCS record class - one of the classes that resides on the real-time database. Within this class, there is one record type for each DASD record size. All short-term pool-file records are also allocatable pool records (they have a special status of "in use for short-term pool"). ALCS application programs can use short-term pool records for short-lived low-integrity data. See pool file, real-time database, record class, and record type.

simplex transmission
Data transmission in one direction only. See also duplex and half-duplex.

sine in/out
Those applications that provide different functions to different end users of the same application can require the user to sine in [7] to the specific functions they require. The sine-in message can, for example, include an authorization code.

single-block message
In SLC, a message that is transmitted in one link data block. See link data block.

single-phase commit
A method in which a program can commit updates to a message queue or relational database without coordinating those updates with updates the program has made to resources controlled by another resource manager. Contrast with two-phase commit.

SLC
See synchronous link control.

SLC channel
A duplex telecommunication line using ATA/IATA SLC protocol. There can be from 1 to 7 channels on an SLC link.

SLC error index byte (EIB)
A 1-byte field generated by Line Control - Airline (LICRA) and transferred to ALCS with each incoming link control block and link data block. Certain errors cause LICRA to set on certain bits of the EIB. See also Link Control -- Airline (LICRA).

SLC information block
Synonym for SLC link data block.

SLC link
A processor-to-processor or processor-to-HLN connection. ALCS supports up to 255 SLC links in an SLC network.

An SLC link that is in the process of an open, close, start, or stop function is said to be "cycling".

SLC link control block (LCB)
A 4-byte data item transmitted across an SLC link to control communications over the link. LCBs are used, for example, to confirm that a link data block (LDB) has arrived, to request retransmission of an LDB, and so on.

SLC link data block (LDB)
A data item, transmitted across an SLC link, that contains a message or part of a message. One LDB can contain a maximum of 240 message characters, messages longer than this must be split and transmitted in multiple LDBs. Synonymous with SLC information block.

SLC link trace
A function that provides a record of SLC communication activity. It can either display the information in real time or write it to a diagnostic file for offline processing, or both. Its purpose is like that of an NCP line trace, but for the SLC protocol.

SLC message block indicator (MBI)
A 1-byte field in the SLC link data block that contains the SLC message label and the block number. A multiblock message is transmitted in a sequence of up to 16 link data blocks with block numbers 1, 2, 3, ... 16. See also multiblock message, SLC link data block, and SLC message label.

[7] This spelling is established in the airline industry.

SLC message label
A number in the range 0 through 7, excluding 1. In P.1024, consecutive multiblock messages are assigned SLC message labels in the sequence: 0, 2, 3, ... 6, 7, 0, 2, and so on. In P.1124, single-block messages are (optionally) also included in the sequence. See also P.1024, P.1124 and SLC message block indicator.

SLC transmission status indicator (TSI)
A 1-byte field in the SLC link data block that contains the SLC transmission sequence number. See also SLC transmission sequence number.

SLC transmission sequence number (TSN)
A number in the range 1 through 31. Consecutive SLC link data blocks transmitted in one direction on one SLC channel are assigned TSNs in the sequence: 1, 2, 3, ... 30, 31, 1, 2, and so on. See also SLC link data block, SLC channel, and SLC transmission status indicator.

SLC Type A traffic
See Type A traffic.

SLC Type B traffic
See Type B traffic.

Société Internationale de Télécommunications Aéronautiques (SITA)
An international organization which provides communication facilities for use within the airline industry.

SQL Communication Area (SQLCA)
A structure used to provide an application program with information about the execution of its SQL statements.

SQL Descriptor Area (SQLDA)
A structure that describes input variables, output variables, or the columns of a result table used in the execution of manipulative SQL statements.

stack
An area of storage that a compiler uses to allocate variables defined in a high-level language. ALCS provides separate stacks for each entry (if needed). The stack is part of entry storage.

standard message format
For input and output messages, a message format which includes a 2-byte field for the message length.

standby
The state of ALCS after it has been initialized but before it has been started. Standby is not considered one of the system states.

static program linkage
Program linkage where the connection between the calling and called program is established before the execution of the program. The connection is established by the assembler, compiler, prelinker, or linkage editor. Static program linkage does not invoke ALCS monitor services. See also dynamic program linkage.

static SQL
See embedded SQL.

statistical report generator (SRG)
An offline ALCS utility that is a performance monitoring tool. It takes the data written to the ALCS data collection or diagnostic file processor by the data collection function and produces a variety of reports and bar charts. The SRG is the equivalent of TPF "data reduction".

STATMON
See NetView resource.

storage block
An area of storage that ALCS allocates to an entry. It is part of entry storage. See storage block sizes.

storage block size
ALCS allows storage blocks of up to 9 different sizes. These are identified in programs by the assembler symbols (or defined C values) L0, L1, L2, ..., L8. Installations need not define all these block sizes but usually define at least the following:

Size L0 contains 127 bytes of user data
Size L1 contains 381 bytes of user data
Size L2 contains 1055 bytes of user data
Size L3 contains 4000 bytes of user data
Size L4 contains 4095 bytes of user data.

The system programmer can alter the size in bytes of L1 through L4, and can specify the remaining block sizes.

storage level
An area in the ECB or a DECB used to hold the address and size of a storage block. See ECB level and DECB level.

storage unit
The ALCS storage manager allocates storage in units called storage units. Entry storage is suballocated within storage units; for example, one storage unit can contain an ECB and several storage blocks attached to that ECB.

ALCS uses three types of storage units:

- Prime and overflow storage units for entry storage and heap storage (if an entry storage block can be used). Also called type 1 storage units.
- High-level language storage units for stack storage Also called type 2 storage units.
- Storage units for heap storage (if an entry storage block can not be used) for programs. Also called type 3 storage units.

The size of a storage unit, and the number of each type of storage unit, is defined in the ALCS generation. See entry storage.

store access
Access which only involves writing (not reading). Compare with fetch access.

striping
A file organization in which logically adjacent records are stored on different physical devices. This organization helps to spread accesses across a set of physical devices.

Structured Query Language (SQL)
a standardized language for defining and manipulating data in a relational database.

symbolic line number (SLN)
In TPF, a 1-byte address of an ALC link, derived from the line number but adjusted so that all ALC links connected to the TPF system have a different symbolic line number. See also line number.

Synchronous Data Link Control (SDLC)
A discipline conforming to subsets of the Advanced Data Communication Control Procedures (ADCCP) of the American National Standards Institute (ANSI) and High-level Data Link Control (HDLC) of the International Organization for Standardization, for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection.

Transmission exchanges can be duplex or half-duplex over switched or nonswitched links. The configuration of the link connection can be point-to-point, multipoint, or loop.

Synchronous Link Control (SLC)
A discipline conforming to the ATA/IATA Synchronous Link Control, as described in the ATA/IATA publication *ATA/IATA Interline Communications Manual*, ATA/IATA document DOC.GEN 1840.

syncpoint
An intermediate or end point during processing of a transaction at which the transaction's protected resources are consistent. At a syncpoint, changes to the resources can safely be committed, or they can be backed out to the previous syncpoint.

system error
Error that the ALCS monitor detects. Typically, ALCS takes a dump, called a system error dump, to the ALCS diagnostic file. See also ALCS diagnostic file and ALCS diagnostic file processor. See also system error dump, system error message.

system error dump
(1) A storage dump that ALCS writes to the ALCS diagnostic file when a system error occurs. See also ALCS diagnostic file and system error.

(2) The formatted listing of a storage dump produced by the ALCS diagnostic file processor. See also ALCS diagnostic file processor.

system error message
A message that ALCS sends to receive only CRAS when a system error occurs. See also receive only CRAS and system error.

system error option
A parameter that controls what action ALCS takes when it detects a system error. See also system error.

system fixed file
An ALCS record class - one of the classes that reside on the real-time database. All system fixed-file records are also allocatable pool records (they have a special status of "in use for system fixed file").

System fixed-file records are reserved for use by ALCS iteslf. See real-time database, record class, and record type.

system macro trace block
There is one system macro trace block. Each time an entry issues a monitor-request macro (or equivalent C function), ALCS records information in the system macro trace block.

This information includes the ECB address, the macro request code, the name of the program that issued the macro, and the displacement in the program. The ALCS diagnostic file processor formats and prints the system macro trace block in ALCS system error dumps. See also entry macro trace block.

System Modification Program/Extended (SMP/E)
An IBM licensed program used to install software and software changes on MVS systems. In addition to providing the services of SMP, SMP/E consolidates installation data, allows flexibility in selecting changes to be installed, provides a dialog interface, and supports dynamic allocation of data sets.

Systems Application Architecture (SAA)
A set of software interfaces, conventions, and protocols that provide a framework for designing and developing applications with cross-system consistency.

Systems Network Architecture (SNA)
The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of networks.

system sequential file
A class of sequential data sets used by ALCS itself. Includes the ALCS diagnostic file, the ALCS data collection file, and the ALCS update log file or files.

system state
The ALCS system can run in any of the following system states: IDLE, CRAS, message switching (MESW), and normal (NORM).

Each state represents a different level of availability of application functions. Altering the system state is called "cycling the system". See also standby.

system test compiler (STC)
An offline ALCS utility that compiles data onto data files for loading on to the real-time database. STC also builds test unit tapes (TUTs) for use by the system test vehicle (STV).

system test vehicle (STV)
An online ALCS function that reads input messages from a general sequential file test unit tape (TUT) and simulates terminal input. STV intercepts responses to simulated terminals and writes them to the ALCS diagnostic file.

T

terminal
A device capable of sending or receiving information, or both. In ALCS this can be a display terminal, a printer terminal, or a NetView operator identifier.

terminal address (TA)
In ALC, the 1-byte address of an ALC terminal. Different terminals connected to the same terminal interchange have different terminal addresses. Different terminals connected to different terminal interchanges can have the same terminal address. See also terminal interchange.

terminal circuit identity (TCID)
Synonym for line number.

terminal hold
When an ALCS application receives an input message, it can set terminal hold on for the input terminal. Terminal hold remains on until the application sets it off. The application can reject input from a terminal that has terminal hold set on. Also referred to as AAA hold.

terminal interchange (TI)
In ALC, synonym for terminal control unit.

terminate
(1) To stop the operation of a system or device.

(2) To stop execution of a program.

test unit tape (TUT)
A general sequential file that contains messages for input to the system test vehicle (STV). TUTs are created by the system test compiler (STC).

time available supervisor (TAS)
An ALCS or TPF function that creates and dispatches low priority entries.

time-initiated function
A function initiated after a specific time interval, or at a specific time. In ALCS this is accomplished by using the CRETC monitor-request macro or equivalent C function. See create service.

TP profile
The information required to establish the environment for, and attach, an APPC/MVS transaction program on MVS, in response to an inbound allocate request for the transaction program.

trace facility
See ALCS trace facility, generalized trace facility, and SLC link trace.

transaction
The entirety of a basic activity in an application. A simple transaction can require a single input and output message pair. A more complex transaction (such as making a passenger reservation) requires a series of input and output messages.

Transaction Processing Facility (TPF)
An IBM licensed program with many similarities to ALCS. It runs native on IBM System/370 machines, without any intervening software (such as MVS). TPF supports only applications that conform to the TPF interface. In this book, TPF means Airline Control Program (ACP), as well as all versions of TPF.

Transaction Processing Facility Database Facility (TPFDF)
An IBM licensed program that provides database management facilities for programs that run in an ALCS or TPF environment.

Transaction Processing Facility/Advanced Program to Program Communications (TPF/APPC)
This enables LU 6.2 for TPF.

Transaction Processing Facility/Data Base Reorganization (TPF/DBR)
A program which reorganizes the TPF real-time database.

Transaction Processing Facility/MVS (TPF/MVS)
Alternative name for ALCS V2 .

Transaction program identifier (TP_ID)
A unique 8-character token that APPC/MVS assigns to each instance of a transaction program. When multiple instances of a transaction program are running sumultaneously, they have the same transaction program name, but each has a unique TP_ID.

transaction scheduler name
The name of an APPC/MVS scheduler program. The ALCS transaction scheduler name is ALCSx000, where x is the ALCS system identifier as defined during ALCS generation.

transfer vector
An ALCS application program written in assembler, SabreTalk, or C, can have multiple entry points for dynamic program linkage. These entry points are called transfer vectors. Each transfer vector has a separate program name.

transmission status indicator
See SLC transmission status indicator.

transmission sequence number
See SLC transmission sequence number.

trigger event
In message queuing, an event (such as a message arriving on a queue) that causes a queue manager to create a trigger message on an initiation queue.

trigger message
In message queuing, a message that contains information about the program that a trigger monitor is to start.

trigger monitor
In message queuing, a continuously-running application that serves one or more initiation queues. When a trigger message arrives on an initiation queue, the trigger monitor retrieves the message. When ALCS acts as a trigger monitor, it uses the information in the trigger message to start an ALCS application that serves the queue on which a trigger event occurred.

triggering
In message queuing, a facility that allows a queue manager to start an application automatically when predetermined conditions are met.

TSI exhaustion
The condition of an SLC channel when a sender cannot transmit another SLC link data block (LDB) because the maximum number of unacknowledged LDBs has been reached. The sender must wait for acknowledgement of at least one LDB so that it can transmit further LDBs. See also SLC channel, SLC link data block, SLC transmission sequence number, and SLC transmission status indicator.

two-phase commit
A protocol for the coordination of changes to recoverable resources when more than one resource manager is used by a single transaction. Contrast with single-phase commit.

type
See record type.

Type A traffic
ATA/IATA conversational traffic - that is, high-priority low-integrity traffic transmitted across an SLC or AX.25 link.

Type B application-to-application program (BATAP)
In any system (such as ALCS) that communicates with SITA using AX.25 or MATIP, this is the program which receives and transmits type B messages.

Type B traffic
ATA/IATA conventional traffic - that is, high-integrity, low-priority traffic transmitted across an SLC or AX.25 link or a MATIP TCP/IP connection.

type 1 pool file dispense mechanism
The mechanism used in ALCS prior to V2 Release 1.3 (and still available in subsequent releases) to dispense both short-term and long-term pool-file records.

type 1 storage unit
Prime or overflow storage unit for entry storage and small heap storage. See storage unit.

type 2 pool file dispense mechanisms
The mechanisms available since ALCS V2 Release 1.3 to dispense pool-file records (the mechanisms are different for short-term and long-term pool-file records).

IBM recommends users to migrate to type 2 dispense mechanisms as part of their migration process.

type 2 storage unit
High-level language storage unit for stack storage. See storage unit.

type 3 storage unit
Storage unit for heap storage that is used when an entry storage block cannot satisfy a request. See storage unit.

U

unit of recovery
A recoverable sequence of operations within a single resource manager (such as WebSphere MQ for z/OS or DB2 for z/OS). Compare with unit of work.

unit of work
A recoverable sequence of operations performed by an application between two points of consistency. Compare with unit of recovery.

Universal Communications Test Facility (UCTF)
An application used by SITA for SLC protocol acceptance testing.

update log
See ALCS update log.

user data-collection area
An optional extension to the data-collection area in the ECB. Application programs can use the DCLAC macro to update or read the user data-collection area.

user exit
A point in an IBM-supplied program at which a user exit routine can be given control.

user exit routine
A user-written routine that receives control at predefined user exit points. User exit routines can be written in assembler or a high-level language.

V

version number
In ALCS and TPF, two characters (not necessarily numeric), optionally used to distinguish between different versions of a program. Sometimes also used with other application components such as macro definitions.

virtual file access (VFA)
An ALCS caching facility for reducing DASD I/O. Records are read into a buffer, and subsequent reads of the same record are satisfied from the buffer. Output records are written to the buffer, either to be written to DASD - immediately or at a later time - or to be discarded when they are no longer useful.

virtual SLC link
Used to address an X.25 PVC or TCP/IP resource for transmitting and receiving Type B traffic. Some applications (such as IPARS MESW) address communication resources using a symbolic line number (SLN) instead of a CRI. These applications can address X.25 PVC and TCP/IP resources by converting the unique SLN of a virtual SLC link to the CRI of its associated X.25 PVC or TCP/IP resource.

W

WAS Bridge
The ALCS WAS Bridge allows ALCS application programs to send and receive messages using optimized local adapters (OLA) for WebSphere Application Server for z/OS without the need to code those callable services in ALCS programs. The ALCS WAS Bridge installation-wide monitor exits USRWAS3, USRWAS4, USRWAS5, and USRWAS6 allow you to customize the behaviour of the WAS Bridge to suit your applications.

WebSphere MQ for z/OS
An IBM product that provides message queuing services to systems such as CICS, IMS, ALCS or TSO. Applications request queuing services through MQI.

wide character
A character whose range of values can represent distinct codes for all members of the largest extended character set specified among the supporting locales. For the z/OS XL C/C++ compiler, the character set is DBCS, and the value is 2 bytes.

workstation trace
One mode of operation of the ALCS trace facility. Workstation trace controls the remote debugger facility. The remote debugger is a source level debugger for C/C++ application programs.

World Trade Teletypewriter (WTTY)
Start-stop telegraph terminals that ALCS supports through Network Terminal Option (NTO).

Z

Z message
See ALCS command.

Bibliography

Note that unless otherwise specified, the publications listed are those for the z/OS platform.

Airline Control System Version 2 Release 4.1

- *Application Programming Guide*, SH19-6948
- *Application Programming Reference - Assembler Language*, SH19-6949
- *Application Programming Reference - C Language*, SH19-6950
- *Concepts and Facilities*, SH19-6953
- *General Information Manual*, GH19-6738
- *Installation and Customization*, SH19-6954
- *Licensed Program Specifications*, GC34-6327
- *Messages and Codes*, SH19-6742
- *Operation and Maintenance*, SH19-6955
- *Program Directory*, GI10-2577
- *ALCS World Wide Web Server User Guide*
- *OCTM User Guide*

MVS

- *Data Areas, Volumes 1 through 5*, GA22-7581 through GA22-7585
- *Diagnosis Reference*, GA22-7588
- *Diagnosis Tools and Service Aids*, GA22-7589
- *Initialization and Tuning Guide*, SA22-7591
- *Initialization and Tuning Reference*, SA22-7592
- *Installation Exits*, SA22-7593
- *IPCS Commands*, SA22-7594
- *IPCS User's Guide*, SA22-7596
- *JCL Reference*, SA22-7597
- *JCL User's Guide*, SA22-7598
- *JES2 Initialization and Tuning Guide*, SA22-7532
- *JES2 Initialization and Tuning Reference*, SA22-7533
- *Program Management: User's Guide and Reference*, SA22-7643
- *System Codes*, SA22-7626
- *System Commands*, SA22-7627
- *System Messages, Volumes 1 through 10*, SA22-7631 through SA22-7640

APPC/MVS

- *MVS Planning: APPC/MVS Management*, SA22-7599
- *MVS Programming: Writing Transaction Programs for APPC/MVS*, SA22-7621

DFSMS

- *Access Method Services for Catalogs*, SC26-7394

- *DFSMSdfp Storage Administration Reference*, SC26-7402
- *DFSMSdss Storage Administration Guide*, SC35-0423
- *DFSMSdss Storage Administration Reference*, SC35-0424
- *DFSMShsm Storage Administration Guide*, SC35-0421
- *DFSMShsm Storage Administration Reference*, SC35-0422
- *Introduction*, SC26-7397

RMF

- *RMF Report Analysis*, SC33-7991
- *RMF User's Guide*, SC33-7990

Data Facility Sort (DFSORT)

- *Application Programming Guide*, SC33-4035
- *Messages, Codes and Diagnostic Guide*, SC26-7050

Language Environment

- *Language Environment Concepts Guide*, SA22-7567
- *Language Environment Customization*, SA22-7564
- *Language Environment Debugging Guide*, GA22-7560
- *Language Environment Programming Guide*, SA22-7561
- *Language Environment Programming Reference*, SA22-7562
- *Language Environment Run-Time Messages*, SA22-7566

z/OS XL C/C++

- *Standard C++ Library Reference*, SC09-4949
- *z/OS XL C/C++ Compiler and Run-Time Migration Guide*, GC09-4913
- *z/OS XL C/C++ Language Reference*, SC09-4815
- *z/OS XL C/C++ Messages*, GC09-4819
- *z/OS XL C/C++ Programming Guide*, SC09-4765
- *z/OS XL C/C++ Run-Time Library Reference*, SA22-7821
- *z/OS XL C/C++ User's Guide*, SC09-4767

COBOL

- *Enterprise COBOL for z/OS and OS/390 Language Reference*, SC27-1408
- *Enterprise COBOL for z/OS and OS/390 Programming Guide*, SC27-1412
- *VisualAge COBOL for OS/390 and VM Language Reference*, SC26-9046
- *VisualAge COBOL for OS/390 and VM Programming Guide*, SC26-9049

PL/I

- *Enterprise PL/I for z/OS and OS/390 Language Reference*, SC27-1460
- *Enterprise PL/I for z/OS and OS/390 Messages and Codes*, SC27-1461
- *Enterprise PL/I for z/OS and OS/390 Programming Guide*, SC27-1457
- *VisualAge PL/I for OS/390 Compile-Time Messages and Codes*, SC26-9478

- *VisualAge PL/I for OS/390 Language Reference*, SC26-9476
- *VisualAge PL/I for OS/390 Programming Guide*, SC26-9473

High Level Assembler

- *Language Reference*, SC26-4940
- *Programmer's Guide*, SC26-4941

CPI-C

- *SAA CPI-C Reference*, SC09-1308

DB2 for z/OS

- *Administration Guide*, SC18-9840
- *Application Programming and SQL Guide*, SC18-9841
- *Codes*, GC18-9843
- *Command Reference*, SC18-9844
- *Installation Guide*, GC18-9846
- *Messages*, GC18-9849
- *SQL Reference*, SC18-9854
- *Utility Guide and Reference*, SC18-9855
- *DB2 for z/OS What's New?*, GC18-9856

ISPF

- *ISPF Dialog Developer's Guide*, SC34-4821
- *ISPF Dialog Tag Language*, SC34-4824
- *ISPF Planning and Customizing*, GC34-4814

WebSphere MQ for z/OS

- *An Introduction to Messaging and Queuing*, GC33-0805
- *Application Programming Guide*, SC34-6595
- *Application Programming Reference*, SC34-6596
- *Command Reference*, SC34-6597
- *Concepts and Planning Guide*, GC34-6582
- *Intercommunication*, SC34-6587
- *Messages and Codes*, GC34-6602
- *MQI Technical Reference*, SC33-0850
- *Problem Determination Guide*, GC34-6600
- *System Administration Guide*, SC34-6585

WebSphere Application Server for z/OS

- IBM Information Center for WebSphere Application Server
- IBM Information Center for WebSphere Application Server - Network Deployment z/OS
- *WebSphere on z/OS - Optimized Local Adapters* (Redpaper), REDP-4550

Tivoli Netview

- *Administration Reference*, SC31-8854
- *Automation Guide*, SC31-8853
- *Installation, Getting Started*, SC31-8872
- *User's Guide*, GC31-8849
- *Security Reference*, SC31-8870

SMP/E

- *Reference*, SA22-7772
- *User's Guide*, SA22-7773

Communications Server IP (TCP/IP)

- *API Guide*, SC31-8788
- *Configuration Guide*, SC31-8775
- *Configuration Reference*, SC31-8776
- *Diagnosis Guide*, SC31-8782
- *Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7534
- *IP and SNA Codes*, SC31-8791
- *Messages Volume 1*, SC31-8783
- *Messages Volume 2*, SC31-8784
- *Messages Volume 3*, SC31-8785
- *Messages Volume 4*, SC31-8786
- *Migration Guide*, SC31-8773

Web Enablement Toolkit

- *MVS Programming: Callable Services for High-Level Languages*, SA23-1377

TPF

- *Application Programming*, SH31-0132
- *Concepts and Structures*, GH31-0139
- *C/C++ Language Support Users Guide*, SH31-0121
- *General Macros*, SH31-0152
- *Library Guide*, GH31-0146
- *System Macros*, SH31-0151

TPF Database Facility (TPFDF)

- *Database Administration*, SH31-0175
- *General Information Manual*, GH31-0177
- *Installation and Customization*, GH31-0178
- *Programming Concepts and Reference*, SH31-0179
- *Structured Programming Macros*, SH31-0183
- *Utilities*, SH31-0185

TSO/E

- *Administration*, SA22-7780
- *Customization*, SA22-7783
- *System Programming Command Reference*, SA22-7793
- *User's Guide*, SA22-7794

Communications Server SNA (VTAM)

- *IP and SNA Codes*, SC31-8791
- *Messages*, SC31-8790
- *Network Implementation* , SC31-8777
- *Operations*, SC31-8779
- *Programming*, SC31-8829
- *Programmers' LU6.2 Guide*, SC31-8811
- *Programmers' LU6.2 Reference*, SC31-8810
- *Resource Definition Reference*, SC31-8778

Security Server (RACF)

- *RACF General User's Guide*, SA22-7685
- *RACF Messages and Codes*, SA22-7686
- *RACF Security Administrator's Guide*, SA22-7683
- *RACF Callable Services*, SA23-2293
- *z/OS Security Server RACF Data Areas*, GA32-0885

Integrated Cryptographic Service Facility (ICSF)

- *ICSF Administrator's Guide*, SA22-7521
- *ICSF Application Programmer's Guide*, SA22-7522
- *ICSF Messages*, SA22-7523
- *ICSF Overview*, SA22-7519
- *ICSF System Programmer's Guide*, SA22-7520

Other IBM publications

- *IBM 3270 Information Display System: Data Stream Programmer's Reference*, GA23-0059
- *Input/Output Configuration Program User's Guide and Reference*, SB0F-3741
- *NTO Planning, Migration and Resource Definition*, SC30-3347
- *Planning for NetView, NCP, and VTAM*, SC31-7122
- *SNA Formats*, GA27-3136
- *X.25 NPSI Planning and Installation*, SC30-3470
- *z/Architecture Principles of Operation*, SA22-7832

ALCS product kit

- *ALCS PDF Product Kit*, SK3T-6944

SITA publications

- *ACS protocol acceptance tool*, SITA document 032-1/LP-SD-001
- *Communications control procedure for connecting IPARS agent set control unit equipment to a SITA SP*, SITA document P.1024B (PZ.7130.1)
- *P.1X24 automatic testing (with UCTF on DIS)*, SITA document 032-1/LP-SDV-001
- *P.1024 Test Guide*, SITA Document PZ.1885.3
- *Status Control Service Step 2: Automatic Protected Report to ACS*, SITA document 085-2/LP-SD-001
- *Synchronous Link Control Procedure*, SITA Document P.1124

SITA produces a series of books which describe the SITA high level network and its protocols. These may be obtained from:

```
Documentation Section
SITA
112 Avenue Charles de Gaulle
92522 Neuilly sur Seine
France
```

Other non-IBM publications

Systems and Communications Reference Manual (Vols 1-7). This publication is available from the International Air Transport Association (IATA). You can obtain ordering information from the IATA web site <http://www.iata.org/> or contact them directly by telephone at +1(514) 390-6726 or by e-mail at Sales@iata.org.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Product Number: 5695-068

SH19-6955-23