Airline Control System
Version 2.4.1

Installation and Customization

SH19-6954-23

Note

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

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.

© Copyright International Business Machines Corporation 2003, 2019.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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 Installation and Customization Manual is intended to help the customer to install, run, and optimize the performance of Airline Control System Version 2 Program Number 5695-068.

This Installation and Customization Manual also documents General-Use Programming Interface and Associated Guidance Information, Product-Sensitive Programming Interface and Associated Guidance Information, and Diagnosis, Modification, and Tuning Information.

General-Use programming interfaces allow the customer to write programs that obtain the services of Airline Control System Version 2 Program Number 5695-068.

General-Use Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

General-use Programming Interface

General-Use Programming Interface and Associated Guidance Information...

End of General-use Programming Interface

Product-Sensitive programming interfaces allow the customer installation to perform such tasks as diagnosing, modifying, 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

© Copyright IBM Corp. 2003, 2019

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

This Installation and Customization Manual also documents Diagnosis, Modification, and Tuning Information, which is provided to help the customer do modification, customization, diagnosis, and tuning of Airline Control System Version 2, Program Number 5695-068

Attention

Do not use this Diagnosis, Modification, and Tuning information as a programming interface.

Diagnosis, Modification, and Tuning Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Diagnosis, Modification, and Tuning Information

Diagnosis, Modification, and Tuning Information....

End of Diagnosis, Modification, and Tuning Information

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 contains installation and customization information for 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.
- 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. MVS, s/390, and "OS/390®" also refer to operating systems.
- DB2®, CICS®, AIX®, Tivoli®, NetView®, WebSphere®, RACF®, and IMS are IBM products.

Who should read this book

This book is intended for system programmers, system analysts, and customer operations personnel who are involved in installing, generating, and customizing an ALCS system.

How this book is organized

This book is divided into the following task-oriented chapters and appendixes:

© Copyright IBM Corp. 2003, 2019

Appendix C, “Sample symbolic line number conversion,” on page 550
Sample code to convert an SLC line number to and from the ALCS internal identifier.

Appendix D, “Sample logon mode table,” on page 551
Sample code to define a VTAM logon mode table

Appendix E, “LU 6.1 Communication generation,” on page 555
Sample code to define LU 6.1 links to IMS and CICS.

Appendix F, “Network Control Program sample definition,” on page 557
Sample code to define a large test network.

Appendix G, “ALCI sample definition,” on page 573
Sample code to define an ALCI test network.

Appendix H, “Generating a test ALC terminal,” on page 575
Describes how to define a test ALC terminal.

Appendix J, “Sample definition for OCTM sequential file,” on page 579
Describes how to define sequential files for the online communication table management (OCTM) facility.

ALCS macro and control statement syntax

The notation used to define the ALCS macro and control statement syntax in this book consists of symbols, fonts, and format. They are described in detail below.

Symbols

The following symbols define the macro and control statement format. Do not use them in the actual macro or control statement:

Braces
{ }

Brackets
[ ]

Ellipsis
...

OR symbol
|

Use the following symbols as specified:

Asterisk
*

Comma
,

Hyphen
-

Equals sign
=

Parentheses
( )

Period
.

Colon
:

Quotation
'

Fonts

The following fonts indicate a keyword or a variable, and the default if there is one:

variable
A variable, for which specific information is to be substituted.

KEYword
A keyword. Enter either KEY or KEYWORD.

default
The default. If the parameter is omitted this option is assumed.

Format

Braces: Indicate options where one of the options *must* be selected. For example:

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

or:

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

means select one of A, or B, or C.

Brackets: Indicate options that can be omitted, or one of which can be selected. For example

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

means either omit the parameter or select one of A, or B, or C.

Nested braces or brackets: Mean that operand selection depends on the selection of the operand of a higher level of nesting. For example:

```
[level_1[,level_2[,level_3]]]
```

means that the following selections are possible:

- Omit all three operands
- *level_1* only
- *level_1* qualified by *level_2*
- *level_1* qualified by *level_2* qualified by *level_3*.

Do not specify a lower level without the higher level.

Ellipses: Mean that the preceding item or group can be repeated more than once in succession. For example:

(*option*,...)

means specify one or more options, separated by commas within parentheses.

Register notation

Within this book, general registers are referred to as follows:

Table 1. Notation conventions used in this book	
Text examples	**Code examples**
General register 0 (RAC)	R00
General register 1 (RG1)	R01
General register 2 (RGA)	R02
General register 3 (RGB)	R03
General register 4 (RGC)	R04
General register 5 (RGD)	R05
General register 6 (RGE)	R06
General register 7 (RGF)	R07
General register 8 (RAP)	R08
General register 9 (REB)	R09
General register 10 (RLA)	R10
General register 11 (RLB)	R11
General register 12 (RLC)	R12
General register 13 (RLD)	R13
General register 14 (RDA)	R14
General register 15 (RDB)	R15

Chapter 1. Using ALCS ISPF panels

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

```
  File  Options
------------------------------------------------------------------------------
                            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: The ALCS primary menu

Online help for the ALCS ISPF panels

Press F1 to request help for a task. The type of help 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**.

Table 2 on page 1 shows where you can find more information about tasks in the main ALCS ISPF panel.

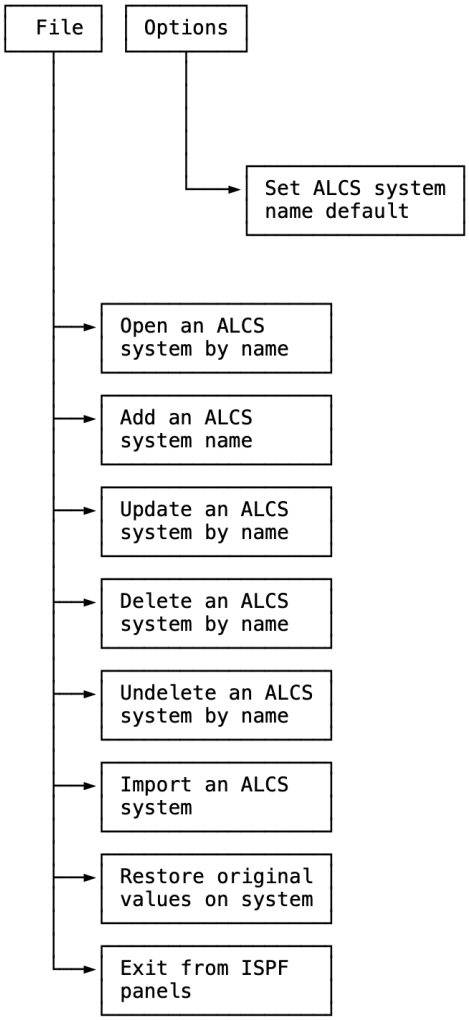
Table 2. Where to find more information about the primary menu ISPF panel

Task	
General	Figure 2 on page 2
Installation	Table 7 on page 50
Generation and Database creation	Table 8 on page 61
Maintenance	*ALCS Operation and Maintenance*
Operations	*ALCS Operation and Maintenance*
Application Development	*ALCS Application Programming Guide*

© Copyright IBM Corp. 2003, 2019

Route maps for the File and Options action bars

Figure 2 on page 2 shows the actions available for the **File** and **Options** action bars in the ALCS primary menu. The **Help** action is not shown. Most ALCS ISPF panels have action bars in addition to the menu-selection fields.




```
        ALCS primary menu
 ──────────────────────────────────────

 Actions:
   _ 1.  Installation
     2.  Generation and database creation
     3.  Maintenance
     4.  Operations
     5.  Application Development
```

Table 2 shows where you can find more information about tasks in the main ALCS ISPF panel.

1. Table 2 on page 1

Figure 2. Route map for the ALCS primary menu ISPF panel

Installing the ISPF panels

See the *ISPF Dialog Management Guide and Reference* for information about:

- The data sets ISPF uses
- Getting ready to run on MVS
- Dialog management services

See *TSO Extensions Version 2 Command Reference* for information on ALTLIB.

Invoking the panels for ALCS installation

When you first install ALCS, you need to load some information from the shipment tape. This is explained in the *ALCS Program Directory*. After you do this, you can use the ALCS ISPF panels to complete the installation process.

If you use the temporary ("bootstrap") data set name suggested in the *ALCS Program Directory*, then you invoke the ALCS ISPF panels with a command of the form:

```
TSO EXEC 'DXC.V2R4M1.BOOTSTRP.JCL(DXCE000)'
```

Do not use this method to invoke the ALCS ISPF panels for any purpose other than completing the ALCS installation. (This "bootstrap" data set is not updated by IBM supplied maintenance - you will probably want to delete it after you complete the ALCS installation.)

Invoking the panels for production use

The installation process loads the ALCS ISPF panels (and associated components) into the library *hlq*.DXCISPF, where *hlq* represents the data set name qualifiers that your installation uses for the ALCS SMP/E-controlled data sets.

Before you use the production version of the panels, you must perform a panel customization process. This process includes:

1. Update your TSO environment to specify libraries as shown in Table 3 on page 3.

   A sample exec to perform these allocations is supplied as *hlq*.DXCISPF(DXCEP241).

Note: Be sure to use the NOLIB parameter when you invoke the ALCS ISPF panels direct from TSO. If you omit NOLIB, the ALCS ISPF panels write information to the SMP/E-controlled library that contains the panels.

Table 3 on page 3 lists the DDNAMES which must be set up.

Table 3. DDNAMES required for ISPF panels

DDNAME	Description	Library name
ISPPLIB	Panel library	*hlq*.DXCISPF
ISPMLIB	Message library	*hlq*.DXCISPF
ISPSLIB	Skeleton library	*hlq*.DXCISPF
SYSPROC or SYSEXEC	Command procedures library	*hlq*.DXCISPF
ISPTLIB	Table input library	See note 1.
ISPTABL	Table output library	See note 1.
ISPPROF	User profile library	See note 2.

1. “Allocating table libraries” on page 3 gives an overview of these definitions.
2. The ALCS ISPF panels do not place any special requirements on the ISPPROF library (other than the requirements for ISPF itself), but you must make sure it is large enough to hold the additional data.
3. The *hlq* represents the data set name qualifiers that your installation uses for the ALCS SMP/E controlled data sets.

Allocating table libraries

The ALCS ISPF panels make extensive use of the ISPF table services to store and retrieve common system information. This information is common to a **system definition** and is shared between a group of users working with that definition.

To understand the use of these definitions consider an organization with two ALCS systems (a live system and a test system).

There are a number of different projects in the organization:

- Reservation
- Cargo
- Departure control

Each of these is a separate definition, but each is related to the others. Figure 3 on page 4 shows these five definitions.

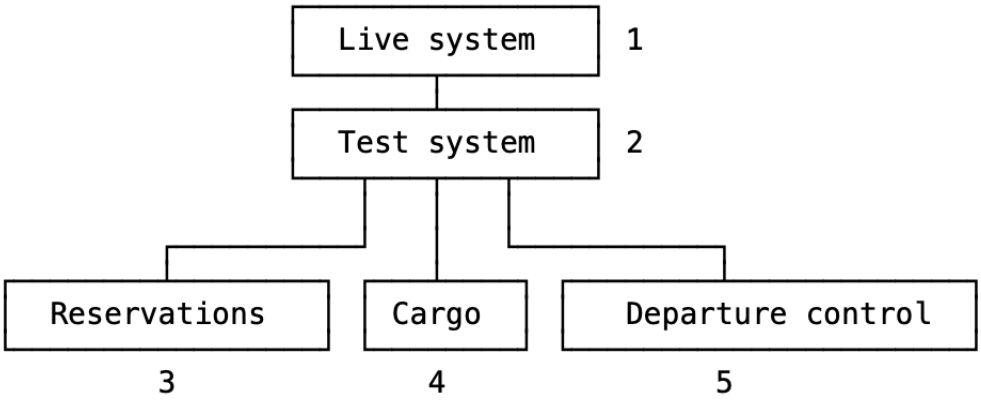


Figure 3. Example of five different system definitions

You can generate the live system (1) either:

- From scratch
- By **adding** a new system (using the IBM-supplied SMP/E system as a model)

Note: When you use an existing system as a model for a new system, you produce a copy (not an inheritance). If you modify the model, the changes are not copied to the new system.

You can copy and modify the definitions of the live system (1) to produce the test system (2).

You can base the three applications projects (3, 4, and 5) on the test system (2).

You should consider the following points:

- **Allocating libraries**

  Any user with write access to the system-definitions-table library can update (or overwrite) the system definitions shared by your users.
- **Output library**

  The results are unpredictable if you allow more than one user to specify a system-definitions-table library as their output-table library (ISPTABL).

Updating the TSO ISPF menu panel to include ALCS function

You can update either:

- The ISPF primary menu panel
- One of its subsidiary menu panels

to allow your users easy access to the ALCS ISPF panels.

To do this, you update the ISPF menu definition for the menu where you want to add the ALCS functions. Figure 4 on page 5 shows an example of how you might do this.

You must add a line into the body of the panel definition to add the ALCS ISPF panels as an option. In Figure 4 on page 5 this is option 14.

You must also update the processing section to invoke the ALCS ISPF panels application when the option is selected. This can use either:

- The primary exec DXCE000 with the NOLIB argument.

  To use this, you *must* allocate the required data sets at TSO logon or ISPF startup (see "Allocating table libraries" on page 3 for more information).
- A version of the sample startup exec DXCEP241 which you have customized for your own needs.

  This allocates the required data sets when you start the ALCS ISPF panels application.

Figure 4 on page 5 shows how you might update the processing section to use a customized version of DXCEP241 in a library called MY.CUSTOM.PANELS.

```
%OPTION  ===>_ZCMD
%
%  .
%  .
%  .
% 14 +ALCS      -  ALCS ISPF panels
%  .
%  .
%  .
)PROC
  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
                .
                .
                .
               14,'CMD(exec ''my.custom.panels(dxcep241)'' ''smpe'') NEWAPPL(DXC)'
                .
                .
                .
                )
```

Figure 4. Customizing DXCEP241

Additional ISPF commands

You can use the following standard ISPF commands with the ISPF panels:

- PFSHOW
  If you want to display PF key settings for any panel, enter PFSHOW. Enter PFSHOW OFF to remove the PF-key display.

  Enter PFSHOW with no parameters to switch between these options.
- PANELID Enter PANELID to show the internal panel name for any panel. This name is useful when you are reporting a fault or requesting an enhancement.

  Enter PANELID with no parameters to toggle the display of the panel name.

Chapter 2. Installing ALCS

In this section, any reference to an IBM licensed program is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

General considerations

ALCS can run on any processor supported by z/OS in z/Architecture® mode, provided that the processor can be configured with enough primary and auxiliary storage to satisfy ALCS minimum requirements.

The minimum required level of operating system for ALCS Version 2 Release 4 is z/OS (5694-A01) at currently supported release levels in z/Architecture mode

ALCS requires a minimum of 32MB of primary storage.

The amount of auxiliary storage required depends on the amount of data the application programs use and the system's transaction rate.

Required licensed programs

A complete list of required programs with their current release levels is given in the *Licensed Program Specifications* that accompany this program.

Defining ICSF support for ALCS

ALCS Concepts and Facilities provides an overview of the Integrated Cryptographic Service Facility (ICSF) support in ALCS. Use the ICSF parameter of the SCTGEN macro to specify the ICSF support in ALCS.

The key label names that are defined in the cryptographic key data set (CKDS) must be specified in a CSFKEYS RACF profile or a general profile that is used to allow access to all keys or a subset of keys that have specific naming characteristics, for example:

```
ALCS.*
```

If encrypted keys are to be used, the SYMCPACFWRAP(YES) option must be specified for the CSFKEYS RACF class.

Security Authorization Facility (SAF) profiles for ALCS

IBM strongly recommends that you use RACF (or other compatible product) to protect your ALCS systems against unauthorized users. Preventing unauthorized access to the APF authorized libraries is the minimum recommended level of access control.

This section is intended to help you plan and implement appropriate security for ALCS. It does not cover more general security considerations for your installation.

In addition to reading this section, you will need to consider:

- Publications and other information specific to the security product (for example, RACF) or products that your installation uses.
- Security guidelines, standards, and practices specific to your company or installation.
- Any legal requirements or restraints that might apply to your installation's security, particularly in connection with safeguarding personal or other sensitive data.
- Physical and other security characteristics of communication links and terminal equipment that connect to your ALCS systems.

- Physical and other security characteristics of other computer systems, including personal computers, that connect to your ALCS systems.
- Program changes that may be required on other computer systems, including personal computers, that connect to your ALCS systems.

Note: This section assumes that you use IBM's Resource Access Control Facility (RACF) as the external security manager (ESM) in your installation. If you use a different ESM, you may need to modify the techniques described here. If you have no ESM, IBM recommends that you code an MVS router exit `ICHRTX00` (see "ICHRTX00 - MVS Router Exit with no external security manager" on page 17).

Why you need to protect ALCS resources

Your ALCS systems probably provide services and maintain data that are important to your enterprise. If you do not adequately protect your ALCS systems against unauthorized access, your systems are at risk. Risks include:

- Accidental or malicious damage to your systems.
- Exposure of confidential, personal, or other sensitive information.
- Exploitation of your system for personal gain.

ALCS resources you can protect

To minimize risks from unauthorized access, you can protect the following ALCS system resources:

- ALCS libraries.

  These are the libraries that contain ALCS source and executable modules, configuration information, and so on.

  You need to be careful whom you allow to update these libraries. Anyone with this authority can accidentally or deliberately damage your ALCS systems.

  Some ALCS libraries are authorized by the MVS Authorized Program Facility (APF). You need to be particularly careful whom you allow to update APF-authorized libraries. Anyone with this authority can bypass RACF protection of other data and programs on your MVS system - and possibly other connected systems.

  Only staff who install, apply maintenance (fixes) to, or customize ALCS need UPDATE authority for ALCS libraries.
- Application libraries.

  These are the libraries that contain your application source and executable modules, and so on.

  You need to be careful whom you allow to update these libraries. Anyone with this authority can accidentally or deliberately damage your applications.

  Only staff who develop, install, apply maintenance (fixes) to, or customize your applications need UPDATE authority for application libraries.
- ALCS online data sets.

  These are the real-time database, configuration data sets, sequential files, and so on.

  You need to be careful whom you allow to update these data sets. Anyone with this authority can accidentally or deliberately corrupt or delete valuable data, or damage your applications, or both.

  Depending on the information that you store on these data sets, you may want to control who can read them. This is likely to apply if you store confidential or personal information, or information that could be valuable to a third party.

  Don't forget that backups, database update logs, and so on can also contain sensitive data - you may want to protect them too.
- ALCS application node names.

These are the names that end users provide to identify to which ALCS system they want to log on; in the VTAM command `LOGON APPLID(`*name*`)`, *name* is the application node name.

You can protect ALCS application node names to prevent unauthorized persons from starting an ALCS system that "masquerades" as a production system (it looks like the production system to end users).

- ALCS online systems.

  You can protect ALCS online systems to control who can be end users. In particular, you may want to control who can be end users of your production ALCS systems.

- ALCS online system functions.

  You can protect specific functions of your ALCS systems to control which end users can invoke them. For example, you can control which end users can use computer room agent set (CRAS) or Prime CRAS ALCS commands. Similarly, you can control which end users can use specific functions of your application.

- ALCS terminals.

  These are the terminals that connect to your ALCS systems. They can include both SNA terminals such as IBM 3270 and non-SNA terminals such as ALC terminals.

  You can protect ALCS terminals to control which end users can use which terminals.

Your installation may have a number of ALCS systems. For example, you may have:

- One or more production systems
- Test systems for application development
- Test systems for verifying changes before production use.

Typically these systems have different security requirements. But for *all* ALCS systems, you need to be careful whom you allow to update APF-authorized libraries.

Who needs to access ALCS resources

Different staff who work with or use your ALCS systems need different access authority for ALCS resources. Table 4 on page 9 summarizes the main categories of staff, and the access authority they require. The categories shown in Table 4 on page 9 may not correspond to your staff organization. For example, your application programmers may also be end users (especially of test systems) and your operations staff may also be end users (through ALCS CRASs).

Table 4. Who needs to access ALCS resources

Category	Role	ALCS libraries	Application libraries	ALCS data sets	Application node and XCF group names	ALCS online systems	ALCS online system functions	Terminals
System programmers	Install, apply maintenance (fixes) to, or customize ALCS.	UPDATE	NONE	NONE	NONE	NONE	NONE	NONE
Application programmers	Develop, install, apply maintenance (fixes) to, or customize your applications.	READ	UPDATE	NONE	NONE	NONE	NONE	NONE
Operations	Run ALCS online systems, run ALCS offline (batch) utilities, perform backups, and so on.	READ	READ	Note “1” on page 10	READ	NONE	NONE	NONE
End users	Access and use ALCS online systems from terminals.	NONE	NONE	NONE	NONE	READ	Note “2” on page 10	READ

Notes:

1. *(SAFTNOP)* Access authority depends on the data set and on whether or not the ALCS test database facility is in use. READ access is required for data sets that the online monitor accesses read-only, UPDATE access is required for data sets that the online monitor accesses read-write, and ALTER access is required for data sets that the online monitor creates or deletes.
2. *(SAFTNEU)* Access authority depends on the function and the end user. READ access is required for functions that the end user is permitted to use.

Protecting the ALCS product libraries

In the following examples, *racfadm* is a user with RACF SPECIAL, and *alcssysp* is a RACF group of ALCS system programmers *userid1* and *userid2* responsible for installing and maintaining the ALCS product on your system.

Define a RACF generic profile appropriate for your security requirements. For example:

```
ADDGROUP DXC OWNER(racfadm)
ADDSD 'DXC.*' UACC(READ) OWNER(racfadm)
ADDGROUP alcssysp OWNER(racfadm)
CONNECT userid1 GROUP(alcssysp)
CONNECT userid2 GROUP(alcssysp)
PERMIT 'DXC.*' ID(alcssysp) ACCESS(ALTER)
SETROPTS GENERIC(*) GLOBAL(*) REFRESH
```

Note: `ACCESS(ALTER)` on `PERMIT 'DXC.*' ID(alcssysp) ACCESS(ALTER)` is required to maintain the product libraries. `ACCESS(READ)` should be specified for all other ALCS system programmers and application developers.

Implementing ALCS security - Overview

To provide security, ALCS uses the MVS Security Authorization Facility (SAF) to route authorization requests to an external security manager (ESM), for example IBM's RACF. ALCS does no security verification of its own.

You must define the *resources* that you want to protect and the *users* who you permit to access these resources, and you must specify the level of access that each user has to each resource. You will almost certainly find it convenient to define *groups* of users. Instead of directly permitting each user to access a resource, you can connect many users to a group and permit the group to access the resource.

Figure 5 on page 11 shows four users, A, B, C, and D who are permitted access to the resource R. User A is directly permitted access to R. Users B, C, and D are connected to group G which is permitted access to R.

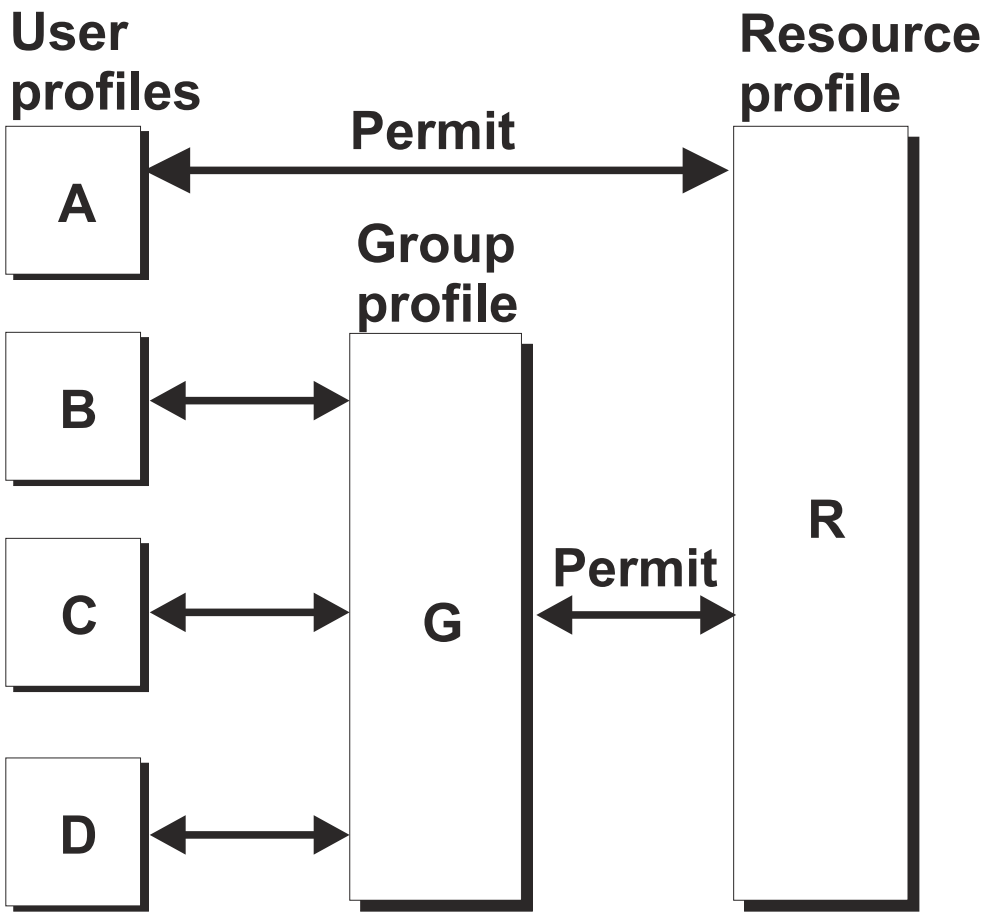


Figure 5. Permitting access to resources - users and groups

The information that the ESM holds for each resource, user, and group is called a *profile*[1]. Note that information about which users and groups can access a resource is in the resource profile. If many users require access to the same resource, there may not be room enough in the resource profile to hold references to all the individual user profiles. You can overcome this restriction by using groups - each group requires only a single reference in the resource profile.

Profiles belong to *classes*. For example, the class USER contains all the user profiles. ALCS security uses many of the same classes that other MVS applications and subsystems use, together with the ALCSAUTH class which contains profiles specific to ALCS security.

The following sections describe how you define profiles for ALCS security. They do not provide a comprehensive description of the facilities available with RACF or other ESMs, and you should read them in conjunction with the publications and other documentation for your ESM.

No ESM decision:

The decision to allow access to a resource is made by the ESM and ALCS follows that decision. In some cases, the ESM may not be able to make a decision - for example, if the ESM is inactive, or if there is no profile for the resource.

In the following sections, boxes like this one explain how ALCS reacts when the ESM cannot make a decision.

[1] A *generic profile* holds information about several resources with similar names.

User (USER and GROUP) profiles

You define user IDs for ALCS users, including end users, in the same way as for any other MVS application or subsystem. A person who accesses for example TSO, CICS, and ALCS will normally have the same user ID (the same USER profile) for all these systems.

IBM recommends that you define *all* users of ALCS, including end users. User IDs for ALCS end users can be any string of 1 through 8 characters that is acceptable to your ESM, but cannot be 'LOGOFF' or 'ZLOGF'.

RACF commands that you use with USER and GROUP profiles include:

ADDUSER
Defines a user by creating a profile in the USER class.

ADDGROUP
Defines a group by creating a profile in the GROUP class.

ALTUSER
Alters fields in a defined profile in the USER class.

ALTGROUP
Alters fields in a defined profile in the GROUP class.

CONNECT
Connects a user to a group. This permits the user to access the resources that the group can access.

PERMIT
Permits a user or group to access a resource.

For certain categories of user who only ever access ALCS, you may want to exploit the WHEN information in the user profile. This restricts access to the system to specified dates and times. For example, an airline reservations installation might restrict a general sales agent to access the system only during working hours. You can also control an end user's access to ALCS by using the WHEN parameter of the RACF PERMIT command. For example, you could use the WHEN parameter to permit CRAS authority to an end user only when they logon at a terminal in a secure area.

You can add installation-defined data to existing profiles by using the DATA parameter on the ALTUSER and ALTGROUP commands respectively. An application program can read the data using the AUTHC monitor-request macro with the DATA parameter.

No ESM decision:

When an end user logs on, ALCS checks that the user ID and password are correct. This check is called VERIFYing the user ID.

If the ESM can not make a decision for a VERIFY, ALCS assumes the user ID is not authorized to use ALCS. The end user cannot enter ALCS commands or application input messages.

Default user IDs

For each ALCS communication resource (for example, terminal), the ALCS communication generation specifies whether or not the end user must logon - that is, provide their user ID and password - before they can access ALCS. For good security, you should normally specify that the end user must logon, but there can be exceptions, as follows:

- ALCS does not support a logon for some types of communication resource (for example, WTTY).
- Some programmable communication resources may need program changes to provide a user ID and password. You may need to specify that no logon is required until these program changes are complete.
- You may have physical security measures which ensure that only authorized staff can access certain terminals. You may want to allow these staff to access ALCS without a logon.
- You may have certain terminals where a restricted set of functions are available. You may want to allow *anyone* to use these terminals to access ALCS without a logon.

For these exception cases, the end user does not provide a user ID. Instead, the ALCS communication generation provides a *default user ID*. Note that ALCS allows you to specify the same default user ID for

more than one communication resource. When you specify default user IDs in the ALCS communication generation, you must also create profiles for them in the USER class (see "User (USER and GROUP) profiles" on page 12).

ALCS VERIFYs default user IDs (without checking the passwords) when it loads communication configuration information. This happens during ALCS restart and when the ALCS operator enters the ZACOM LOAD command. If a VERIFY fails (the user ID is not valid or is not authorized to access ALCS) then ALCS takes action depending on the type of communication resource:

- If ALCS does support a logon from that type of resource, ALCS ignores the default user ID and requires a logon.
- If ALCS does not support a logon from that type of resource, ALCS does not add the communication resource.

No ESM decision:

ALCS VERIFYs default user IDs without checking the password.

If the ESM can not make a decision for a VERIFY, ALCS assumes that the verify failed.

You must permit default user IDs access to the resources that the actual end users require. In particular, if you define profiles to protect ALCS CRASs (see "ALCSAUTH profiles - CRAS authority" on page 16) and you assign default user IDs to CRASs then you must permit the default IDs READ access to the corresponding CRAS profiles.

You must also permit default user IDs READ access to a special profile in the ALCS group (see "ALCSAUTH profiles - default user IDs" on page 15). This allows your security coordinator (who maintains ESM profiles) to control which user IDs can be default user IDs.

NetView user IDs

ALCS VERIFYs NetView user IDs without checking the password. To make your ALCS systems secure, you must ensure that operator identification and password checking is done using a SAF security product. To do this, specify SECOPTS.OPERSEC=SAFDEF or SECOPTS.OPERSEC=SAFCHECK or SECOPTS.OPERSEC=SAFPW in the NetView CNMSTYLE member (see *IBM Tivoli Netview for z/OS Administration Reference*).

No ESM decision:

ALCS VERIFYs NetView user IDs without checking the password.

If the ESM can not make a decision for a VERIFY, ALCS assumes that the verify failed.

You must permit NetView user IDs access to the resources that they require. In particular, if you define profiles to protect ALCS CRASs (see "ALCSAUTH profiles - CRAS authority" on page 16) and you define NetView user IDs as CRASs then you must permit the NetView IDs READ access to the corresponding CRAS profiles.

Terminal (TERMINAL and GTERMINL) profiles

You can create profiles to control who can use particular terminals. For example, you can use terminal profiles to:

- Allow only specified users to logon at a particular terminal.

  For example, you might want to prevent all but a select group of users from logging on at certain terminals.
- Restrict specified users to logon at a particular terminal or terminals.

  For example, you might want to restrict some users so that they can only logon at certain terminals.
- Restrict the times when a terminal can be used.

  For example, you might want to specify that certain terminals can only be used during working hours.

You use the RACF RDEFINE command to define a terminal by creating a profile in the TERMINAL or GTERMINL class. For SNA terminals, the profile name is the logical unit (LU) name of the terminal. For terminals known only to ALCS (for example, ALC terminals), the profile name is the ALCS CRN.

If you create profiles for terminals where ALCS end users logon, you must ensure that the end users have READ access to the terminals that they use. If you create profiles for terminals that have ALCS default user IDs (see "Default user IDs" on page 12), you must ensure that the default user IDs have READ access to the terminals.

No ESM decision:

When an end user logs on at a terminal, ALCS checks that the user ID is authorized to log on from that terminal.

If the ESM can not make a decision, ALCS assumes the user ID is authorized.

Application (APPL) profiles

You can create profiles to control who can log on to particular ALCS systems.

You use the RACF RDEFINE command to define an ALCS system by creating a profile in the APPL class. The profile name is the VTAM application logical unit (LU) name of the ALCS system - for ALCS systems that use a VTAM generic LU, it is the generic LU name.

If you create APPL profiles for ALCS systems, you must ensure that the end users have READ access to the ALCS systems that they use. You must also ensure that default user IDs (see "Default user IDs" on page 12) have READ access to the ALCS systems that specify them.

No ESM decision:

When an end user logs on to an ALCS system, ALCS checks that the user ID is authorized to log on to that ALCS system.

If the ESM can not make a decision, ALCS assumes the user ID is authorized.

ACB (VTAMAPPL) profiles

You can create a profile to control who can run an ALCS system with a particular VTAM application logical unit (LU) name. Note that ALCS does not use the password protection that VTAM provides for ACBs.

You use the RACF RDEFINE command to define a VTAM application logical unit (LU) name by creating a profile in the VTAMAPPL class. The profile name is the VTAM application logical unit (LU) name of the ALCS system - for ALCS systems that use a VTAM generic LU, it is the generic LU name.

If you create VTAMAPPL profiles for ALCS systems, you must ensure that the people who run ALCS systems have READ access to the corresponding VTAM LU names. (End users do not require this access authority.)

No ESM decision:

When an ALCS system starts, ALCS checks that its address space user ID is authorized to use the VTAM application LU name. This is the user ID on the job card of the ALCS job, or the user ID assigned to the ALCS started task by the MVS started procedures table.

If the ESM can not make a decision, ALCS assumes the user ID is authorized.

ALCSAUTH profiles - overview

You can create a variety of profiles in the ALCSAUTH class to control access to ALCS facilities. Each profile has a name that is similar to an MVS data set name. That is, it comprises two or more parts joined by period (full stop) characters. Each part is a character string of at least 1 and up to 8 characters.

For all profiles in the ALCSAUTH class, the first part of the profile name is the VTAM application logical unit (LU) name - for ALCS systems that use a VTAM generic LU, it is the generic LU name. For example, to

control who can access ALCS systems with Prime CRAS authority, you define profiles with names of the form:

luname.CRAS.PRIME

If you have a production ALCS system that uses the LU name 'ALCSPROD' and test systems that use LU names 'ALCST1', and 'ALCST2', you define the profiles:

ALCSPROD.CRAS.PRIME
ALCST1.CRAS.PRIME
ALCST2.CRAS.PRIME

You use the RACF RDEFINE command to define a profile in the ALCSAUTH class. You use the RACF PERMIT command to grant access to the profile. You must PERMIT all users who need to use the function or facility that the profile protects. For some of these profiles, you must PERMIT end users; remember that these can include default user IDs (see "Default user IDs" on page 12) and users who access ALCS through NetView (see "NetView user IDs" on page 13).

ALCSAUTH profiles - HFS functions

You can create a profile to control which user IDs can access functions which update the ALCS hierarchical file system (HFS). This profile has a name of the form:

luname.ALCSAPPL.HFS

ALCS automatically checks access to this profile for:

HFS commands that update the HFS
PC file transfer

No ESM decision:

When an end user issues a command to update the HFS, or starts PC file transfer, ALCS checks that user ID has UPDATE access to the corresponding HFS profile in the ALCSAUTH class.

If the ESM can not make a decision, ALCS assumes the user ID is authorized.

ALCSAUTH profiles - XCF group names

You can create profiles to control who can start ALCS instances that join XCF groups. These profiles have names of the form:

luname.XCFGROUP

For example, if your production system comprises a group of ALCS instances, and the generic LU name that the group uses is 'ALCSPROD', you define the profile:

ALCSPROD.XCFGROUP

If you create ALCSAUTH profiles for XCF groups, you must ensure that the people who run ALCS systems have READ access to the profiles. (End users do not require this access authority.)

No ESM decision:

When an ALCS instance joins an XCF group, ALCS checks that its address space user ID has READ access to the corresponding XCFGROUP profile in the ALCSAUTH class. This is the user ID on the job card of the ALCS job, or the user ID assigned to the ALCS started task by the MVS started procedures table.

If the ESM can not make a decision, ALCS assumes the user ID is authorized.

ALCSAUTH profiles - default user IDs

You can create profiles to control which user IDs can be default user IDs of your ALCS systems (see "Default user IDs" on page 12). These profiles have names of the form:

luname.NOLOG

For example, if your production system uses the LU name 'ALCSPROD', you define the profile:

ALCSPROD.NOLOG

If you create ALCSAUTH profiles for default user IDs, you must ensure that the default user IDs have READ access to the NOLOG profiles.

No ESM decision:

When ALCS loads communication configuration information, it checks that the default user ID associated with each communication resource has READ access to the corresponding NOLOG profile in the ALCSAUTH class.

If the ESM can not make a decision, ALCS assumes the user ID is authorized.

ALCSAUTH profiles - CRAS authority

You can create profiles to control which user IDs can access your ALCS systems with CRAS authority. These profiles have names of the form:

luname.CRAS.*type*

Where *type* is one of:

PRIME
Prime CRAS authority

ALT1-16
alternate CRAS 1 through 16 authority

ALT
alternate CRAS 17 through 256 authority.

For example, if your production system uses the LU name 'ALCSPROD', you define the profiles:

ALCSPROD.CRAS.PRIME
ALCSPROD.CRAS.ALT1-16
ALCSPROD.CRAS.ALT

If you create ALCSAUTH profiles for CRAS authority you must ensure that CRAS users have READ access to the appropriate profiles. Remember that these can include default user IDs (see “Default user IDs” on page 12) and users who access ALCS through NetView (see “NetView user IDs” on page 13).

No ESM decision:

When an end user logs on at a terminal that is a CRAS, ALCS checks that the user ID has READ access to the corresponding CRAS authority profile in the ALCSAUTH class.

Also, when ALCS loads communication configuration information that associates a default user ID with a terminal that is a CRAS, it checks that the default user ID has READ access to the corresponding CRAS authority profile in the ALCSAUTH class.

If the ESM can not make a decision, ALCS assumes the user ID is authorized.

ALCSAUTH profiles - application functions

You can create profiles to control which user IDs can access various functions and facilities of your ALCS applications. These profiles have names of the form:

luname.ALCSAPPL.*appl*.*qual*

Where:

appl
CRN of an ALCS application

qual
One or more qualifiers joined by period (full stop) characters. Together these qualifiers identify a facility or function of your application.

For example, if you have a seat reservation application you might want to restrict who can invoke the schedule-change function. If your production system uses the LU name 'ALCSPROD', your seat reservation application has the CRN 'RES0', and you chose the qualifier 'SC' for the schedule-change function, you define the profile:

ALCSPROD.ALCSAPPL.RES0.SC

ALCS does not check access to these profiles automatically. Your application must explicitly call the ALCS AUTHC service to check authorization. For example, an assembler language application can check authority for the schedule-change function with the following AUTHC macroinstruction:

```
        AUTHC ALCSENTITY='RES0.SC',ATTR=READ
```

No ESM decision:

When an ALCS application calls the AUTHC service, ALCS checks that the user ID that originated the transaction has access to the corresponding profile in the ALCSAUTH class. AUTHC parameters specify the required level of access (READ, UPDATE, and so on).

If the ESM can not make a decision, your application program must decide what to do.

ICHRTX00 - MVS Router Exit with no external security manager

Installations with no external security manager (such as IBM's RACF) should code an MVS Router exit routine called ICHRTX00.

This installation exit routine is called from the MVS router when access control and other authorization related checking is required. Your new ALCS does this using the External Security Interface Macro - RACROUTE. The ICHRTX00 exit returns to the MVS router indicating whether further security checking is to be performed by an external security product.

If your installation has no external security product, IBM recommends that exit ICHRTX00 performs the following minimum functions:

- Interrogate the RACROUTE parameter list passed to the exit by the MVS router.
- Return the following return codes to the MVS router:
  - X'CC' (204) for RACROUTE AUTH requests. This is translated to a return code of 4 to ALCS, indicating that no decision on granting authority could be made. ALCS will decide the next action.
  - X'C8' (200) for all other RACROUTE requests. This is translated to a return code of 0 to ALCS, indicating that authority for the particular request is granted.

Your new ALCS requires these different return codes for two reasons. The first is to ensure that the security checking for CRAS assignment and transfer is comparable with previous ALCS versions. The second is to ensure that all end users can log-on to your ALCS system successfully.

Note: Your ICHRTX00 exit must also set RACF-compatible return and reason codes.

For further information regarding the exit ICHRTX00 see *MVS Installatiopn Exits* and *Security Server RACROUTE Macro Reference*.

MVS environment for ALCS

This section lists MVS configuration parameters that can affect ALCS installation and operation. For further details, see the appropriate versions of these publications:

- *MVS Initialization and Tuning Manual*
- *JES Initialization and Tuning Manual*

MVS configuration - required parameters

MVS Master Catalog

The ALCS product data sets must be cataloged in either a user catalog or the master catalog. IBM recommends that you define an alias for DXC (or your first level qualifier if you intend to change it) in the master catalog, and specify which user catalog you wish to use. For example:

```
DEFINE ALIAS ( NAME('DXC') RELATE('SYS1.UCIAL810') )
```

Parameters from IEAAPFxx

Libraries containing the ALCS monitor program load module (DXCMON) and the ALCS configuration load modules must be included in the MVS authorized program facility (APF) authorization list. Add the DXCLMD3 and DXCLMD4 libraries to either IEAAPFxx in SYS1.PARMLIB, or your APF authorization list, and activate this definition.

If you are running HLL applications, the library containing the LE run-time library routines must be included in the MVS APF authorization list.

If you are using the C/C++ remote debugger facility, the ALCS load module library containing modules DXC10OSM and DXCCSATP, and the Debug Tool load library, must be included in the MVS APF authorization list.

Current versions of z/OS allow dynamic updates of the APF authorization list.

MVS configuration - optional parameters

Parameters from COMMNDxx

This is the initial command list that the MVS operator specifies at MVS initialization time. It can contain a command to start ALCS. Note that VTAM initialization must be complete before ALCS can progress beyond standby state.

If you are using APPC/MVS, Communication Server IP, WebSphere for z/OS, or DB2 for z/OS, then you may want to include a command to start them.

Parameters from GRSRNLxx

This member contains resource name lists (RNLs) that determine how to process ENQ/RESERVE/DEQ requests in a global resource serialization (GRS) complex. See also the ALCS startup message number **DXC112R** in the *ALCS Operation and Maintenance*.

Use of this member applies when a standby ALCS and an active ALCS run on different systems in the same GRS complex.

By default, ALCS resources are globally serialized. If global resource serialization is **not** wanted, include the following in GRSRNLxx:

```
RNLDEF RNL(EXCL) TYPE(GENERIC) QNAME(DXCMON)
RNLDEF RNL(EXCL) TYPE(GENERIC) QNAME(DXCMON1)
```

Parameters from IEASYSxx

If your VFA buffers are above the bar and you specified AMODE64=LPSVFA in your ALCS SCTGEN, then you also must code the LFAREA parameter. See z/OS documentation for any details.

Parameters from SMFPRMxx

If your VFA buffers or online message trace area are above the bar, or you coded in your communication generation one or more TCP/IP connections with the IPMGSZ parameter, then you must specify a MEMLIMIT for ALCS in the PARMLIB SMFPRMxx member. An alternative to this is to specify MEMLIMIT on either a JOB or an EXEC job control statement.

Program properties table (PPT)

ALCS runs as a non-swappable address space. It is **not** necessary to add ALCS to the PPT to make it non-swappable. However, you may consider making ALCS non-cancellable.

Miscellaneous MVS considerations

Data Facility Hierarchical Storage Manager (DFHSM)

You must ensure that DFHSM does not migrate ALCS data sets during ALCS operation. This can happen if DFHSM is running on another system and GRS is not being used.

You may allow DFHSM to migrate ALCS output sequential files that are subsequently used as input general sequential files. If you do this, you must enable ALCS to wait for DFHSM to recall a migrated sequential file (see the description of the SCTGEN MIGSEQ parameter in "SCTGEN macro" on page 72.) However, ALCS cannot dynamically allocate any other data set while it is waiting for DFHSM to recall a migrated data set.

MVS generalized trace facility (GTF)

ALCS uses the MVS generalized trace facility (GTF) to provide diagnostic information for VTAM communication. For further information on GTF, see *ALCS Operation and Maintenance*. Two event identifiers (EIDs) are currently used:

EID - X'5FA'
The data traced is a VTAM RPL. The RPL has been created by ALCS to request that VTAM carries out some specified action. The data traced shows the RPL just before the request is passed to VTAM for execution.

EID - X'5FB'
The data traced is a VTAM RPL. The RPL has been completely processed by VTAM. The data traced shows the RPL immediately after ALCS has issued the VTAM CHECKRPL macro against that RPL.

The format and contents of the VTAM RPL can be obtained from the relevant *VTAM Programming* manual. There is no format appendage available to format the RPL GTF trace data if it is processed by the MVS interactive program control system (IPCS).

Access to DB2 for z/OS modules

If you are using DB2 for z/OS then put the DB2 for z/OS CAF modules in an accessible link list library. This avoids the need to include them in a STEPLIB or JOBLIB DD statement in the ALCS job.

Access to WebSphere MQ for z/OS modules

If you are using WebSphere MQ for z/OS then put the WebSphere MQ for z/OS load modules in an accessible link list library. This avoids the need to include them in a STEPLIB or JOBLIB DD statement in the ALCS job.

Access to WebSphere Application Server for z/OS optimized local adapters (OLA)

If you are using OLA then put the library which contains the OLA load modules in an accessible link list library. This avoids the need to include them in a STEPLIB or JOBLIB DD statement in the ALCS

job. You must either include the OLA load list library in the APF table or specify LNKAUTH=LNKLST in your PARMLIB IEASYSxx member so that all data sets in the LNKLST concatenation are treated as APF authorized.

Access to ICSF

If you are using the Integrated Cryptographic Service Facility (ICSF), make the library that contains the ICSF stub routines (hlq.SCSFMOD0) available to the DXCICSF link job. To do this, include the ICSF stub library in one of the following places:

- The MVS linklist
- The SYSLIB statement in the link job for DXCICSF.

Note: This library must be APF authorized.

Access to APPC/MVS modules

If you are using CPI-C or APPC/MVS then put the APPC/MVS service modules in an accessible link list library. This avoids the need to include them in a STEPLIB or JOBLIB DD statement in the ALCS job.

Access to TCP/IP

If you are using Communication Server IP, make the Communication Server IP link library (*hlq*.SEZALINK) available to the ALCS job or started task. To do this either:

- Include the Communication Server IP link library in a MVS LINKLIST, or
- Include the Communication Server IP link library in a STEPLIB or JOBLIB DD statement in the ALCS job.

Access to LE run-time library routines.

If you are running HLL applications, make the library containing the LE run-time library routines (*hlq*.SCEERUN) available to the ALCS job. To do this either:

- include the LE run-time library in the MVS linklist, or
- include the LE run-time library in a STEPLIB, or JOBLIB DD statement in the ALCS job, or
- include the LE run-time library in a DXCHLIB DD statement in the ALCS job.

Note that this library must be APF authorized.

Access to C/C++ remote debugger modules.

If you are using the C/C++ remote debugger, make the following libraries available to the ALCS job:

- ALCS load module library containing modules DXC10OSM and DXCCSATP
- Debug Tool load library (hlq.SEQAMOD)
- LE run-time library (hlq.SCEERUN).

To do this either:

- include all the libraries in the MVS linklist, or
- include all the libraries in a STEPLIB, or JOBLIB DD statement in the ALCS job, or
- include all the libraries in a DXCHLIB DD statement in the ALCS job.

Note that these libraries must be APF authorized.

MVS dataspaces

ALCS uses dataspaces for a number of purposes. These dataspaces are all pageable. If you use MVS storage isolation to limit paging in the ALCS region (and IBM recommends that you do this) you may need to increase the working set size for ALCS to allow for the extra storage within these dataspaces.

You need to check that MVS allocates enough entries in the primary address space and dispatchable unit access lists (PASN-ALs and DU-ALs). The number of access list entries that ALCS uses depends on what pool files you define for your installation. This is discussed in more detail below.

You also need to check if you have an MVS installation-wide exit that restricts the size of dataspaces. If you do then you may need to change this exit to satisfy ALCS requirements.

File-address lookup tables

ALCS uses one dataspace to hold the tables that it uses to locate short-term pool, fixed-file, and system-fixed file records on the real-time database.

This dataspace requires approximately 4*_n_* bytes, where *n* is the total number of fixed-file and short-term pool file records in your ALCS database plus an allowance for system fixed-file records and for expansion.

Short-term pool directories (type 2 short-term pool dispense)

When type 2 short-term pool dispense is active, ALCS uses one dataspace for each short-term pool that you define. For example, if you define L1, L2, L3, and L4 short-term pools then ALCS will use four dataspaces for the short-term pool directories.

Each dataspace requires approximately one byte for each record in the corresponding pool.

Remember to allow for the possibility of adding new short-term pools, and of adding more records to a short-term pool.

Allocatable pool directories (type 2 long-term pool dispense)

When type 2 long-term pool dispense is active, ALCS uses one dataspace for each allocatable pool that you define. While Recoup is running, ALCS uses **two** dataspaces for each allocatable pool that you define. For example, if you define L1, L2, L3, and L4 real-time database records then ALCS will use four dataspaces for the allocatable pool directories when Recoup is not running, and eight dataspaces when Recoup is running.

Each dataspace requires approximately one bit for each record in the corresponding allocatable pool - that is, for each real-time database record of that size.

Remember to allow for the possibility of adding new record sizes and of adding more records within any size.

DASD configuration table updates

ALCS uses one dataspace to hold the update items used to maintain your DASD configuration. This dataspace requires approximately 16 bytes for every `USRDTA`, `GFGEN`, `DBSPACE`, or `DBHIST` macro in your DASD generation.

Short-term pool event logging

While short-term pool event logging is active, ALCS uses one dataspace for each short-term pool which is being used. For example, if you define L1, L2, L3, and L4 short-term pools, and records are being accessed for each pool, then ALCS will use four dataspaces for the short-term pool event logging.

Each dataspace requires approximately 64 bytes for each record in the corresponding pool.

Remember to allow for the possibility of adding new short-term pools, and of adding more records to a short-term pool.

MVS log streams

About this task

The Emergency Pool Recovery (PDU) function in ALCS uses MVS system logger log streams to save long-term pool file addresses that have been released. Normally, long-term pool file is recovered by Recoup (see *ALCS Concepts and Facilities* for an overview of Recoup). If your available long-term pool file decreases to zero, and you have enabled PDU by defining the PDULOGSTREAM parameter on the ALCS SCTGEN generation macro, then ALCS will start dispensing the file addresses from the log stream.

An MVS log stream is a collection of data that is used as a log. There are two types of MVS log streams: coupling facility log streams and DASD-only log streams. You can use either type for PDU. The actual location of the log data in the log stream is transparent to ALCS.

In order to use the Emergency Pool Recovery function, you must define the log streams it uses to MVS. This includes the following steps:

Procedure

1. Define the sysplex coupling facility

   A coupling facility is not required if all the log streams are defined as DASD-only.

   See *MVS Setting Up a sysplex* for information on how to set up a z/OS system to run in a sysplex.
2. Define the sysplex LOGR policy

   See *MVS Assember Services Guide* for information on using the system logger services and the LOGR policy in a sysplex. The LOGR policy includes:

   - log stream definitions
   - coupling facility list structure assignments for the log streams, if applicable.
3. Define the log streams

   ALCS uses a separate log stream for each long-term pool file size. It constructs the log stream names as:

   ```
   logstream_prefix.Ls.i
   ```

   where:

   logstream_prefix
   Value of the PDULOGSTREAM parameter on the ALCS SCTGEN generation macro.

   L*s*
   Long-term pool file size.

   i
   ALCS system identifier.

   For example, if your system has three pool file sizes - L1, L2, and L3 - and you have generated ALCS with:

   ```
           ALCS ID=T,...
           SCTGEN PDULOGSTREAM=ALCSTEST,...
   ```

   then you would define three log streams to MVS:

   ALCSTEST.L1.T
   ALCSTEST.L2.T
   ALCSTEST.L3.T

   You can use the IXCMIAPU administrative data utility to define the PDU log streams to MVS. ALCS reads and writes data on the PDU log streams in blocks of 4KB. use this as the average and maximum

log block size. Continuing the example given above: if you decide to assign a single-coupling facility structure called ALCSTEST_PDU for the three log streams, you would set up LOGR policy information like this:

```
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *

  DATA TYPE(LOGR)

  DEFINE LOGSTREAM NAME(ALCSTEST.L1.T)
         STRUCTNAME(ALCSTEST_PDU)
         HLQ(IXGLOGR)

  DEFINE LOGSTREAM NAME(ALCSTEST.L2.T) LIKE(ALCSTEST.L1.T)

  DEFINE LOGSTREAM NAME(ALCSTEST.L3.T) LIKE(ALCSTEST.L1.T)

  DEFINE STRUCTURE NAME(ALCSTEST_PDU) LOGSNUM(3)
          MAXBUFSIZE(4096) AVGBUFSIZE(4096)
/*
```

Alternatively, you may decide to assign DASD-only log streams and therefore avoid the need to have a coupling facility installed. The LOGR policy information would then look like this:

```
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *

  DATA TYPE(LOGR)

  DEFINE LOGSTREAM NAME(ALCSTEST.L1.T)
         DASDONLY(YES)
         MAXBUFSIZE(4096)
         HLQ(IXGLOGR)

  DEFINE LOGSTREAM NAME(ALCSTEST.L2.T) LIKE(ALCSTEST.L1.T)

  DEFINE LOGSTREAM NAME(ALCSTEST.L3.T) LIKE(ALCSTEST.L1.T)
/*
```

Note: The HLQ parameter specifies the high-level qualifier that MVS uses for log stream data set names (IXGLOGR is the default).

4. Define the coupling facility list structure(s)

   For coupling facility log streams, the log streams must be assigned to coupling facility structures. Multiple log streams can share one structure, or each log stream can have a separate structure, depending on your installation preference.

   You define the coupling facility structures in the sysplex CFRM policy. Each log block for PDU contains approximately 1 020 file addresses. Use the ALCS long-term pool file release rates for your system together with information in *MVS Assember Services Guide* to calculate the size of the structures required for PDU.

Communication environment for ALCS

The VTAM supported network

Using VTAM and one or more channel-attached control units, ALCS supports any or all of:

- IBM 3270 Information Display System (including IBM 3192/3194) and IBM workstation devices
- Token Ring attached devices (including IBM 3192/3194) through a Token Ring gateway such as the IBM 3174
- IBM workstation devices attached directly to a Token Ring LAN

Using VTAM and one or more IBM 37xx controllers with the ACF/Network Control Program (ACF/NCP) and ACF/System Support Programs, ALCS supports any or all of:

- Remotely attached IBM 3270 (BSC or SDLC) and IBM workstation devices (3270 emulation mode).
- Token Ring attached devices (including IBM 3192/3194) and IBM workstation devices (3270 emulation mode) through a Token Ring gateway such as the IBM 3174.
- LAN attached devices (including IBM workstation devices in 3270 emulation mode) through a Token Ring or Ethernet LAN adapter such as the IBM 3174.
- WTTY communication lines in simplex, half-duplex, or full-duplex operation. This also requires Network Terminal Option (NTO).
- ALC communication lines. This also requires Network Extension Facility (NEF) or Airlines Line Control Interconnection (ALCI).
- X.25 (including AX.25) communication lines. This also requires NCP Packet Switching Interface (NPSI). AX.25 may be used to connect to the SITA network as described in ATA/IATA *Systems and Communications Reference Manual (Vols 1-7)*, and the appropriate SITA P3000 users' manual.
- Connection to one or more IMS, CICS, and ALCS using the intersystem communication (ISC) protocol (LU 6.1).
- Connection to one or more TPF and ALCS using the advanced program-to-program communications protocol (APPC).

Figure 6 on page 24 gives an overview of the VTAM supported network.

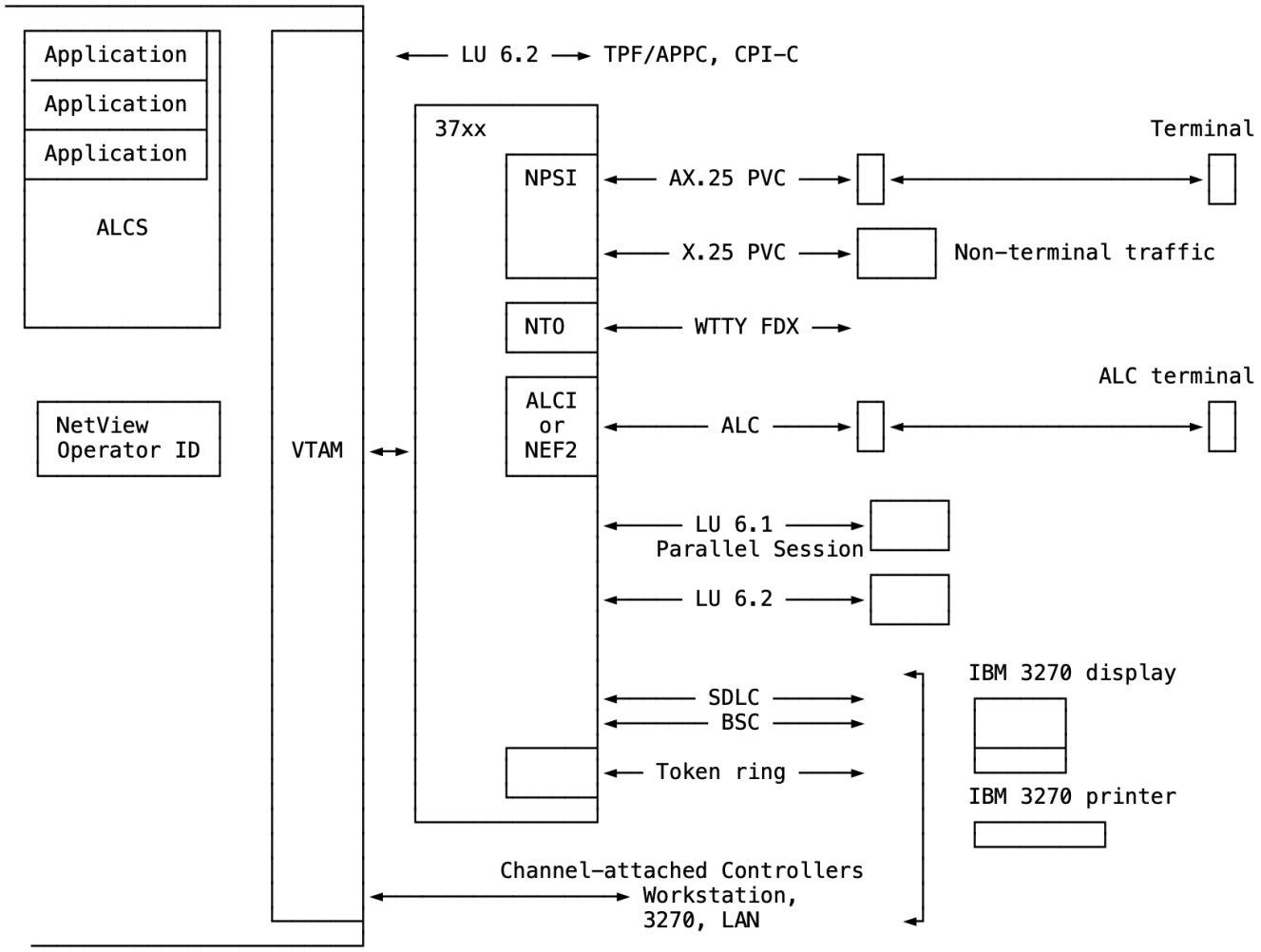


Figure 6. VTAM support

A communication management configuration (CMC) processor, typically running NetView, can be used to manage network resources.

In general, communication equipment in the VTAM network can be used to communicate with either ALCS or other applications, or both. However, in an operational environment, IBM recommends that at least two devices be available for use by ALCS while it is executing:

- A 3270 display station, NetView operator ID, or workstation in 3270 emulator mode (called Prime CRAS)
- a 3270 printer (called Receive Only CRAS or RO CRAS).

The TCP/IP network

Using Communication Server IP and one or more OSA cards and TCP/IP routers, ALCS supports any or all of:

- IBM workstation devices (3270 emulation mode) connected using TN3270 or TN370E.
- Connection to one or more remote applications using TCP or UDP sockets. This can includes HTTP connections to the ALCS Web Server.
- Connection to ALC terminals (or workstations in ALC emulation mode) through a MATIP gateway such as IBM's A2CS product.
- Connection to a high-level network (HLN) such as the SITA HLN for MATIP Type A conversational, Type A host-to-host, and Type B message traffic.

Figure 7 on page 25 gives an overview of the TCP/IP supported network.

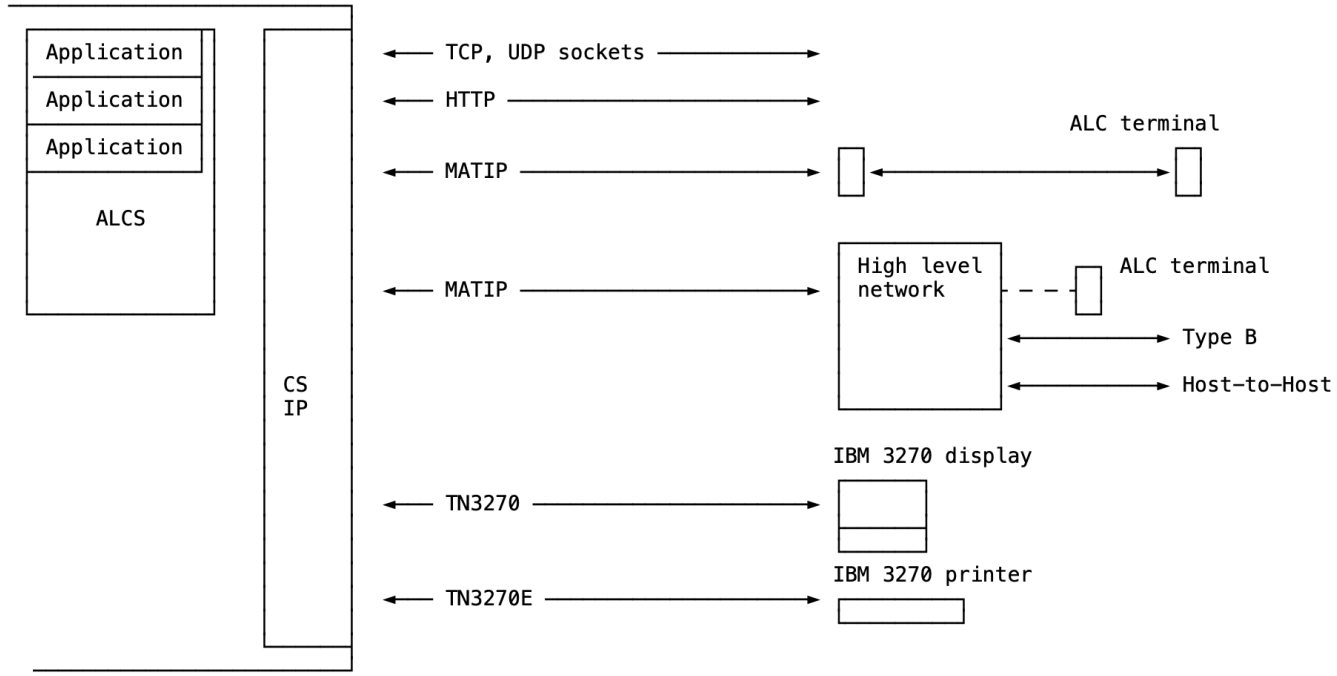


Figure 7. TCP/IP support

Defining the VTAM network for ALCS

Before deciding details of the VTAM network definition, consider the topics covered in this section. Read this section in conjunction with the pre-installation planning manual for the appropriate version of Communication Server SNA.

Sharing the communication network

By using VTAM, ALCS can share communication network facilities with other applications. When defining a shared network facility, consider the requirements of each of the application systems that can use the facility. Network facilities can be shared by:

- **One or more ALCS systems.** These can run in different regions in the same processor. Alternatively, using multisystem networking, they can run in separate processors.
- **One or more other application systems.** These can run in the same processor as the ALCS system(s), or they can run in other processors, possibly under different operating systems.

The communication network facilities can be shared in various ways, as follows:

- Application systems that run in the same processor can share the same copy of the VTAM routines.
- Using multisystem networking, several application systems and several processors can share one or more communication controllers.
- Terminals can access more than one application, logging on to each one as required. Thus, the same terminal can access both ALCS and other application systems.
- If your terminal is capable of supporting multiple host sessions (for example, an IBM workstation operating in distributed function terminal (DFT) mode), you can log on simultaneously to more than one application.
- Different terminals on the same terminal control unit and/or communication line can log on to different application systems at the same time. Thus, the same communication line can attach terminals used with ALCS and terminals used with other application systems. Note that SDLC terminals used with other applications, even if connected by the same communication line, can be device types that ALCS does not support.
- The IBM Token Ring network supports a wide variety of terminal types for use with ALCS. Some, such as the IBM workstation, are attached directly; others, such as the IBM 347x are attached through an IBM 3174 terminal control unit.

Performance of the communication network

Users of ALCS applications typically enter large numbers of messages, and require a rapid response time.

As a guide, to achieve the response times typically required by users of ALCS applications, specify:

- The number of terminals that connect to each terminal control unit
- The number of terminal control units that connect to each line

so that there is no more than a 50 per cent chance that a control unit or line is busy when a message is to be transmitted. Otherwise, transmission time increases because messages have to wait for the control unit or line to become available. Allow for recovery procedures used when communication errors occur.

Defining unformatted system services (USS) definition tables

See the programming and customization manuals for the appropriate version of VTAM for information on how to create USS definition tables.

Consult SITA for information on their special requirements if you are connecting to the SITA network (for example, an X.25 PVC, BATAP, and so on).

Defining logon mode table entries

This section describes logon mode tables for LUs that can log on to ALCS. Read it in conjunction with the programming and customization manuals for the appropriate version of VTAM.

The SNA protocols (session parameters) used between ALCS and communication resources are determined when the SNA sessions are established. VTAM uses information in a VTAM logon mode table to create a **BIND image**. A VTAM logon mode table contains one or more MODEENT statements. Each MODEENT statement defines one logon mode; that is, one set of session parameters. Appendix D,

"Sample logon mode table," on page 551 shows a sample logon mode table and presentation service (PSERVIC) parameter values for different LU types. ALCS does not issue its own BIND image, but always uses the one that VTAM supplies with the logon request. The only exception is an LU 6.1 parallel session with ALCS in send mode. For more information refer to the SESSION=SEND parameter of "COMDEF parameters for an LU 6.1 link" on page 125.

The maximum request unit (RU) size that ALCS accepts is 4000 bytes. If an RU is longer than this, ALCS discards the excess data. This means, for example, that the longest valid message from a display is just under 4000 characters.

The LDTYPE parameter of the COMDEF macro specifies the **logical device type** for ALCS communication resources. The LDTYPE parameter is explained in detail in "COMDEF macro" on page 107.

The following sections describe the parameters of the MODEENT statement for each resource type.

Display logon mode

This type of LU is defined in the sample MODEENT with LOGMODE=DXCLOGS*n*.

The following applies to all SNA display terminals (LDTYPE=VTAM3270):

- ALCS (primary):
  - Can send multiple element chains
  - Uses immediate request mode only
  - Can request definite or exception responses to chains
  - Does not use compression
  - Must send end bracket indicator
- Display (secondary):
  - Can send multiple element chains
  - Uses immediate request mode only
  - Can request exception response to chains
  - Does not use compression
  - Does not send end bracket indicator
- Common (ALCS and display):
  - Does not use function management headers
  - Uses brackets
  - Uses bracket termination rule 1
  - Uses EBCDIC only
  - Uses half-duplex flip-flop mode
  - ALCS performs any recovery action
  - Displays initiates brackets

LU presentation services profile 2

ALCS always looks for an 80 column presentation space size. The number of columns and rows that ALCS actually uses depends on the contents of the 1-byte presentation space size code field[2] of the BIND image as follows:

[2] This is the byte BINPRESZ in the field BINPSCHR in the VTAM ISTDBIND DSECT. Code the value in the PSERVIC parameter of MODEENT.

Table 5. ALCS presentation space size evaluation			
PSERVIC	**Evaluation**	**Rows**	**Cols**
--|--|--|--|00|--	Undefined, ALCS uses default values	24	80
--|--|--|--|01|--	ALCS does not support this.	12	40
--|--|--|--|02|--		24	80
--|--|--|--|03|--	Dynamic logon mode, ALCS queries the screen to determine the presentation space size.		
pr|*pc*|--|--|7E|--	Fixed primary size: X'*pr*' rows, X'*pc*' columns	*pr*	*pc*
pr|*pc*|*ar*|*ac*|7F|--	Primary size: X'*pr*' rows, X'*pc*' columns Alternate size: X'*ar*' rows, X'*ac*' columns Table 6 on page 28 shows how ALCS determines the presentation space size.		

Table 6. ALCS presentation space size evaluation (primary and alternate)						
pr	***pc***	***ar***	***ac***	**Evaluation**	**Rows**	**Cols**
>*ar*	80	*ar*	80	Both 80, use most rows	*pr*	80
<*ar*	80	*ar*	80	Both 80, use most rows	*ar*	80
pr	80	*ar*	*ac*	Use primary (primary is 80)	*pr*	80
pr	*pc*	*ar*	80	Use alternate (primary is not 80)	*ar*	80
pr	>*ac*	*ar*	*ac*	Neither are 80, use most columns	*pr*	*pc*
pr	<*ac*	*ar*	*ac*	Neither are 80, use most columns	*ar*	*ac*

Printer logon mode

This type of LU is defined in the sample MODEENT with LOGMODE=DXCLOGP*n*.

The following applies to all SNA printer terminals (LDTYPE=VTAM3270):

- ALCS (primary):
  - Can send multiple element chains
  - Uses immediate request mode only
  - Can request definite or exception responses to chains
  - Does not use compression
  - Must send end-bracket indicator
- Printer (secondary):
  - Can send multiple element chains
  - Uses immediate request mode only
  - Can request exception response to chains
  - Does not use compression
  - Does not send end-bracket indicator
- Common (ALCS and printer):

  - Does not use function management headers
  - Uses brackets
  - Uses bracket termination rule 1
  - Uses EBCDIC only
  - Uses half-duplex flip-flop mode
  - ALCS performs any recovery action
  - Printer initiates brackets

LU presentation services profile 1

ALCS treats any printer that uses this profile as an IBM 3287 Model 1. The following settings are mandatory:

- Function management (FM) header subset and data stream profile:
  - FM header subset 0 used; no FM headers allowed
  - Basic controls (data stream subset 0)
- Primary LU FM header subset flags must be zero.
- Primary LU data stream flags must indicate that a full base set data stream can be sent.
- Primary LU media flags must be zero.
- Secondary LU FM header subset flags must be zero.

ALCS determines the maximum output message size using the "primary logical unit maximum RU size" in the printer logon mode (MODEENT RUSIZES parameter). If this is set to zero, or omitted, then the default size of 65 535 bytes is used. You should ensure that the MODEENT RUSIZES parameter specifies a non-zero PLU maximum RU size which is not greater than the size of the printer's physical buffer.

LU presentation services profile 3

ALCS treats any printer that uses this profile as an IBM 3287 Model 1 or Model 2. It determines the model from the presentation size. If the presentation size is 1920 or greater, ALCS treats it as a Model 2 (presentation size 1920 bytes). Otherwise ALCS treats it as a Model 1 (presentation size 480 bytes).

ALCS determines the maximum output message size using the "primary logical unit maximum RU size" in the printer logon mode (MODEENT RUSIZES parameter). If this is set to zero, or omitted, then the default size of 65 535 bytes is used. Otherwise ALCS uses the smaller of the presentation size or the PLU maximum RU size. You should ensure that the MODEENT RUSIZES parameter specifies a non-zero PLU maximum RU size which is not greater than the size of the printer's physical buffer.

NEF or ALCI logon mode

This type of LU is defined in the sample MODEENT with LOGMODE=DXCLOGA.

For NEF or ALCI LUs (LDTYPE=VTAMALC), ALCS does not check the BIND image. The device type cannot be obtained from the BIND image; it is defined in the ALCS communication table generation (see “COMDEF macro” on page 107 for details). Appendix D, “Sample logon mode table,” on page 551 shows an example of the entry required for NEF or ALCI.

NTO WTTY logon mode

This type of LU is defined in the sample MODEENT with LOGMODE=DXCLOGW.

For WTTY links (LDTYPE=WTTY), ALCS does not check the BIND image. The device type cannot be obtained from the BIND image; it is defined in the ALCS communication table generation (see “COMDEF macro” on page 107 for details). Appendix D, “Sample logon mode table,” on page 551 shows an example of the entry required for NTO.

X.25 PVC logon mode

This type of LU is defined in the sample MODEENT with LOGMODE=DXCLOGX. ALCS does not check the BIND image. The device type cannot be obtained from the BIND image (it is defined in the ALCS communication table generation.

The following applies to all X.25 PVCs (LDTYPE=X25PVC):

- ALCS (primary):
  - Can send end-bracket indicator.
  - Maximum RU size is 3840 characters.
  - Requests exception response.
- NPSI (secondary):
  - Can send end-bracket indicator.
  - Maximum RU size is 3840 characters.
  - Requests exception response.

Application logon mode (LU 6.1)

This LU is defined in the sample MODEENT with LOGMODE=ALCSPARS.

Appendix D, “Sample logon mode table,” on page 551 shows a sample MODEENT entry for an ALCS application program node that uses VTAM parallel session support. This is required if ALCS uses LU 6.1 links (LDTYPE=ALCSLINK).

Other devices

If a device must be defined with LU presentation services profile 0, ALCS can support the device, provided all the following requirements are met:

- The device operates in the same way as a supported display or printer.
- The PSERVIC parameter of the MODEENT statement is entered as though the device is a supported printer or display, except that:
  - The first byte of the PSERVIC parameter (LU presentation services profile) is X'00'.
  - The last byte of the PSERVIC parameter contains X'01', X'02', or X'03', according to the type of support the device requires:

    X'01'
    ALCS supports the device like a printer terminal that uses presentation services profile 1.

    X'02'
    ALCS supports the device like a display terminal that uses presentation services profile 2.

    X'03'
    ALCS supports the device like a printer terminal that uses presentation services profile 3.
  - If the last byte of the PSERVIC parameter does not contain X'01', X'02', or X'03', ALCS ignores the rest of the PSERVIC parameter and treats the device according to its definition in the ALCS communication generation:

    If COMDEF TERM=3270DSP is specified, or TERM is omitted, ALCS treats the device as an IBM 3270 display with a presentation space size of 80 columns and 24 rows.

    If COMDEF TERM=3270PRT is specified, ALCS treats the device as an IBM 3287 Model 1 printer. The maximum output message size for the printer is 480 bytes.

Defining ALCS as a logical unit

This section describes how to define ALCS systems as application program nodes. Read this section in conjunction with the appropriate *VTAM Installation* manual.

Application program nodes are defined to VTAM in groups called **application program major nodes**. Each of these major nodes can contain one or more application program nodes. Define ALCS as an application program node. Group application program nodes into application major nodes to suit the operational requirements of your installation.

An APPL definition statement defines each application program node. Code one APPL definition statement for each ALCS system, using the following parameters:

ACBNAME
Specify the access method control block (ACB) name of the ALCS system to match the ACBNAME parameter of the ALCS COMGEN generation macro. Refer to "COMGEN macro" on page 98.

AUTH
Specify AUTH=ACQ so that ALCS can acquire sessions with logical units.

SRBEXIT
Specify SRBEXIT=YES. ALCS uses VTAM authorized path processing.

The LOGAPPL parameter of node definitions refers to the label on the APPL definition statement, not to the ACBNAME. To avoid confusion you can use the same symbol for both the ACBNAME and the label on the APPL definition statement.

Code the following additional parameters on the APPL definition statement for ALCS if the communication network is to include LU 6.1 links:

PARSESS
Specify PARSESS=YES so that ALCS can use VTAM parallel session support.

DLOGMOD
Specify the default logon with DLOGMOD=*logmode*, where *logmode* is a suitable entry in the VTAM logmode table for this ALCS application program node (see the LOGMODE parameter in Appendix D, "Sample logon mode table," on page 551).

The following example defines an ACB name for ALCS to VTAM:

```
PYEZ1WWW APPL  AUTH=(ACQ),ACBNAME=PYEZ1WWW,SRBEXIT=YES,              X
               PARSESS=YES,DLOGMOD=ALCSPARS
```

Defining local major nodes

This section describes how to define local major nodes that include terminals (LUs) that can log on to ALCS. Read this section in conjunction with the installation manual for the appropriate version of VTAM. See also "The VTAM supported network" on page 23. Note that this section describes only the aspects that are ALCS requirements.

VTAM definition statements define each terminal as part of one or more logical groups of local terminals. Each group is called a **major node**. There are separate major nodes for SNA and non-SNA terminals.

Each non-SNA terminal node (LU) is defined by a LOCAL definition statement. Each SNA terminal node is defined by physical unit (PU) and LU definition statements.

Remember that in local major nodes, ALCS supports only IBM 3270 Information Display System and compatible terminals.

The name on the LU or TERMINAL statement is the name that ALCS uses as the CRN of the terminal. Specify this name in ALCS communication generation in the NAME parameter of the COMDEF macro; see "COMDEF macro" on page 107.

For terminals that communicate only, or mainly, with ALCS, it is normally convenient to specify a default logon mode that is suitable for use with ALCS. To do this, specify the DLOGMOD and MODETAB parameters as described in Appendix D, "Sample logon mode table," on page 551.

To log the terminal automatically on to ALCS when it becomes active, specify the LOGAPPL parameter; see "Defining ALCS as a logical unit" on page 30.

Note: Do not specify the LOGAPPL parameter for ALCS CRASs.

Defining NCP major nodes and generating NCP

This section describes how to define VTAM NCP major nodes that include terminals (LUs) that can log on to ALCS. Read this section in conjunction with the installation manuals for the appropriate versions of VTAM, ACF/NCP, and NTO. See also "The VTAM supported network" on page 23. Note that this section describes only aspects of the definition that are directly related to ALCS requirements.

Normally, the same set of statements defines the NCP major node to VTAM and also generates ACF/NCP.

Defining BSC components

Use the GROUP or LINE statements to specify the characteristics of binary synchronous communication (BSC) lines. Use the CLUSTER and TERMINAL statements to specify the characteristics of BSC IBM 3270 terminal control units and terminals. When defining lines that attach terminals used with ALCS, the following values are recommended:

GROUP statement operands (BSC)

ALCS has no requirements for GROUP statement operands.

LINE statement operands (BSC)

CODE
Specify CODE=EBCDIC, or omit this parameter.

POLLED
Specify POLLED=YES.

RETRIES
This parameter controls the error recovery procedures that ACF/NCP uses when there are transmission errors. Specify RETRIES=NONE to avoid performance degradation.

CLUSTER statement operands (BSC)

BHSET
Specify BHSET=NONE, or omit this parameter.

CDATA
Specify CDATA=NO, or omit this parameter.

GPOLL
General polling must be used. Specify the general polling characters assigned to the terminal control unit.

TERMINAL statement operands (BSC)

Remember that the only BSC terminals that ALCS supports are IBM 3270 Information Display System and compatible terminals.

The name on the TERMINAL statement is the name that ALCS uses as the CRN of the terminal. Specify this name in ALCS communication generation in the NAME parameter of the COMDEF macro; see "COMDEF parameters for a VTAM 3270 terminal" on page 153.

ADDR
Specify the selection characters assigned to the terminal.

BFRDLAY
Specify BFRDLAY=1 for IBM 3284 and 3286 and compatible printers. ALCS waits for each message block to print before it transmits the next block; consequently no delay in ACF/NCP is required (BFRDLAY=1 is the minimum valid specification).

BHSET
Specify BHSET=NONE, or omit this parameter.

CDATA
Specify CDATA=NO, or omit this parameter.

DIRECTN
Specify DIRECTN=INOUT for displays, DIRECTN=OUT for printers.

For terminals that communicate only, or mainly, with ALCS specify a default logon mode that is suitable for use with ALCS. To do this, specify the DLOGMOD and MODETAB parameters as described in Appendix D, "Sample logon mode table," on page 551.

To log the terminal automatically on to ALCS when it becomes active, specify the LOGAPPL parameter; see "Defining ALCS as a logical unit" on page 30.

Note: Do not specify the LOGAPPL parameter for ALCS CRASs.

Defining SDLC components

Use the GROUP or LINE statements to define the characteristics of Synchronous Data Link Control (SDLC) lines.

Use the PU and LU statements to define the characteristics of SDLC IBM 3270 terminal control units and terminals. When defining lines that attach terminals used with ALCS, the following values are recommended:

GROUP statement operands (SDLC)

ALCS has no requirements for GROUP statement operands.

LINE statement operands (SDLC)

POLLED
Specify POLLED=YES.

RETRIES
This parameter controls the error recovery procedures that ACF/NCP uses when there are transmission errors. Specify RETRIES=NONE to avoid performance degradation.

PU statement operands (SDLC)

ALCS has no requirements for PU statement operands.

LU statement operands (SDLC)

The name on the LU statement is the name that ALCS uses to reference the terminal. Specify this name in ALCS communication generation in the NAME parameter of the COMDEF macro; see "COMDEF parameters for a VTAM 3270 terminal" on page 153.

Remember that the only SDLC terminals that ALCS supports are IBM 3270 Information Display System and compatible terminals.

For terminals that communicate only, or mainly, with ALCS, specify a default logon mode that is suitable for use with ALCS. To do this, specify the DLOGMOD and MODETAB parameters as described in Appendix D, "Sample logon mode table," on page 551.

To log the terminal automatically on to ALCS when it becomes active, specify the LOGAPPL parameter; see "Defining ALCS as a logical unit" on page 30.

Note: Do not specify the LOGAPPL parameter for ALCS CRASs.

Defining WTTY lines

ALCS supports WTTY lines using NTO. Define WTTY devices twice in the ACF/NCP major node definition: once as real WTTY devices[3], and once as virtual SNA representations. Define the real WTTY devices before the virtual SNA representations in the ACF/NCP generation.

Include the devices in the NTO generation, as described in “Generating NTO” on page 36.

Define all WTTY lines, in the VTAM major node definition and the ACF/NCP generation, as half-duplex. Specify `DIRECTN=INOUT` for the terminals, regardless of the actual characteristics of the communication equipment. This applies even if the line is used for processor-to-processor connection (when there are no actual terminals attached to the line).

Define each ALCS full-duplex WTTY line as two separate WTTY lines in the VTAM major node definition and the ACF/NCP generation. That is, two real WTTY lines (each with one real WTTY device) and the corresponding virtual SNA representations.

Defining WTTY real devices

Use the GROUP, LINE, and TERMINAL statements to define lines that attach terminals used with ALCS. The following values are recommended:

GROUP statement operands (real WTTY)

CRETRY
Specify `CRETRY=0`.

DIAL
Specify `DIAL=NO`.

ISTATUS
Specify `ISTATUS=INACTIVE` (NTO requirement).

LNCTL
Specify `LNCTL=SS`.

PADCNT
Specify `PADCNT=0`, or omit this parameter if the terminals are equipped with continuously running motors; otherwise specify `PADCNT=10`.

TEXTTO
Specify TEXTTO=28, to give a 28 second (nominal) text timeout. This allows 28 seconds between entry of consecutive characters at a WTTY terminal.

If there is a pause longer than 28 seconds following the entry of a character at a WTTY terminal, ACF/NCP assumes that no more characters are going to be entered. ACF/NCP passes the characters entered so far to ALCS and IPARS message switching processes them as an "abruptly terminated message"; the WTTY line then becomes available for ALCS to transmit messages.

TYPE
Specify `TYPE=NCP`.

WTTYEOB
Specify this parameter in International Telegraph Alphabet number 2 (ITA2) Parallel Data Field (PDF) code. The ITA2 codes, including ITA2 PDF code, are listed in the *ACF/NCF/VS Program Reference Summary*. For IPARS Message Switching use `FIGS V`; that is, specify `WTTYEOB=1B1E`.

WTTYEOT
Specify this parameter in ITA2 PDF code. The ITA2 codes, including ITA2 PDF code, are listed in the *ACF/NCF/VS Program Reference Summary*. IPARS Message Switching supports a number of end-of-transmission sequences; the most commonly used are `NNNN` and `FIGS H LTRS`.

[3] In this section, the expression "real WTTY device" is used in the same sense as in the *Network Terminal Option Installation Manual*. That is, it refers to the WTTY terminal that must be defined in the VTAM major node definition and the ACF/NCP generation, whether or not any actual terminal exists.

- Specify WTTYEOT=0C0C0C0C for NNNN.
- Specify WTTYEOT=1B141F for FIGS H LTRS.

LINE statement operands (real WTTY)

CODE
Specify CODE=ITA2.

CRRATE
Specify CRRATE=69 to prevent transmission of unnecessary idle characters by ACF/NCP.

CUTOFF
Specify CUTOFF=NO, or omit this parameter.

DUPLEX
Specify DUPLEX=HALF.

FEATURE
Specify FEATURE=(IMEND) to disable the line quiet test.

FGSLTRS
Specify FGSLTRS=OUT, or omit this parameter.

LINESIZ
Specify LINESIZ=69.

MONITOR
Specify MONITOR=YES.

POLLED
Specify POLLED=NO.

SPSHIFT
Specify SPSHIFT=NO, or omit this parameter.

TRANSFR
This parameter, in conjunction with the BFRS parameter of the BUILD statement, controls the way ACF/NCP segments WTTY input messages. Code TRANSFR so that TRANSFR multiplied by BFRS is more than 3840 but less than MAXDATA.

ALCS does not use the name specified on this LINE statement. ALCS uses the name specified on the virtual SNA representation LU statement to refer to WTTY lines.

TERMINAL statement operands (real WTTY)

LGRAPHS
Specify LGRAPHS=(REJECT,REJECT).

TERM
Specify TERM=WTTY.

BHSET
Specify BHSET=NONE, or omit this parameter.

CDATA
Specify CDATA=NO, or omit this parameter.

ENDTRNS
Specify ENDTRNS=EOT, or omit this parameter.

DIRECTN
Specify DIRECTN=INOUT, regardless of the actual characteristics.

FEATURE
Specify FEATURE=(NOATTN,NOBREAK), or omit this parameter.

INHIBIT
Specify INHIBIT=(TIMEFILL,ERPR,ERPW).

ALCS does not use the name specified on this TERMINAL statement. ALCS uses the name specified on the virtual SNA representation LU statement to refer to WTTY lines.

Defining virtual SNA representations for WTTY devices

Use the GROUP, LINE, PU, and LU statements to define the characteristics of WTTY devices that are used for communication with WTTY lines. See the *Network Terminal Option Installation Manual* for details. ALCS does not have any special requirements.

Define each ALCS full-duplex WTTY line with two separate virtual SNA representations: one for the send side and one for the receive side. That is, define two real WTTY lines (each with one real WTTY terminal) and corresponding virtual SNA representations.

ALCS uses the names on the virtual SNA representation LU statements as the CRNs of WTTY lines. Specify them in ALCS communication generation in the NAME parameter of COMDEF macros; see "COMDEF parameters for a WTTY link" on page 158.

Defining LU 6.1 links

See "Application logon mode (LU 6.1)" on page 30 and "Defining ALCS as a logical unit" on page 30 for information about defining ALCS as a logical unit to VTAM when the communication network includes LU 6.1 links.

See "COMDEF parameters for an LU 6.1 link" on page 125 for information about defining the link to ALCS.

Appendix E, "LU 6.1 Communication generation," on page 555 contains sample data to generate an:

- LU 6.1 link between ALCS and IMS/VS
- LU 6.1 link between ALCS and CICS/VS.

Generating NTO

The *Network Terminal Option Installation Manual* gives information about how to generate NTO. ALCS does not have any special requirements. Define each ALCS full-duplex WTTY line with two separate virtual SNA representations: one for the send side and one for the receive side. That is, define two real WTTY lines (each with one real WTTY terminal) and corresponding virtual SNA representations.

Defining NEF and ALCI resources

For detailed information about:

- Generating NEF or ALCI
  - See the appropriate *Installation and Operations* manual.
  - See Appendix G, "ALCI sample definition," on page 573
- Defining NEF or ALCI resources to ALCS
  - "COMDEF parameters for a NEF or ALCI ALC terminal" on page 152
  - "COMDEF parameters for a NEF or ALCI LU" on page 151

Defining X.25 resources

The *X.25 NCP Packet Switching Interface Initialization and Operation Manual* gives information about how to generate NPSI. When defining X.25 permanent virtual circuits (PVCs) that connect ALC terminals and WTTY communication lines to ALCS through an AX.25 high-level network, follow the example in Appendix F, "Network Control Program sample definition," on page 557.

Note that ALCS requires only the specified MAXDATA value. Other specified values are required by the high-level network owner, for example, SITA.

Defining APPC for ALCS

APPC/MVS provides a set of callable services that enable MVS application programs to communicate with other application programs through communication protocols provided by the SNA network.

APPC/MVS exists in a startable address space. When the START APPC operator command is issued (automatically at IPL is the preferred method) the APPC component receives control and begins initialization in its own address space. This initialization includes processing PARMLIB information from the APPCPMxx PARMLIB member, establishing the address space as a service address space, and establishing itself as an APPC/VTAM application. To APPC/MVS, ALCS appears as a transaction scheduler which does not create subordinate address spaces. APPC statistical information is written to SMF (Record type 30).

The values in the APPC component PARMLIB members are used to establish the environment in which APPC/MVS services are provided, and define the names of the APPC/MVS LUs. The LUADD statement in the APPCPMxx PARMLIB member, contains the keyword SCHED, designating which transaction scheduler (in our case ALCS) will use the LU name, and the keyword TPDATA, specifying the data set which holds the TP profiles. Note that SCHED must be coded as ALCS*x*000, where *x* is the ALCS system identifier as defined during ALCS generation.

The APPC administration utility (ATBSDFMU) provides services in the form of commands that create and maintain TP profiles through a batch job. The TPSCHED_EXIT keyword specifies the ALCS routine DXCPROF, to check input that is specific for the scheduler. DXCPROF must be APF-authorized. For an inbound allocate request, APPC/MVS passes information from the transaction scheduler section of the TP profile to ALCS's XCF message user routine. This routine analyzes the information in order to determine which ALCS application or APPC connection to activate. The format of the transaction scheduler information in the TP profile must be one of:

- APPL=*crn* (for compatibility with previous versions of ALCS)
- CRN=*crn*
- CRN=*

You must define *crn* in the ALCS communication generation as the CRN of one of:

- ALCS application
- APPC connection.

If the CRN is an ALCS application then ALCS creates a new entry and passes control to the input edit program for the application; the application manages the APPC conversation.

If the CRN is an APPC connection then ALCS itself manages the APPC conversation.

If you specify CRN=* then ALCS scans all the APPC connections defined in the communication generation until it finds one whose partner LU name matches the partner LU name for this inbound allocate request. See "COMDEF parameters for APPC connections" on page 127 for details of how to define an APPC connection. The COMDEF PLUNAME and PLUNAME2 parameters specify the partner LU names. To find a matching partner LU name for an inbound allocate request, ALCS checks the PLUNAME2 parameter (if it is specified) otherwise it checks the PLUNAME parameter. Specify CRN=* when inbound allocate requests from different partners use the same ALCS TP name.

Figure 8 on page 37, Figure 9 on page 38, and Figure 10 on page 38 show examples of how to set up APPC.

Following are examples of APPC/MVS LU definitions:

```
LUADD ACBNAME(alcs_lu1) TPDATA(SYS1.TPDB) BASE SCHED(ALCSM000)
LUADD ACBNAME(alcs_lu2) TPDATA(SYS1.TPDB) SCHED(ALCSM000)
```

Figure 8. Example of LUADD

Following are examples of APPC/MVS TP profile definitions:

```
TPADD TPSCHED_EXIT(DXCPROF)
  TPNAME(alcs_tp1)
  SYSTEM
  ACTIVE(YES)
  TPSCHED_DELIMITER(##)
    CRN=appl1
  ##

TPADD TPSCHED_EXIT(DXCPROF)
  TPNAME(alcs_tp2)
  SYSTEM
  ACTIVE(YES)
  TPSCHED_DELIMITER(##)
    CRN=appc1
  ##

TPADD TPSCHED_EXIT(DXCPROF)
  TPNAME(alcs_tp3)
  SYSTEM
  ACTIVE(YES)
  TPSCHED_DELIMITER(##)
    CRN=*
  ##
```

Figure 9. Example of TPADD

Following are examples of ALCS communication definitions:

```
COMDEF LDTYPE=ALCSAPPL,                                    -
      NAME=appl1,                                          -
      PROG=prog1

COMDEF LDTYPE=APPC,PRTCOL=TYPE1,                           -
      NAME=appc1,                                          -
      APPL=appl2,                                          -
      TPNAME=partner_tp,                                   -
      PLUNAME=partner_lu1

COMDEF LDTYPE=ALCSAPPL,                                    -
      NAME=appl2,                                          -
      PROG=prog2
```

Figure 10. Example of ALCS communication generation

Storage requirements for APPC

If APPC support is specified in ALCS, APPC requires some storage from the system-queue area (SQA) which is calculated using the formula:

```
SQA storage = 1000 + ( num_srbs SRBs specified in SCTGEN × 56) bytes
```

For additional information, refer to the following manuals:

MVS Initialization and Tuning Manual
MVS System Messages and Codes: Volumes 1 through 3
MVS Programming: Writing Transaction Programs for APPC/MVS

Defining TCP/IP for ALCS

Before deciding details of the TCP/IP network definition, consider the topics covered in this section. Read this section in conjunction with the pre-installation planning manual for the appropriate version of Communication Server IP.

Customizing Communication Server IP

- The *hlq*.PROFILE.TCPIP data set.

  No special changes are required for ALCS. You can optionally reserve a port number for the ALCS concurrent server (Listener) or other ALCS TCP/IP communication resources that behave as servers. Use one or more PORT statements to do this. The format of the PORT statement is:

  ```
  PORT port_num TCP user
  ```

  Where:

  port_num
  Reserved port number.

  user
  The user ID, procedure name, or job name that can use this port, one of:

  – Name of the member in the cataloged procedure library that is used to start ALCS
  – User ID of the ALCS batch job
  – Name of the ALCS batch job.

  For example,

  ```
  PORT 4001 TCP ALCSTEST
  ```

- The *hlq*.TCPDATA.TCPIP data set.

  No special changes are required for ALCS. Check the TCPIPJOBNAME statement - this specifies the member name of the procedure used to start the Communication Server IP address space. This is also the name that you must specify on the TCPNAME parameter of the ALCS SCTGEN macro, or on the NAME parameter of the ALCS ZCTCP command. The format of the TCPIPJOBNAME statement is:

  ```
  TCPIPJOBNAME tcpip_proc
  ```

  Where:

  tcpip_proc
  The name of the member in the cataloged procedure library that is used to start the TCP/IP address space.

  For example,

  ```
  TCPIPJOBNAME TCPIP
  ```

TELNET for IBM 3270 terminals

About this task

This is a standard service provided by Communication Server in z/OS. Telnet sessions using TN3270 or TN3270E appear as standard 3270 devices in ALCS. If you have a TN3270E Telnet client, you can emulate 3270 printers. Telnet does not require any configuration of the ALCS TCP/IP support. The following are the steps to configure Telnet in Communication Server and ALCS:

Procedure

1. VTAM - Define the following APPLs and logmode table

   ```
   TCP32701 APPL  AUTH=NVPACE,EAS=1,PARSESS=NO,SESSLIM=YES,
   ```

```
                MODETAB=LOGIASC,DLOGMOD=PYE32782
TCP32702 APPL   AUTH=NVPACE,EAS=1,PARSESS=NO,SESSLIM=YES,
                MODETAB=LOGIASC,DLOGMOD=PYE32782
```

Logmode table:

```
LOGIASC  MODETAB
*  3278 MODEL 2
         MODEENT LOGMODE=PYE32782,
                 FMPROF=X'03',
                 TSPROF=X'03',
                 PRIPROT=X'B1',
                 SECPROT=X'90',
                 COMPROT=X'3080',
                 RUSIZES=X'87F8',
                 PSERVIC=X'020000000000185000007E00'
                 MODEEND
```

2. ALCS - communication resources

```
         COMDFLT CLEAR,LDTYPE=VTAM3270,APPL=RES0,UPDATE=ADD
         COMDEF NAME=TCP32701,CRAS=AT51
         COMDEF NAME=TCP32702,CRAS=AT52
```

3. TCP/IP - PROFILE configuration

```
BEGINVTAM
;
TELNETDEVICE  3278-2   PYE32782
TELNETDEVICE  DYNAMIC  ,PYE3287
;
  LUGROUP ALCSLU TCP32701..TCP32710 ENDLUGROUP
  DEFAULTLUS TCP00001..TCP00030  ENDDEFAULTLUS
;
  ALLOWAPPL PYEZ1* DISCONNECTABLE    ;ALCS Application name
     LU TCP32701
     LU TCP32702
 ENDVTAM
```

ALCS controlled sockets - server service

ALCS controlled sockets means that ECB programs can receive and transmit data to TCP/IP socket clients (applications that have connected to the ALCS Server service) without having to use any TCP/IP socket calls inside the ECB programs. ALCS takes care of accepting the sockets, maintaining the connection and receiving or sending data on your behalf.

Your must perform the following tasks in order to enable ALCS ECB application programs to process data received from TCP/IP and to use the macros ROUTC and RC0PL to send data back to the originator:

- Define the use of TCP/IP in the ALCS system table.

  Specify TCPIP=YES and optionally TCPNAME=*tcpip_address_space_name* on the SCTGEN macro.
- Define the following TCP/IP communication resources in the ALCS communications table (COMDEF macro):

  1. An ALCS application which will receive input messages from the socket connections (LDTYPE=ALCSAPPL)
  2. A TCP/IP server (LDTYPE=TCPIP , TERM=SERVER).
- Agree with the client application the format of the input and output messages.
- Write an input message processor to receive input messages from a socket connection.
- Code the USRTCP2 ALCS installation-wide monitor exit to handle message fragmentation, or message concatenation, by TCP/IP.
- Use the COMIC macro in your application program to determine the connection status, originator's IP address and other characteristics of the TCP/IP connection.

- Use the COMCC macro in your application program to request termination of the socket connection after a response message has been sent.
- Use the ASCIC macro in your application program to translate ASCII to/from EBCDIC if required.
- Use the ROUTC and RC0PL macros in your application program to send data to the socket connection.
- Use the ZACOM command to start and stop the TCP/IP server service, or use COMDEF ISTATUS=ACTIVE to start the service automatically.

ALCS Web Server

The ALCS Web Server is an example of a server service that uses ALCS controlled sockets. The ALCS programs that implement this service are CWW1 to CWW6. The ALCS Web Server uses the ALCS hierarchical file system (HFS). The ALCS HFS is implemented by the programs CHFC, CHFR and CHFS. The ALCS Web Server does not need the USRTCP2 exit since ALCS handles the HTTP messages.

The following ALCS communication definitions are required for the ALCS Web Server:

```
         COMGEN RCVSZE=MAX, ...         For PC file transfer buffers
         COMDFLT CLEAR
         COMDFLT LDTYPE=TCPIP,ICRI=100020
*        This COMDEF will generate 16 new CRIs 100020 - 10002F
         COMDEF NAME=ALCSWEB1,TERM=(SERVER,HTTP),APPL=WEB1,
                        LPORT=80,MAXCONN=15,TIMEOUT=120
*        Program CWW1 will receive all input from the web connections
         COMDFLT CLEAR
         COMDFLT LDTYPE=ALCSAPPL,ISTATUS=ACTIVE,FMSG=YES
         COMDEF NAME=WEB1,PROG=CWW1
         COMDEF NAME=HFS1,PROG=CHFP
```

ALCS controlled sockets - client service

This service is used by ECB-controlled programs that need to send data to an application server using TCP/IP sockets.

Define the TCP/IP resource in the ALCS communication table as a sockets client. For example:

```
         COMDFLT CLEAR
         COMDFLT LDTYPE=TCPIP,ICRI=100011
*        This COMDEF will generate 1 new CRI 100011
         COMDEF NAME=ALCSCLN1,TERM=CLIENT,APPL=CLNT,
               RHOST=9.188.881.12,RPORT=7798
*        Program KLN1 will receive all input from the socket connections
         COMDFLT CLEAR
         COMDFLT LDTYPE=ALCSAPPL,ISTATUS=ACTIVE
         COMDEF NAME=CLNT,PROG=KLN1
```

When you activate this client (see below), it establishes a connection with the server at the specified remote IP address (RHOST) and port number (RPORT).

Application programs use ROUTC and RC0PL macros to send data to the remote server application.

All input will be received by the input application (APPL).

The input program for the application (KLN1 in the above example) and the output generator program use the same macros explained in “ALCS controlled sockets - server service” on page 40.

Typically, a client application initiates the socket connection, sends the first message, and terminates the socket when it is not longer required.

There are two ways to activate the client connection:

- Specify COMDEF ISTATUS=ACTIVE in the ALCS communication definition
- Use the ZACOM ACT command.

When the client connection is activated, ALCS creates an ECB and gives control to ECB-controlled exit program ACE1. ACE1 is passed an indicator that a TCP/IP connection has been established. ACE1 should

identify that the correct TCP/IP connection has been activated before passing control to the application program that is going to initiate the conversation with the remote server application.

Since the ALCS controlled client connection uses stream sockets, you must code the USRTCP2 installation-wide monitor exit in order to handle message fragmentation or message concatenation in the TCP/IP network. You must also agree on a message format with the remote host that will enable USRTCP2 to reconstruct the original message; for example by including the length of each message in the first few bytes.

ALC terminal support using TCP/IP

The support of ALC (P1024B) terminals connected to ALCS using the TCP/IP protocol is an extension of the ALCS controlled sockets interface.

Define these terminals using `COMDEF  LDTYPE=TCPIPALC`. Each terminal must be associated with a TCP/IP client or server communication resource. (This is similar to the way AX25ALC devices are associated to an AX25 PVC.)

There are three methods for connecting ALC terminals (or emulations) using TCP/IP:

- Using IBM Advanced Communication System Access (ACSA).

  You can connect workstations running ACSA (in ALC terminal emulator mode) to ALCS using TCP/IP stream socket connections. Each workstation establishes a socket connection with ALCS. There is no need to code the installation-wide monitor exit USRTCP2 for ACSA support, ALCS is handling all the ACSA messages internally.
- Using Mapping of Airline Traffic over TCP/IP (MATIP).

  ALCS supports the MATIP protocol for messages between ALC terminals and host computer systems (Type A traffic). This can include workstations running IBM ACSA/A2CS over your private TCP/IP network. Many terminals can be accessed through each TCP/IP socket connection.

  If the terminals are connected through a high-level network such as SITA then some of the ALCS communication definition data (HEX, TCID, IA and TA values) are provided by the owner of the high-level network. If the terminals are connected through ACSA/A2CS then these values are configured in ACSA and A2CS.
- User specific support.

  ALCS can support other ALC terminal emulations over TCP/IP, called **user terminal emulations** because the terminal support is coded in ALCS installation-wide monitor exits.

  Typically, these user terminal emulations are used to support application-to-application communication using a technique called **screen scraping**. With this technique, a remote application using TCP/IP sockets can send ALCS application input messages and receive responses in the same format as they are in any display terminal. The remote application has to scan the output message to extract the data (this is screen scraping). User terminal emulation is a way of passing data to other systems outside ALCS without any modifications to the applications that run under ALCS. The disadvantage is that the remote application depends on the format of the responses produced by the ALCS applications and the knowledge of the exact sequence and format of the input messages.

Type B message support using TCP/IP

ALCS supports the MATIP protocol for messages between Message Switching systems (Type B traffic). The IPARS Message Switching function is able to use TCP/IP connections, in conjunction with the ALCS BATAP routines, to transmit and receive Type B messages. Define a TCP/IP client resource in the ALCS communication table with the IP address, port number, and high-level designators of the remote host system. Also define a virtual SLC link for Message Switching if required. For example:

```
**       Define Type B client
         COMDFLT CLEAR,LDTYPE=TCPIP,ISTATUS=INACTIVE
         COMDEF NAME=ALCSCLNB,TERM=(CLIENT,MATIPB),                    -
               APPL=MESW,HEX=3344,HEN=5566,                            -
```

```
               RPORT=7700,                                              -
               RHOST=9.188.100.13
**       Define Virtual SLC link
         COMDFLT CLEAR,LDTYPE=SLCLINK,ISTATUS=INACTIVE
         COMDEF NAME=VSLC,LINK=ALCSCLNB
**       Define application
         COMDFLT CLEAR,LDTYPE=ALCSAPPL,ISTATUS=ACTIVE
         COMDEF NAME=MESW,PROG=XIMD
```

Either ALCS or the remote system can initiate the TCP/IP connection.

- ALCS can activate the client connection in two ways:
  - Specify COMDEF ISTATUS=ACTIVE in the ALCS communication definition.
  - Use the ZACOM ACT command.

  You can specify up to four values on the RHOST parameter. ALCS tries to connect using each address each in turn; this may be useful if the remote system uses primary and alternate IP addresses for the service.
- Alternatively, an application on the remote system can start the socket connection. To allow this, you must define a TCP/IP server resource as well as the client resource, in order to monitor for the inbound connection requests. The remote host connects to the port number defined on the COMDEF LPORT parameter for the TCP/IP server resource. For example:

```
**       Define Type B client and server
         COMDFLT CLEAR,LDTYPE=TCPIP,ISTATUS=INACTIVE
         COMDEF NAME=ALCSSERB,TERM=(SERVER,MATIPB),                    -
               APPL=SERB,MAXCONN=2,LPORT=351
         COMDEF NAME=ALCSCLNB,TERM=(CLIENT,MATIPB),                    -
               APPL=MESW,HEX=3344,HEN=5566,                            -
               RPORT=7700,                                             -
               RHOST=(9.188.100.13,9.188,100,14)
**       Define Virtual SLC link
         COMDFLT CLEAR,LDTYPE=SLCLINK,ISTATUS=INACTIVE
         COMDEF NAME=VSLC,LINK=ALCSCLNB
**       Define application
         COMDFLT CLEAR,LDTYPE=ALCSAPPL,ISTATUS=ACTIVE
         COMDEF NAME=MESW,PROG=XIMD
         COMDEF NAME=SERB,PROG=SERB
```

  Activate the server resource (ALCSSERB in this example) using the ZACOM ACT command or by specifying ISTATUS=ACTIVE in the ALCS communication definition. Do not activate the client resource (ALCSCLNB in this example) since it will be activated automatically by the remote system.

  When the TCP/IP server resource receives an inbound connection request, ALCS scans all the TCP/IP client resources, looking for a definition which matches the remote host that is trying to connect. This match is based on:

  - same protocol (MATIP Type B)
  - same remote IP address (COMDEF RHOST)
  - same remote port number (COMDEF RPORT) if RPORTMATCH=YES is specified on the client resource definition.
  - same local port number for the server (COMDEF SPORT) if SPORTMATCH=YES is specified on the client resource definition.

  You can specify up to four values on the RHOST parameter. ALCS compares them all with the remote IP address for inbound connection requests; this may be useful if the remote system uses primary and alternate IP addresses for the connection.

  ALCS automatically activates the matching TCP/IP client resource, which takes over control of the new connection from the TCP/IP server resource.

The COMDEF parameters RPORTMATCH and SPORTMATCH parameters can help to distinguish between services on the remote host which use the same protocol (for example MATIP Type B) and share the same same remote host IP address.

- RPORTMATCH allows the remote system to use the same IP address for similar services, by assigning a particular port number on their system to each connection.

  For example, two MATIP Type B host-to-host connections using the same remote IP address and different remote port numbers:

```
COMDEF NAME=MATIPBS,TERM=(SERVER,MATIPB),                    -
      APPL=SERA,MAXCONN=2,LPORT=351
COMDEF NAME=MATIPBC1,TERM=(CLIENT,MATIPB),                   -
      APPL=HTH1,HEX=3344,HEN=5566,                           -
      RPORT=7701,RPORTMATCH=YES,                             -
      RHOST=9.188.100.13
COMDEF NAME=MATIPBC2,TERM=(CLIENT,MATIPB),                   -
      APPL=HTH2,HEX=7788,HEN=9900,                           -
      RPORT=7702,RPORTMATCH=YES,                             -
      RHOST=9.188.100.13
```

- SPORTMATCH allows the remote system to use the same IP address for similar services, by assigning a particular port number on ALCS for each connection.

  For example, two MATIP Type B host-to-host connections using the same remote IP address and different local server port numbers:

```
COMDEF NAME=MATIPBS1,TERM=(SERVER,MATIPB),                   -
      APPL=SERB,MAXCONN=2,LPORT=1351
COMDEF NAME=MATIPBS2,TERM=(SERVER,MATIPB),                   -
      APPL=SERB,MAXCONN=2,LPORT=2351
COMDEF NAME=MATIPBC1,TERM=(CLIENT,MATIPB),                   -
      APPL=MESW,HEX=3344,HEN=5566,                           -
      SPORT=1351,SPORTMATCH=YES,                             -
      RHOST=9.188.100.13,RPORT=351
COMDEF NAME=MATIPBC2,TERM=(CLIENT,MATIPB),                   -
      APPL=MESW,HEX=7788,HEN=9900,                           -
      SPORT=2351,SPORTMATCH=YES,                             -
      RHOST=9.188.100.13,RPORT=351
```

Type A host-to-host message support using TCP/IP

ALCS supports the MATIP protocol for messages between host systems (Type A host-to-host traffic). Define a TCP/IP client resource in the ALCS communication table with the IP address, port number, and high-level designator of the remote host system. For example:

```
**       Define Type A client
         COMDFLT CLEAR,LDTYPE=TCPIP,ISTATUS=INACTIVE
         COMDEF NAME=ALCSCLNA,TERM=(CLIENT,MATIPA,HTHI),             -
               APPL=HTHA,HEX=1122,                                   -
               RPORT=6600,                                           -
               RHOST=9.188.100.13
**       Define application
         COMDFLT CLEAR,LDTYPE=ALCSAPPL,ISTATUS=ACTIVE
         COMDEF NAME=HTHA,PROG=HTHA
```

Either ALCS or the remote system can initiate the TCP/IP connection.

- ALCS can activate the client connection in two ways:
  - Specify COMDEF ISTATUS=ACTIVE in the ALCS communication definition.
  - Use the ZACOM ACT command.

  You can specify up to four values on the RHOST parameter. ALCS tries to connect using each address each in turn; this may be useful if the remote system uses primary and alternate IP addresses for the service.
- Alternatively, an application on the remote system can start the socket connection. To allow this, you must define a TCP/IP server resource as well as the client resource, in order to monitor for the inbound connection requests. The remote host connects to the port number defined on the COMDEF LPORT parameter for the TCP/IP server resource. For example:

```
**       Define Type A client and server
         COMDFLT CLEAR,LDTYPE=TCPIP,ISTATUS=INACTIVE
         COMDEF NAME=ALCSSERA,TERM=(SERVER,MATIPA),                    -
               APPL=SERA,MAXCONN=2,LPORT=350
         COMDEF NAME=ALCSCLNA,TERM=(CLIENT,MATIPA,HTHI),               -
               APPL=HTHA,HEX=1122,                                     -
               RPORT=6600,                                             -
               RHOST=(9.188.100.13,9.188,100,14)
**       Define application
         COMDFLT CLEAR,LDTYPE=ALCSAPPL,ISTATUS=ACTIVE
         COMDEF NAME=HTHA,PROG=HTHA
         COMDEF NAME=SERA,PROG=SERA
```

Activate the server resource (ALCSSERA in this example) using the ZACOM ACT command or by specifying ISTATUS=ACTIVE in the ALCS communication definition. Do not activate the client resource (ALCSCLNA in this example) since it will be activated automatically by the remote system.

When the TCP/IP server resource receives an inbound connection request, ALCS scans all the TCP/IP client resources, looking for a definition which matches the remote host that is trying to connect. This match is based on:

- same protocol (MATIP Type A)
- same remote IP address (COMDEF RHOST)
- same remote port number (COMDEF RPORT) if RPORTMATCH=YES is specified on the client resource definition.
- same local port number for the server (COMDEF SPORT) if SPORTMATCH=YES is specified on the client resource definition.

You can specify up to four values on the RHOST parameter. ALCS compares them all with the remote IP address for inbound connection requests; this may be useful if the remote system uses primary and alternate IP addresses for the connection.

ALCS automatically activates the matching TCP/IP client resource, which takes over control of the new connection from the TCP/IP server resource.

The COMDEF parameters RPORTMATCH and SPORTMATCH parameters can help to distinguish between services on the remote host which use the same protocol (for example MATIP Type A) and share the same same remote host IP address.

- RPORTMATCH allows the remote system to use the same IP address for similar services, by assigning a particular port number on their system to each connection.

  For example, two MATIP Type A host-to-host connections using the same remote IP address and different remote port numbers:

```
         COMDEF NAME=MATIPHS,TERM=(SERVER,MATIPA),                     -
               APPL=SERA,MAXCONN=2,LPORT=350
         COMDEF NAME=MATIPHC1,TERM=(CLIENT,MATIPA,HTHI),               -
               APPL=HTH1,HEX=1111,                                     -
               RPORT=6601,RPORTMATCH=YES,                              -
               RHOST=9.188.100.13
         COMDEF NAME=MATIPHC2,TERM=(CLIENT,MATIPA,HTHI),               -
               APPL=HTH2,HEX=1112,                                     -
               RPORT=6602,RPORTMATCH=YES,                              -
               RHOST=9.188.100.13
```

- SPORTMATCH allows the remote system to use the same IP address for similar services, by assigning a particular port number on ALCS for each connection.

  For example, two MATIP Type A host-to-host connections using the same remote IP address and different local server port numbers:

```
         COMDEF NAME=MATIPHS1,TERM=(SERVER,MATIPA),                    -
               APPL=SERA,MAXCONN=2,LPORT=1350
         COMDEF NAME=MATIPHS2,TERM=(SERVER,MATIPA),                    -
               APPL=SERA,MAXCONN=2,LPORT=2350
```

```
         COMDEF NAME=MATIPHC1,TERM=(CLIENT,MATIPA,HTHI),               -
               APPL=HTH1,HEX=1111,                                     -
               SPORT=1350,SPORTMATCH=YES,                              -
               RHOST=9.188.100.13,RPORT=350
         COMDEF NAME=MATIPHC2,TERM=(CLIENT,MATIPA,HTHI),               -
               APPL=HTH2,HEX=1112,                                     -
               SPORT=2350,SPORTMATCH=YES,                              -
               RHOST=9.188.100.13,RPORT=350
```

TCP/IP native sockets interface

About this task

ALCS application programs can use the MVS CALL macro (TCP/IP macros are not supported) to invoke native TCP/IP sockets support. Application programs can use stream sockets (TCP protocol), datagram sockets (UDP protocol) and raw sockets (ICMP and IP protocols). Application programs can enable server or client services, but can only service one connection at a time; and cannot pass the connection or the conversation to another entry. The entry that allocates the socket has to manage the connection and the exchange of data through the socket.

The following are the necessary steps to run a program using native TCP/IP support:

Procedure

1. Decide if it is a server or a client service that is required

   A client connection is made when ALCS programs need to connect to specific TCP/IP server applications in order to exchange data. A server connection is needed when ALCS programs are prepared to accept sockets from any client application in order to exchange data. If the decision is to implement a server connection then another decision must be made - to use the ALCS concurrent server listener or not. If you decide to not use the concurrent server listener then your ALCS application program must be permanently active (long running ECB) listening for socket connections.
2. Specify ALCS system configuration parameters

   To activate the support of native TCP/IP sockets in ALCS, specify `TCPIP=(YES,number_of_threads)` on the `SCTGEN` generation macro. The number of threads is the maximum of concurrent ALCS ECB programs that use native TCP/IP sockets.

   If you want to use the ALCS concurrent server listener, specify `TCPLIST=(YES,number_of_threads)` and optionally `TCPPORT=(port_number_1,...)` on the `SCTGEN` generation macro. This number of threads is the maximum of concurrent sockets connections to be accepted by the ALCS concurrent server(s). The port numbers are the ports where the ALCS concurrent servers listen for inbound socket connections. If `TCPPORT` is not specified, the port must be specified using the `ZCTCP LISTEN,START` command.
3. Connect ALCS with TCP/IP using the ZCTCP command

   `ZCTCP CONNECT` connects the ALCS address space with the TCP/IP address space in z/OS and `ZCTCP DISCONNECT` ends the connection. `ZCTCP LISTEN,START` starts one of the ALCS concurrent servers, and `ZCTCP LISTEN,STOP` stops it.
4. Code, compile and link-edit the application programs that use calls to TCP/IP socket services

   Refer to the *z/OS Communications Server IP API Guide* for information about sockets calls. The link-edit must include the ALCS stub module called CDSN.
5. Load and run the ECB programs to implement the server or client socket connections

ALCS concurrent server (Listener)

The ALCS concurrent server is a way to avoid creating a long-running entry when an application program is listening for new socket connections.

The ALCS concurrent server is a service that creates a server socket connection and listens for client socket connections on behalf of your application. This service avoids the need of long running ECBs listening for sockets.

ALCS supports up to 8 concurrent servers active at the same time, each using a different TCP/IP port number. When a client socket connection arrives, ALCS accepts it, creates an ECB and passes control to the ECB-controlled exit program ATCP. ATCP (or a program that ATCP calls) must issue receive and send TCP/IP calls to continue with the conversation with the client. The input and output message formats will have to be agreed between the server and the clients.

ALCS will accept sockets and activate ECBs executing ATCP concurrently up to the maximum number specified in the `SCTGEN TCPLIST` macro parameter. ATCP can decide to pass control to different applications that provide services to clients. Information passed to ATCP includes:

TCP/IP socket descriptor
TCP/IP port number used by the concurrent server
index number of the concurrent server.

ALCS e-mail support

ALCS transmits and receives e-mail messages over TCP/IP networks using an external mail server or mail transfer agent (MTA). The mail server is likely to be part of the customer's corporate e-mail system. ALCS uses the Internet Simple Message Transfer Protocol (SMTP) to communicate with the mail server.

E-mail support uses the ALCS native TCP/IP support and the ALCS concurrent server listener.

ALCS end users can compose and send outbound e-mail messages (using the `ZMAIL` command). They can receive and browse inbound e-mail messages; these messages must specify the terminal's CRN as the mailbox name. Because the mail server is part of the corporate e-mail system, ALCS end users can exchange e-mail with other corporate e-mail users. If the corporate mail server is connected to the Internet they can also exchange mail with the rest of the world.

ALCS application programs can compose and send outbound e-mail messages. They call the service program CSMS to pass messages to the ALCS e-mail support for transmission.

ALCS applications can receive and process inbound e-mail messages; these messages must specify the application's CRN as the mailbox name.

Defining MQSeries queues for ALCS

ALCS Concepts and Facilities gives an overview of the MQSeries® support in ALCS. Use the `MQM, MQMI, MQMM`, and `MQMQ` parameters of the `SCTGEN` macro to specify the MQSeries support.

ALCS initiation queue

ALCS can open an initiation queue during startup, or the ALCS operator can use the `ZCMQI` command to open this (or another) initiation queue. (*ALCS Operation and Maintenance* describes the `ZCMQI` command.)

When ALCS opens the initiation queue, it waits for trigger messages to arrive on the queue. A trigger message arrives on the initiation queue when a message is placed on an application queue that is enabled for triggering. The trigger message contains the name of the application queue and information about the application which serves the queue.

The application queue must be associated with an ALCS application which serves the queue. To do this, you must:

1. Create an WebSphere MQ for z/OS initiation queue for your application queue and specify its name in the *InitiationQName* attribute of the application queue.
2. Create an ALCS application to process trigger messages that arrive on the initiation queue.
3. Create an WebSphere MQ for z/OS process definition object and specify the CRN of your ALCS application in its *ApplId* attribute.

4. Name this process definition object in the *ProcessName* attribute of the application queue.
5. Set the trigger conditions that you require in the attributes of the application queue.

Whenever a trigger message arrives on the initiation queue that ALCS has opened, ALCS creates a new entry and copies the trigger message into a storage block attached to the ECB on level 0. Then it passes control to the ALCS application that is named in the trigger message.

An WebSphere MQ for z/OS queue manager can own more than one initiation queue, and each one is associated with one or more application queues. ALCS can only have one initiation queue open at a time. ALCS closes the initiation queue when it disconnects from the queue manager.

See *WebSphere MQ for z/OS Application Programming Reference* for a description of initiation queues, trigger messages, and the WebSphere MQ for z/OS triggering facility.

ALCS input queue

ALCS can open an input queue during startup (`MQM=(YES,CONNECT)`) ,or the ALCS operator can use the `ZCMQI` command to open this (or another) input queue. (*ALCS Operation and Maintenance* describes the `ZCMQI` command.)

When ALCS opens the input queue, it waits for messages to arrive on the queue. Messages on the ALCS input queue must be in PPMSG message format. In PPMSG format, an application routing control parameter list (RCPL) precedes the message text. The destination field of the RCPL must contain the CRI or CRN of an ALCS application which serves the application queue. Fields in the RCPL are defined in:

- Assembler programs by the RC0PL DSECT macro
- C language programs in a header file <c$rc0pl.h>.

These are described in *ALCS Application Programming Reference - Assembler* and *ALCS Application Programming Reference - C Language* respectively.

When a message arrives on the input queue that ALCS has opened, ALCS creates a new entry and copies the message text into a storage block attached to the ECB on level 0. It also copies the RCPL into the new ECB. Then it passes control to the ALCS application that is specified in the destination field of the RCPL.

ALCS closes the input queue when it disconnects from the queue manager.

ALCS MQ Bridge

The ALCS MQ Bridge allows for MQ messages received on request queues to be formatted and passed on to legacy applications as if they came from ordinary terminal devices. The output from the applications is routed from the ALCS output routines to the MQ Bridge in order to send it on to corresponding response queues.

Minimal changes are required to ALCS. The request queues are known to ALCS through communication definitions for **MQ queues**. **MQ terminals** are defined associated with a request queue so the applications have normal terminal records and addresses to deal with.

This support is intended to provide connectivity to current ALCS applications from remote systems (for example, web servers).

Exits are provided on both the input and output sides of the bridge to allow for user decoding and reformatting of messages.

For more information on the MQ Bridge contact the ALCS Support Group at alcs@uk.ibm.com.

Defining optimized local adapter (OLA) support for ALCS

ALCS Concepts and Facilities provides an overview of the optimized local adapter (OLA) support in ALCS. Use the WAS parameter of the SCTGEN macro to specify the OLA WAS support in ALCS.

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 allow 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 authority of the caller and to identify input and output messages.

ALCS WAS Bridge

The ALCS WAS Bridge allows ALCS application programs to send and receive messages using optimized local adapters 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 behavior of the WAS Bridge to suit your applications when the WAS bridge is a protocol type 1 bridge (the default). When you use a protocol type 2 bridge, you still can use ALCS WAS Bridge installation-wide monitor exits USRWAS4, and USRWAS6, but you no longer need exits USRWAS3 and USRWAS5.

ALCS allows application programs to send and receive large messages to and from terminals connected through WAS with the following restrictions:

- ALCS scroll logging is not allowed.
- ALCS message retrieval is not allowed.

Application programs can send large output messages using the SENDC X or ROUTC (ROUTC) service. The message must be in heap storage which the program has obtained using MALOC, CALOC C, and RALOC monitor-request macros, or `malloc`, `calloc`, and `realloc` C functions.

ALCS passes large input messages to application programs in heap storage which the program must release using the FREEC monitor-request macro or free C function.

For details of the AMSG and OMSG message formats and the RCPL contents see Appendix D, “Sample logon mode table,” on page 551.

For more information on the OLA communication facilities and the WAS Bridge, contact the ALCS Support Group at alcs@uk.ibm.com.

DASD space requirements for ALCS

For details of the space requirements for the distribution and target libraries, see the *ALCS Program Directory* that accompanies the machine-readable installation materials.

Real-time database and general file space requirements

The space required for the ALCS database data sets depends on the number of records specified in the ALCS DASD generation. The ALCS generation stage 1 output lists the number of records in each database data set. To convert to data set size, in tracks or cylinders, use the method described in the appropriate MVS *VSAM Administration Guide*.

The space required for the ALCS Recoup general file, general file 0 (GF-000), depends on the total number of long-term pool file records that Recoup chain-chase processes. To estimate this, divide the total number of long-term pool file records allocated by 256, and allocate that number of records in general file 0. They are size L2 records. To convert this number of records to tracks or cylinders, use the method described in the appropriate MVS *VSAM Administration Guide*.

Configuration data set space requirements

The ALCS generation stage 1 output lists the number of records in the configuration data set. To convert this number of records to tracks or cylinders, use the method described in the appropriate MVS *VSAM Administration Guide*.

Unpacking the ALCS distribution tape

ALCS installation uses System Modification Program Extended (SMP/E). For details about how to unpack the ALCS distribution tape, see the *ALCS Program Directory* that accompanies the machine-readable installation materials.

Installation information:

You should retain the *ALCS Program Directory* for future reference.

Using ISPF panels to install ALCS

ALCS provides Interactive System Productivity Facility (ISPF) panels to simplify installation tasks. Figure 11 on page 50 shows the main screen for these tasks.

```
                              Installation
Command ===> ____________________________________________________________

Choose one of the following actions, then press ENTER


Actions:
__   1.  Create installation data sets (easy option)
     2.  Create installation data sets (flexible option)
     3.  Initialize SMP/E CSIs
     4.  Receive product
     5.  Apply product
     6.  Accept product
     7.  Run post-install jobs
```

Figure 11. ISPF panel: The installation menu

Table 7 on page 50 shows where you can find more information about tasks in the ALCS installation ISPF panel.

Table 7. Where to find more information about the installation ISPF panel

Task	
Create data sets Easy option	Figure 12 on page 51
Create data sets Flexible option	Figure 13 on page 51
Initialize SMP/E CSIs	Figure 14 on page 51
Receive product	Figure 15 on page 51
Apply product	Figure 16 on page 52
Accept product	Figure 16 on page 52
Run Post-Install jobs	Figure 17 on page 52

Creating the ALCS installation data sets

Use the process in Figure 12 on page 51 or Figure 13 on page 51 to create the data sets. The flexible option (in Figure 13 on page 51) allows you to change the size of the space allocation.

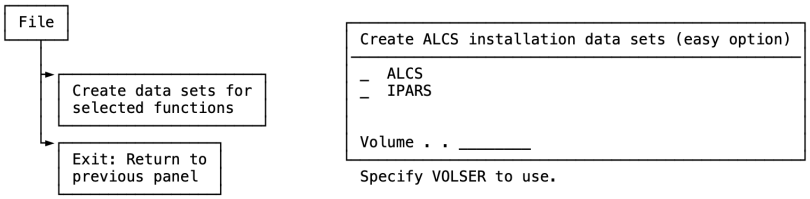


Figure 12. Route map for the ALCS installation data sets creation (easy)

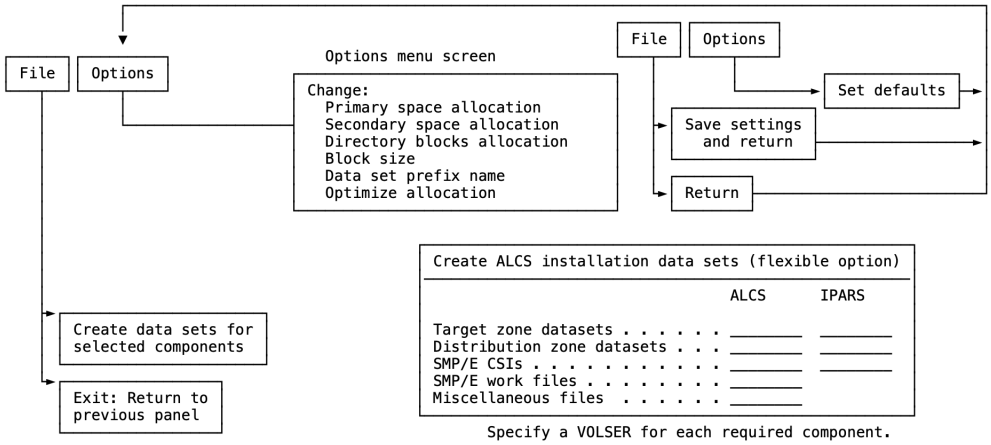


Figure 13. Route map for the ALCS installation data sets creation (flexible)

The SMP/E panels

ALCS provides the following panels for SMP/E:

- One panel to initialize the CSIs
- Three panels to invoke the SMP/E processes

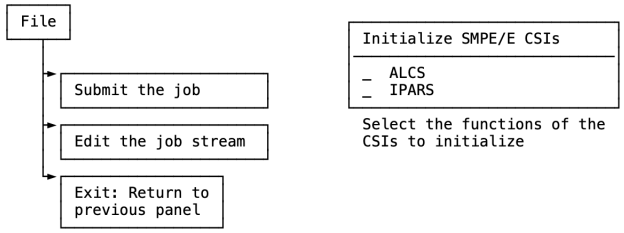


Figure 14. Route map for the SMP/E CSIs initialization

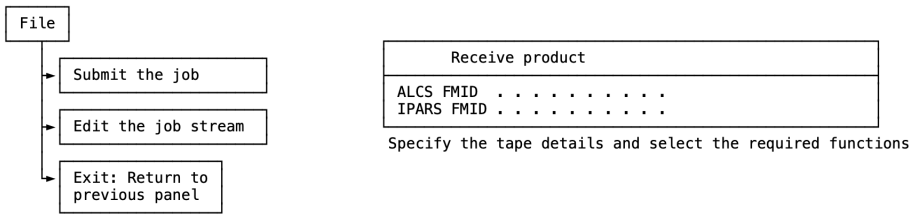


Figure 15. Route map for the SMP/E Receive the product

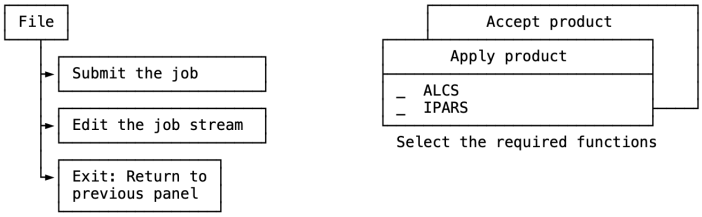


Figure 16. Route map for the SMP/E Apply and Accept the product

Installing PL/I, NetView, and Debug Tool

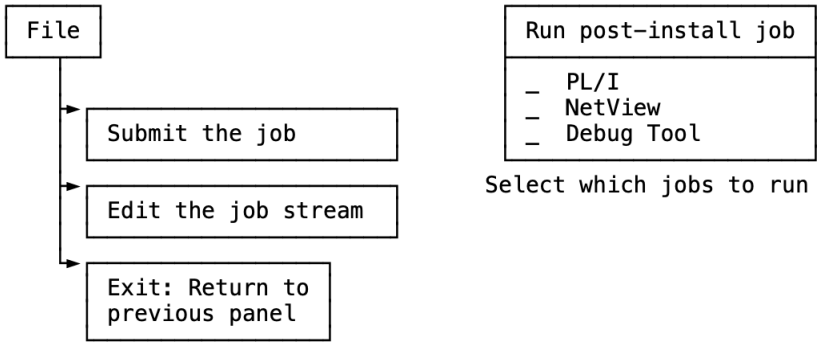


Figure 17. Route map for the Post-install jobs

Building a global tag header file c$globz.h for C language programs

In order to access your global area from programs written in C language, you must create your own version of <c$globz.h>.

Figure 18 on page 52 shows the application-development ISPF panel. Select the **Build global tag header file for C language programs** action to produce a job that uses your global area definitions and assembles a source module DXCBGLST.

DXCBGLST is in the sample library (if you use the recommended data set names in the *ALCS Program Directory*, this is DXC.V2R4M1.DXCSMP1).

```
                          Application development
Command ===> ____________________________________________________________

Choose one of the following actions, then press ENTER


Actions:
__  1.  Assemble an application program
    2.  Mass assemble an application suite
    3.  Compile a C/C++ language application program
    4.  Compile a COBOL application program
    5.  Link-edit an HLL application suite
    6.  Bind a DB2 application load module
    7.  Convert assembler DSECT to C/C++ structure
    8.  Build global tag header file for C/C++ programs
    9.  Assemble 3270 screen maps
```

Figure 18. ISPF panel: The Application development menu

The second step of the job executes the DXCBGTAG program to process the output from the assembler and creates your version of <c$globz.h> which matches your assembler definitions.

If you change the assembler definitions for your global area you must recreate <c$globz.h> to use the changes in your C programs.

Depending on the nature of the change you may need to recompile some or all of the C language programs that access the global area using your new <c$globz.h>.

Defining and updating the program configuration table

Use the **Build program list** option in the **Generation and database creation** ISPF panel to define the program configuration table.

“Updating the program configuration table” on page 213 describes how to update the program configuration table.

Chapter 3. Automated operations

NetView offers a variety of system and network management capabilities for the MVS environment including support for automated operations. The NetView program-to-program Interface (PPI) allows application programs (for example, ALCS) to exchange data with NetView. NetView acts as a server and manages the queues and message forwarding services.

The following information identifies the customization required in ALCS and NetView to enable CRAS operation from NetView operator terminals. With the appropriate changes defined in this chapter, ALCS uses the PPI to forward messages to Prime CRAS, RO CRAS, and alternate CRAS terminals. Messages sent to these CRAS resources can be selected for automation and thereby initiate a programmed response. A typical programmed response is controlled by a NetView command list (clist) program written in REXX.

ALCS events that require a programmed response are identified by the entry of specific messages in the NetView automation table. For example the programmed response can be:

- Responding to ALCS with an operator command
- Highlighting the message on NetView
- Producing a Network Problem Determination Application (NPDA) alert.

Use of automated operations techniques provides several benefits, these include:

- Improvement in system and network availability
- The ability to respond quickly to any system message
- Accurate and consistent message response 24 hours a day
- Reduced levels of human intervention
- Unmanned management of remote systems.

IBM Tivoli Netview for z/OS Automation Guide provides information about using the NetView program to perform system automation, network automation, or both. You must read this book before attempting to use the automated operations feature of ALCS.

NetView environment for ALCS automated operations

For effective system automation, NetView should be started before all other tasks and subsystems. This allows NetView to control the MVS system console operation as well as the startup and control of all other tasks including JES, VTAM, and ALCS. NetView must also be started as an MVS subsystem to enable the interception of messages sent to the MVS system console. These messages can then trigger an action to handle the event being reported. For ALCS it means that any message it sends to the MVS system console can be monitored and reacted to by NetView automation programs.

ALCS automated operations are implemented by exploiting standard NetView facilities. Prime CRAS and alternate CRAS are assigned to NetView operator IDs. RO CRAS and alternate CRAS printers are assigned to NetView operator IDs which automatically route their messages to the NetView log which can be interactively browsed by all NetView operators.

ALCS messages routed to the MVS system console can be processed by NetView and the MVS Message Processing Facility (MPF). Typically, MPF is used to perform message suppression.

ALCS automated operations requires the following environment when ALCS starts:

- NetView is started as an MVS subsystem.
- NetView receiver task DXCPPI is installed.
- NetView command processor DXCCMD is installed.
- NetView (CRAS) operator IDs are logged on.
- NetView automated operators (autotasks) are started.

Note: The NetView subsystem address space must be started with a region size large enough to hold the queues for all applications that are using it.

Installation check list

About this task

Note: This refers to IBM Tivoli NetView for z/OS Version 5 Release 2. For information about configuring other releases of NetView, refer to the appropriate NetView product manuals.

ALCS automation is achieved by assigning CRAS function (Prime, RO, AT1-AT255, and AP1-AP255) to resources which are defined in, and controlled by, NetView. To achieve this, a number of definitions must be added to ALCS and NetView tables.

Use the following summary list to check that you enable automated operations in the correct sequence. Refer to the section headings for more detail.

Procedure

1. Install ALCS programs DXCCMD and DXCPPI in the NetView load module library. See "Install DXCCMD and DXCPPI" on page 55.
2. Create a member AM0TLIST in the NetView DSIPARM library. See "NetView AM0TLIST member" on page 55.
3. Define the NetView command processor for ALCS by adding CMDDEF statements to member CNMCMDU in the NetView DSIPARM library. See "NetView CNMCMDU member" on page 56.
4. Define the NetView receiver task and automated operators for ALCS in member CNMSTYLE in the NetView DSIPARM library. See "NetView CNMSTYLE member" on page 57.
5. Define NetView operator IDs for ALCS in member DSIOPF in the NetView DSIPARM library if required. See "NetView DSIOPF member" on page 57.
6. Update the NetView message automation table if required. See "NetView automation table" on page 57.
7. Update the ALCS communication generation to include the PPINAME parameter on the COMGEN macro for the base communication load module if required. See "ALCS communication generation - base load module" on page 58.
8. Build an ALCS communication load module that includes the NetView operator IDs. See "ALCS communication generation - NetView resources" on page 58.
9. Restart ALCS with the new communication load modules. See "Restart ALCS" on page 58.
10. Logon to NetView and check that you can enter ALCS commands and receive responses. See "Using automated operations" on page 58.

Install DXCCMD and DXCPPI

Install ALCS programs DXCCMD and DXCPPI into the NetView load module library. This must be a library in the STEPLIB concatenation for the NetView started task. See the *ALCS Program Directory* for instructions about running the link edit job to install DXCCMD and DXCPPI. Ensure that the NetView CSECT called CNMNETV is available for this link edit job.

ALCS provides ISPF panels to install DXCCMD and DXCPPI. If you prefer to develop your own JCL procedures, you can use the ISPF panels to generate sample JCL.

NetView AM0TLIST member

The ALCS program DXCPPI runs as a NetView receiver task. It uses a parameter list that you must create. The parameter list is a member called AM0TLIST in the NetView DSIPARM library. AM0TLIST consists of a single 80-character record with the format:

Columns	Contents
1-8	Name of the NetView PPI receiver, left-justified and padded with spaces if required. This is the name that you specify on the PPINAME parameter of the ALCS COMGEN macro (see “COMGEN macro” on page 98).
10-17	Name of the NetView generic alerts autotask for ALCS, left-justified and padded with spaces if required. This is the name that you use to define the autotask in the NetView CNMSTYLE member.
19-26	Name of the NetView log autotask for ALCS, left-justified and padded with spaces if required. This is the name that you use to define the autotask in the NetView CNMSTYLE member.
37-41	NetView generic alert ID, left-justified and padded with spaces if required. Select a token to distinguish between ALCS NCB generic alerts and other generic alerts.
43-46	NetView code point for generic alert description, four hexadecimal digits in the range X'E000' through X'EFFF'.
48-51	NetView code point for generic alert probable cause, four hexadecimal digits in the range X'E000' through X'EFFF'.
53-56	NetView code point for generic alert recommended action, four hexadecimal digits in the range X'E000' through X'EFFF'.
58-80	Available for comments, sequence numbers, and so on. DXCPPI does not check the contents of this field.

All other columns must contain space characters.

```
(1)      (10)     (19)              (37)
ALCSAUTO DXCCNM   DXCPPI            FEB01 EC04 EC04 EC04
```

Figure 19. Example of AM0TLIST showing layout

Refer to your NetView publications for more information about NetView alert IDs and code points.

NetView CNMCMDU member

You must define the command that NetView operators use for directing ALCS commands and other messages to ALCS. You do this by adding CMDDEF statements to member CNMCMDU in the NetView DSIPARM library. The format of the CMDDEF statements for ALCS is:

```
CMDDEF.name.MOD=DXCCMD
CMDDEF.name.TYPE=R
CMDDEF.name.RES=Y
```

Where:

name
Command for directing ALCS commands and other messages to ALCS.

The following example uses ALCSCMD as the name of the command:

```
CMDDEF.ALCSCMD.MOD=DXCCMD
CMDDEF.ALCSCMD.TYPE=R
CMDDEF.ALCSCMD.RES=Y
```

Figure 20. Example of CMDDEF

NetView CNMSTYLE member

You must define the task that NetView uses for receiving messages from ALCS. You do this by adding TASK statements to member CNMSTYLE in the NetView DSIPARM library. The format of the TASK statements for ALCS is:

```
TASK.DXCRCVR.MOD=DXCPPI
TASK.DXCRCVR.PRI=1
TASK.DXCRCVR.INIT=Y
```

You must also define any automated operators for use by ALCS, by adding AUTOTASK statements to CNMSTYLE. Figure 21 on page 57 shows an example of a set of statements included in CNMSTYLE to define two autotasks for use by ALCS.

This example assumes that DXCCNM and DXCPPI are the names of the generic alerts and log autotasks in the AM0TLIST member:

```
AUTOTASK.DXCPPI.Console=*NONE*
AUTOTASK.DXCCNM.Console=*NONE*
```

Figure 21. Example of AUTOTASK

NetView DSIOPF member

Depending on the SECOPTS.OPERSEC statement in CNMSTYLE, you may need to define any NetView operators for use by ALCS in member DSIOPF in the NetView DSIPARM library. When SECOPTS.OPERSEC=SAFDEF is specified, authority to log on as a NetView operator is controlled using a SAF security product and the operator identifiers can be omitted from DSIOPF. For other values of SECOPTS.OPERSEC the operator identifiers must be defined in DSIOPF. For more information, refer to *IBM Tivoli Netview for z/OS Administration Reference*. *IBM Tivoli Netview for z/OS Security Reference*.

Figure 22 on page 57 shows an example of a set of statements included in DSIOPF to define four operator IDs for use by ALCS. This example assumes that IALPRC, IALROC, and IALAT*n* are the names of NetView resources in the ALCS communication generation.

```
****** NetView Operators (Human operators) *****
IALPRC   OPERATOR
         PROFILEN IALXMOCF
IALAT1   OPERATOR
         PROFILEN IALXMOCF
IALAT2   OPERATOR
         PROFILEN IALXMOCF
IALAT3   OPERATOR
         PROFILEN IALXMOCF
```

Figure 22. Example of DSIOPF

NetView automation table

The NetView automation table identifies which messages are automated. It consists of statements filed in a member of the NetView DSIPARM library. To define an automation table, code automation statements in DSIPARM. Then issue the NetView AUTOTBL command using the name of that specific NetView automation table. The automation table can cause various displays, logging, or routing to occur. NetView commands or CLISTs can be invoked to analyze the message before any action is taken.

Refer to your NetView publications for more information about the NetView automation table.

An example of the NetView automation table for ALCS is:

```
 IF MSGID='DXC2'. THEN
    DISPLAY(Y) NETLOG(Y) SYSLOG(N);
```

```
 IF MSGID='DXC'. THEN
    DISPLAY(Y) NETLOG(Y) SYSLOG(Y);
```

ALCS communication generation - base load module

For ALCS and NetView to communicate, they need to know each other's identification. When NetView sends data to ALCS, it uses the ALCS VTAM application name. This is the name coded on the ACBNAME parameter of the ALCS communication generation COMGEN macro. When ALCS sends data to NetView, it uses the NetView identification coded on the PPINAME parameter of the ALCS communication generation COMGEN macro. This is explained in “COMGEN macro” on page 98.

The name coded on the PPINAME parameter of the ALCS communication generation COMGEN macro must be identical to the name defined in the NetView AMOTLIST member. See “NetView AMOTLIST member” on page 55.

ALCS communication generation - NetView resources

The ALCS communication generation defines CRAS resources. For installations that are operating in a conventional manner (that is, without automated operations), the CRAS function is assigned to physical terminal devices controlled by ALCS.

For installations that exploit the automated operations capability, the CRAS function is assigned to NetView operator IDs or autotasks. The ALCS communication generation defines these NetView resources using the LDTYPE=NETVIEW parameter on the COMDEF macro. This is explained in “COMDEF macro” on page 107.

Figure 23 on page 58 shows an example of an ALCS communication generation including automated operations support.

```
******** NetView COMGEN
         COMGEN PPINAME=ALCSAUTO
******** NetView Automated Operations resources
         COMDFLT CLEAR
         COMDFLT LDTYPE=NETVIEW,APPL=RES0
******** ALCS Prime and AT CRAS
         COMDEF NAME=IALPRC,TERM=DISPLAY,CRAS=PRC
         COMDEF NAME=IALAT1,TERM=DISPLAY,CRAS=AT1
         COMDEF NAME=IALAT2,TERM=DISPLAY,CRAS=AT2
         COMDEF NAME=IALAT3,TERM=DISPLAY,CRAS=AT3
******** ALCS RO and AP CRAS
         COMDEF NAME=IALROC,TERM=PRINTER,CRAS=ROC
         COMDEF NAME=IALAP1,TERM=PRINTER,CRAS=AP1
         COMDEF NAME=IALAP2,TERM=PRINTER,CRAS=AP2
         COMDEF NAME=IALAP3,TERM=PRINTER,CRAS=AP3
```

Figure 23. ALCS communication generation - automated operations

Note: If SITA NCBs must be forwarded to NetView then alternate CRAS printer AP1 must be defined as a NetView printer. See “COMDEF parameters for NetView” on page 132 for a description of the parameters.

Restart ALCS

Restart ALCS with the new communication load modules.

Using automated operations

Use the following format to direct an ALCS command or other message to ALCS from a NetView operator ID:

```
name alcsid,[opid],input
```

Where:

name
Command name, as defined to NetView in member CNMCMDU in the NetView DSIPARM library.

alcsid
VTAM ACB name of the target ALCS system.

opid
CRN of the originator. Default is the CRN of the NetView operator ID where the command is entered.

input
ALCS command or other message. If the message consists of more that one line, you must use the field mark character (X'1E') to delimit lines.

Possible return codes from DXCCMD are:

0
Function executed successfully.

4
NetView PPI is inactive.

8
Invalid input buffer.

12
DSIGET/DSIFRE failure.

16
ALCS is inactive. ALCS is either not started or is in standby state.

Logon to NetView with the Prime CRAS operator ID (CRN IALPRC in these examples) and send commands to ALCS. Figure 24 on page 59 shows an example that uses NetView command name ALCSCMD, VTAM ACB name PYEZ1ARE, and ALCS command ZDCOM.

```
ALCSCMD PYEZ1ARE,,ZDCOM
```

Figure 24. Example of an ALCS command from NetView

Responses to ALCS commands should be displayed on the NetView screen and logged in the NetView log file. Use the NetView command BR NETLOGA to browse the NetView log file.

Messages to RO CRAS should be logged in the NetView log file. Use the NetView command BR NETLOGA to browse the NetView log file.

You can also use the NetView command STATMON to browse the NetView log file from the status monitor, and to select messages of different severity or priority. Your system programmer can customize the NetView status monitor to initiate important message indicators. Ask your system programmer or lead operator for more information about how these are used in your network.

Chapter 4. Generating ALCS

The ALCS online monitor program and some of the offline programs need information that varies from one installation to another, for example:

- The size of the real-time database
- The real-time and general sequential files that the application needs
- A description of the ALCS communication network

This information is in tables called the configuration dependent tables. These tables are assembler language CSECTs. Each table is a separate load module. Figure 25 on page 60 gives an overview of the generation process. The data set names such as SYSLIB are explained with the JCL.

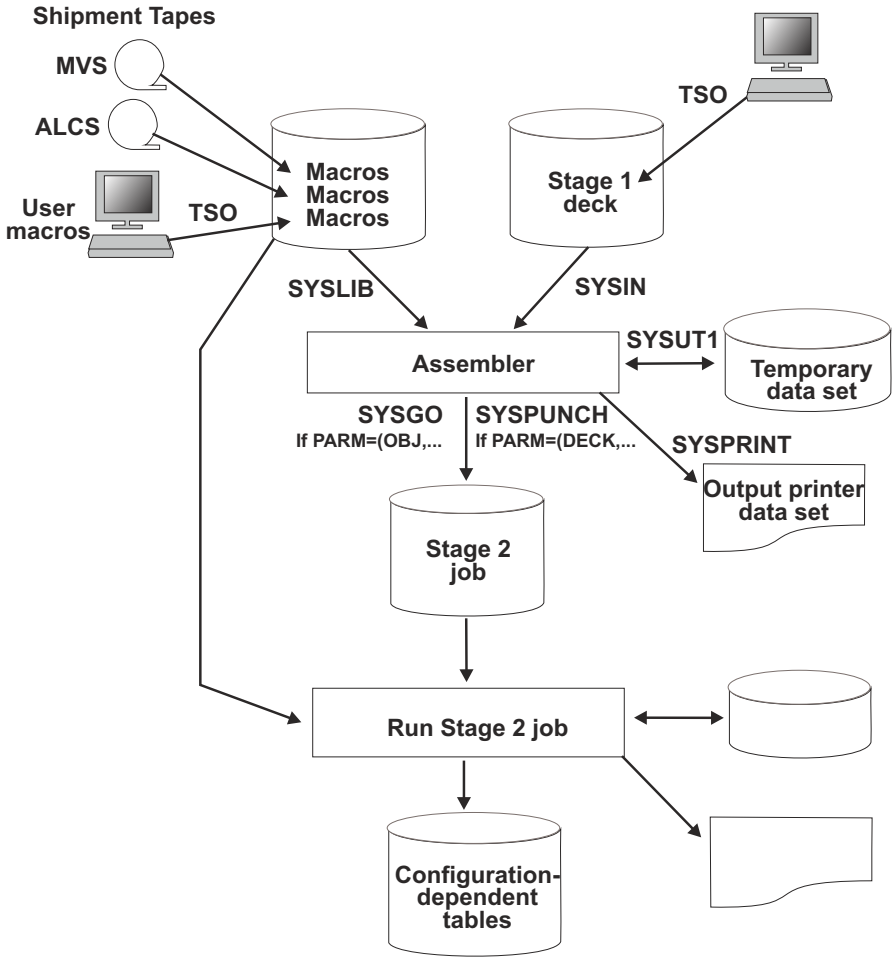


Figure 25. Overview of the ALCS tables generation

Using ISPF panels to generate ALCS

ALCS provides Interactive System Productivity Facility (ISPF) panels to simplify generation tasks. Figure 26 on page 61 shows the screen for these tasks.

```
                    Generation and database creation
Command ===> ____________________________________________________________

Choose one of the following actions, then press ENTER


Actions:
__  1.  System generation
    2.  Sequential file generation
    3.  DASD generation
    4.  Create database data set
    5.  Create test database data set
    6.  Build program List
    7.  Communication generation
    8.  Build communication list
    9.  Run communication report file generator
```

Figure 26. ISPF panel: The generation menu

Table 8 on page 61 shows where you can find more information about tasks in the ALCS generation ISPF panel.

Table 8. Where to find more information about the generation ISPF panel

Task	
System Generation	Figure 27 on page 62
Communications generation	Figure 27 on page 62
DASD generation	Figure 27 on page 62
Sequential file generation	Figure 27 on page 62
Create database data set	Figure 28 on page 63
Create test database data set	“Creating a test data set” on page 462
Build Program List	“Updating the program configuration table” on page 213
Run communication list utility	*ALCS Operation and Maintenance*
Build Communication list	“Updating the communication configuration load list” on page 205

Figure 27 on page 62 shows the route map for generating system, sequential file, DASD, and communication tables (options 1,2,3, and 7 on the generation menu).

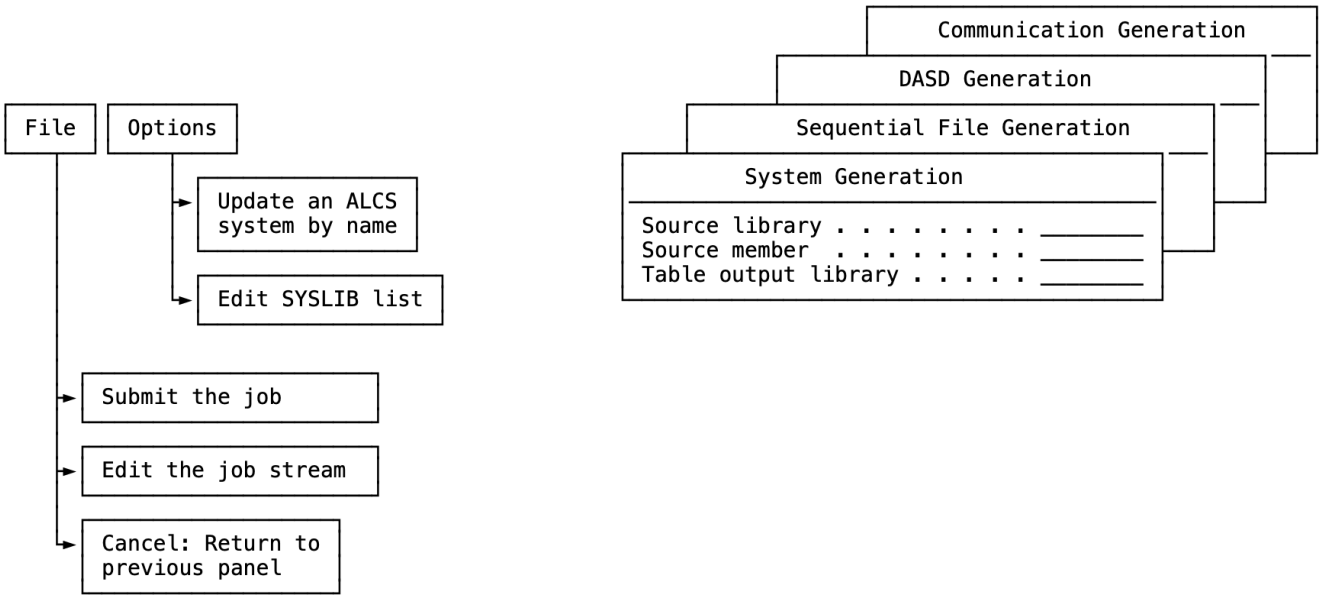


Figure 27. Route map for generating: System, sequential file, DASD, and communication tables

Figure 28 on page 63 shows the route map for creating database data sets (option 4 in the generation menu).

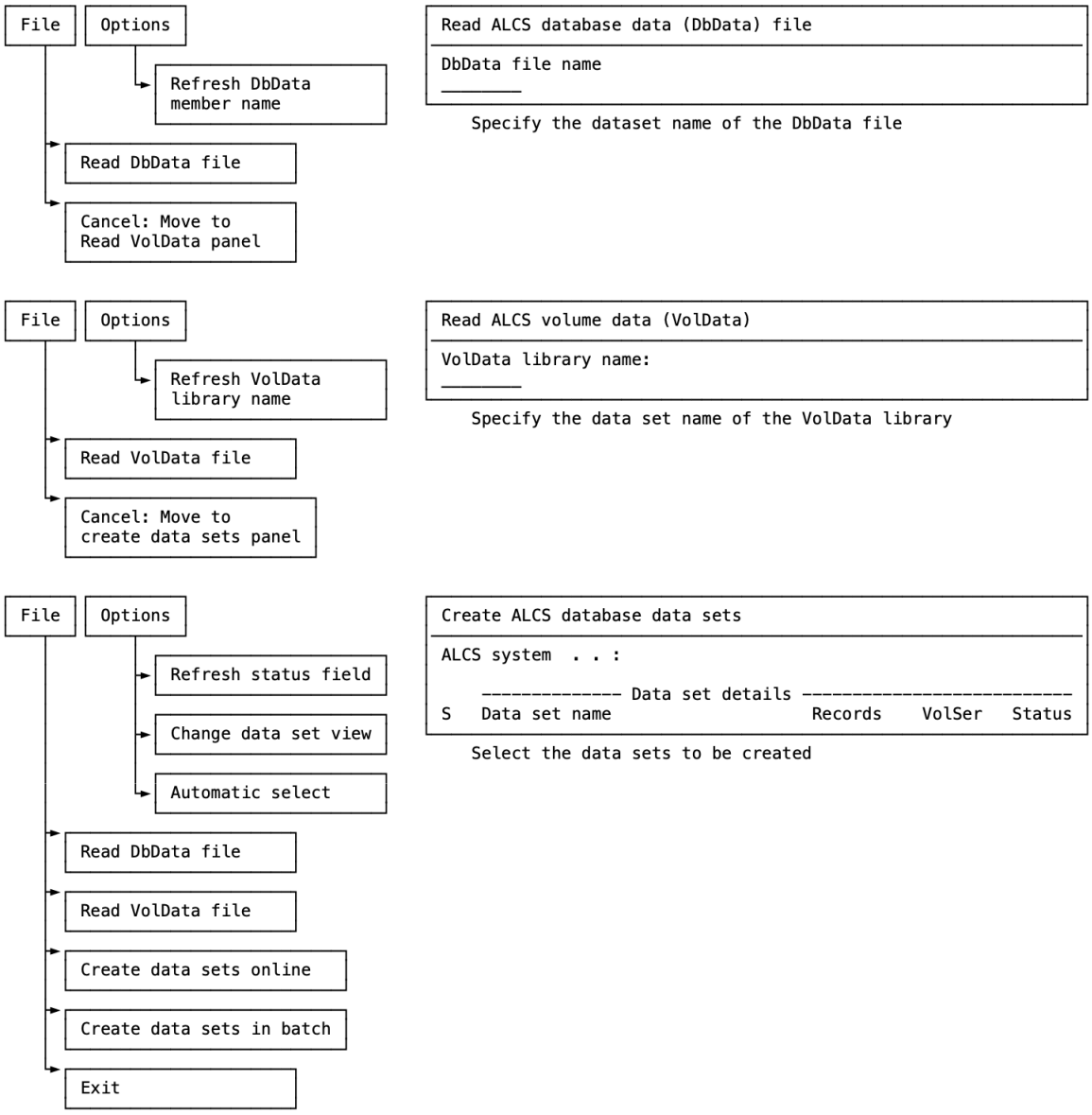


Figure 28. Route map for creating ALCS database data sets

Table 9 on page 63 shows where you can find more information about system generation.

Table 9. Where to find more information about generation tasks

Task	
System	"Generating the system configuration table" on page 72
Communications	"Generating a communication load module" on page 96
DASD	"Generating a DASD configuration table" on page 173
Sequential File	"Generating a sequential file configuration table" on page 165
Database data sets	"Using ISPF panels to create the data sets" on page 457
Test database	"Creating a test data set" on page 462
Build Program List	"Updating the program configuration table" on page 213

Table 9. Where to find more information about generation tasks (continued)	
Task	
Run communication list utility	*ALCS Operation and Maintenance*
Build communication list	"Updating the communication configuration load list" on page 205

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 in batch 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**.

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

System programmer tasks

About this task

To generate an ALCS system, the system programmer must:

Procedure

1. Code the ALCS generation macroinstructions. A complete set of macroinstructions specify:
   - Configuration dependent information for ALCS
   - Job control details for stage 2 of the generation
2. Assemble the macroinstructions. This is stage 1 of the ALCS generation procedure. The output consists of:
   - A job stream punched to a data set with ddname SYSGO or SYSPUNCH. This is the input to stage 2 of the generation procedure.
   - A listing of this job stream, with information, attention and error messages. This is on a data set with ddname SYSPRINT.
3. Execute the job stream generated by stage 1. This is stage 2 of the ALCS generation procedure. The job stream consists of job steps that:
   - Assemble configuration dependent tables, and put the resulting object modules in a temporary data set.
   - Link-edit these object modules, and put the resulting load modules in a library.

Results

Table 10 on page 64 shows where you can find more information about ALCS system generation.

Table 10. Where to find more information about ALCS generation	
Macro	
General	Page "Generating the initial ALCS configuration" on page 65 and Page "Updating the ALCS configuration" on page 68
ALCS	Page "ALCS macro" on page 70
ALCSGEN	Page "ALCSGEN macro" on page 199

Table 10. Where to find more information about ALCS generation (continued)	
Macro	
COMDEF	Page “COMDEF macro” on page 107
COMGEN	Page “COMGEN macro” on page 98
DBGEN	Page “DBGEN macro” on page 186
DBHIST	Page “DBHIST macro” on page 195
DBSPACE	Page “DBSPACE macro” on page 198
GFGEN	Page “GFGEN macro” on page 193
SCTGEN	Page “SCTGEN macro” on page 72
SEQGEN	Page “Generating a sequential file configuration table” on page 165
USRDTA	Page “USRDTA macro” on page 190

Run ALCS generation to generate the initial configuration. Subsequent runs generate modifications to the initial configuration. “Generating the initial ALCS configuration” on page 65 describes the input deck for the initial configuration. “Updating the ALCS configuration” on page 68, describes the input deck for modifying the configuration.

Sequence of generation macroinstructions

This section explains how to prepare the stage 1 generation deck as part of your ALCS generation.

Note for ISPF panels:

If you use ISPF panels to generate your system configuration, **do not** code the following macroinstructions:

ALCS
JOBCARD
ALCSGEN
The final END

The ISPF panel function adds the correct version of these from its configuration data.

“Generating sample job streams” on page 64 describes how to produce a copy of a job stream.

Generating the initial ALCS configuration

About this task

The ALCS stage 1 generation deck comprises a sequence of ALCS generation macroinstructions, followed by an assembler END instruction. It can (optionally) also include assembler language comments and TITLE, SPACE, and EJECT instructions. The sequence of ALCS generation macroinstructions is:

Procedure

1. An ALCS macroinstruction. Omit this if you are using the ALCS ISPF panels for the generation.
2. An optional JOBCARD macroinstruction. Omit this if you are using the ALCS ISPF panels for the generation.
3. Generation macroinstructions to build one or more of the following configuration-dependent tables:
   - System configuration
   - Communication configuration

- Sequential file configuration
- DASD configuration.

4. An ALCSGEN macroinstruction. Omit this if you are using the ALCS ISPF panels for the generation.

Results

Additional to the configuration dependent tables that you build using the ALCS generation macros, you also need to build a communication load list and a program configuration table.

Using multiple stage 1 generation decks

You can use a single stage 1 generation deck to build all the configuration-dependent tables for your initial configuration. But IBM recommends that you use separate decks for each configuration-dependent table (the ALCS ISPF panels always use separate decks).

You may not be able to define a large communication configuration in a single stage 1 generation deck. In this case, you must split the communication configuration definition into several stage 1 generation decks.

If you use separate stage 1 generation decks for each configuration-dependent table, be sure to specify the same parameters on the ALCS macroinstruction in all the decks for the same system (the ALCS ISPF panels do this automatically).

System configuration

To define the initial system configuration, you include a single SCTGEN macroinstruction.

Communication configuration

To define the initial communication configuration, include a single COMGEN macroinstruction, followed by COMDEF macroinstructions. You can optionally include COMDFLT macroinstructions to save repeating the same parameters on many COMDEF macroinstructions.

If your communication configuration is too big for a single stage 1 generation deck, you can create a base load module for part of the configuration, and one or more update load modules for the remainder of the configuration.

"Generating a communication load module" on page 96 explains how you create base and update communication configuration load modules.

Note: You **must** create your custom macrodefinition for DXCZCUSR before you run the stage 1 generation for your communication configuration. See "ALCS DXCZCUSR user macro" on page 444.

The names of your base and update communication configuration load modules must be defined in the **communication configuration load list**. At restart, ALCS reads the load list and loads each communication configuration load module whose name is in the load list. ALCS loads the load modules in the order they appear in the load list. You must include the base load module as the first load module in the load list, followed by the update load modules. See "Updating the communication configuration load list" on page 205 for an explanation on how you create the communication configuration load list.

Specify the name of the communication configuration load list in the PARM field on the EXEC statement that starts the ALCS job. At restart, ALCS loads all the communication configuration load modules that are referenced in the load list. If your ALCS system uses the optional communication configuration data set (CDS2), the communication configuration load list is written to CDS2 during the initial startup of your ALCS system. Subsequent restarts of ALCS load the communication configuration load modules that are referenced in the load list, which was written as the first item on CDS2.

Sequential file configuration

To define the initial sequential file configuration, include one SEQGEN macroinstruction for each sequential file. You must define all the sequential files that you need in a single stage 1 generation deck. These include:

System sequential files
These are the sequential files that ALCS itself uses.

Application sequential files
These are the sequential files that your application uses. There are two types:

Real-time sequential files
General sequential files.

If your application sometimes uses different 3-character names to refer to the same general file, you need to define these as synonyms (the USE parameter of the SEQGEN). You include additional SEQGEN macroinstructions to define these synonyms.

"Generating a sequential file configuration table" on page 165 explains how you create sequential file configuration load modules.

DASD configuration

About this task

To define the initial DASD configuration, you include:

Procedure

1. One DBGEN macroinstruction
2. USRDTA macroinstructions that specify your real-time database requirements
3. GFGEN macroinstructions that specify your general file and general data set requirements
4. Optionally, one DBSPACE macroinstruction
5. One DBHIST macroinstruction.

Results

You must define your DASD configuration in a single stage 1 generation deck.

"Generating a DASD configuration table" on page 173 explains how you create DASD configuration load modules.

Note: You **must** create your custom macrodefinition for DXCURID before you run the stage 1 generation for your DASD configuration. See "ALCS DXCURID user macro" on page 443.

Program configuration table

This table lists load modules containing:

- Application programs
- ECB-controlled installation-wide exit programs
- Installation-wide monitor exits

At restart, ALCS loads all the load modules that you include in the sequence they appear in the list.

"Updating the program configuration table" on page 213 describes how you build the program configuration table.

Specify the name of the program configuration table in the PARM field on the EXEC statement that starts the ALCS job. At restart, ALCS loads all the program load modules and installation-wide monitor exit load modules in the program configuration table. If your ALCS system uses the optional program configuration data set (CDS1), the program configuration table is written to CDS1 during the initial startup of your ALCS system.

Subsequent restarts of ALCS load the program load modules and installation-wide monitor exit load modules that are referenced in the program configuration table, which was written as the first item on CDS1.

Generation macroinstructions

Table 11 on page 68 shows where you can find more information about ALCS system generation macros.

Table 11. Where to find more information about ALCS generation macros

Macro	
ALCS	Page "ALCS macro" on page 70
ALCSGEN	Page "ALCSGEN macro" on page 199
COMGEN COMDEF COMDFLT	Page "Generating a communication load module" on page 96
DBGEN DBHIST DBSPACE GFGEN USRDTA	Page "Generating a DASD configuration table" on page 173
JOBCARD	Page "JOBCARD macro" on page 71
SCTGEN	Page "Generating the system configuration table" on page 72
SEQGEN	Page "Generating a sequential file configuration table" on page 165

Updating the ALCS configuration

How you update the initial ALCS configuration depends on which configuration-dependent table or tables you want to change.

System configuration

Update your initial ALCS system configuration stage 1 generation deck. Change SCTGEN macro parameters as required. Then run the generation to create a new system configuration load module.

To use this new configuration, you must stop ALCS and restart it with the new configuration load module.

Communication configuration

There are two methods of applying changes to your initial ALCS communication configuration. You can either update your initial ALCS communication configuration stage 1 generation deck, or create an update communication configuration load module.

- **Updating the initial communication configuration**

  Update the COMGEN, COMDFLT and COMDEF macroinstructions in the initial communication configuration stage 1 generation deck(s), and run the generation to create the communication configuration load module(s). If the names of the communication configuration load modules remain the same, you do not need to change your communication configuration load list. If the load module names are different, create a new communication configuration load list. "Updating the communication configuration load list" on page 205 describes how you update the communication configuration load list.

- **Creating an update communication configuration**

  If frequent changes occur in your ALCS communication network, you should create an update ALCS communication configuration load module that reflects those changes. This load module should only define the **changes** to your existing ALCS communication configuration. Define the network changes in

ALCS communication generation COMGEN, COMDFLT and COMDEF macroinstructions. Create an update communication configuration stage 1 generation deck, and run the generation to create the update communication configuration load module.

Before you apply the communication network changes to your ALCS system, IBM advises you to check them by running the ALCS communication report file generator (see the *ALCS Operation and Maintenance*).

To apply the changes, you do not need to stop ALCS. You can use the ZACOM command to load the update communication configuration load module (see the *ALCS Operation and Maintenance*). When you are confident that ALCS functions correctly with the changed configuration, take one of the following actions to ensure that the update communication configuration load module is reloaded automatically during a restart of the ALCS system.

- Add the name of update communication configuration load module to the end of the load module list in the communication configuration load list. "Updating the communication configuration load list" on page 205 describes how you update the communication configuration load list.
- If your ALCS system is using the communication configuration data set (CDS2) to manage the communication configuration, use the ZACOM CONFIRM command to request ALCS to reload the update communication configuration load module during the next restart of the ALCS system. "Using the Communication Configuration Data Set" on page 201 describes how you use the communication configuration data set (CDS2).

"Loading communication configuration load modules" on page 200 describes how the communication configuration load list is updated and the procedures for loading and consolidating update communication configuration load modules.

Sequential file configuration

Update your initial ALCS sequential file configuration stage 1 generation deck. Change SEQGEN macro parameters, or add or delete SEQGEN macroinstructions, as required. Then run the generation to create a new sequential file configuration load module.

To use this new configuration, you do not need to stop ALCS. You can use the ZASEQ command to load the new sequential file configuration load module (see the *ALCS Operation and Maintenance*).

When you are confident that ALCS functions correctly with the new configuration, you can change the ALCS parameters to use it automatically at every restart.

DASD configuration

Update your initial ALCS DASD configuration stage 1 generation deck. Do **not** change any of the existing macroinstructions. Instead, specify the changes you require by **adding** macroinstructions (see "Generating a DASD configuration table" on page 173).

Be sure to include at least one DBHIST macroinstruction that describes the configuration change - this provides an audit trail.

To apply the changes, you do not need to stop ALCS. You can use the ZDASD command to load the update DASD configuration load module (see the *ALCS Operation and Maintenance*).

Program configuration table

You can use the ZPCTL command (see the *ALCS Operation and Maintenance*) to load new or changed:

- Application programs
- ECB-controlled installation-wide exit programs
- Installation-wide monitor exits

Create additional load modules for the application programs and installation-wide exits that you need to load online. Use the ZPCTL command to load each additional load module. When you are confident that

ALCS functions correctly with the new or changed programs and exits, take one of the following actions to ensure that the load module is reloaded automatically during a restart of the ALCS system.

- Relink-edit your existing load modules to include the new or changed programs and installation-wide exits.
- Update the load list in the program configuration table to include the name of the additional load module.
- If your ALCS system is using the program configuration data set (CDS1) to manage the program and installation-wide monitor exit load modules, use the `ZPCTL CONFIRM` command to request ALCS to reload the load module during the next restart of the ALCS system.

"Loading program and installation-wide monitor exit load modules" on page 207 describes how the program configuration table is updated and the procedures for loading programs and installation-wide exits.

ALCS macro

Use ALCS to specify job control details for the stage 2 job stream. ALCS also defines ALCS storage block sizes.

Note for ISPF panels:

If you use ISPF panels to generate your system configuration, **do not** code this macroinstruction. Instead, update your ISPF system definitions with any changes.

"Generating sample job streams" on page 64 describes how to produce a copy of a job stream.

```
[label] ALCS ID=alcsid
                        ,VERSION=alcsver
                        ,SOURCE=(sourcepds,....)
                        ,LOAD='loadpds'
                        [,DBDATA=dbdatapds]
                        [,RECSIZE=([381|r1],{1055|r2},{4000|r3},[r4],...[r8])]
                        [,CISIZE=([512|c1},{1536|c2},{4096|c3},[c4],...[c8])]
                        [,PROC=(([HLASMC|proc1][,C|,step1])
                                    ,({LKED|proc2][,LKED|,step2]))]
```

Where:

label
Any valid assembler label.

ID=*alcsid*
ALCS system identifier (ID); 1 alphanumeric character. Use the system ID to identify different ALCS systems that can run at the same time. For example, use a different system ID for each test ALCS system. ALCS messages to the MVS operator include the system ID.

VERSION=*alcsver*
ALCS system version; 1 alphanumeric character. Use the system version to identify different ALCS configurations. ALCS messages to the MVS operator include the system version.

SOURCE=(*sourcepds,....*)
Data set names, separated by commas, of cataloged libraries that contain the macrodefinitions which the stage 2 generation uses. The stage 2 generation job concatenates these libraries in the order in which you specify them here.

You must specify at least the following libraries preferably in this order:

1. The library that contains the ALCS general purpose programming interface macrodefinitions.
2. The library that contains the ALCS product-sensitive programming interface definitions.

3. The library that contains the ALCS internal-use macrodefinitions.
4. The library that contains your installation's macrodefinitions for DXCURID.

LOAD=*'loadpds'*
Data set name of a cataloged library. When the stage 2 job stream generates load modules, it puts them in this library.

DBDATA=*dbdatapds*
Data set name of a cataloged library. When the stage 2 job stream generates IDCAMS data for the ALCS data sets, it puts them in this library. This data is used by the ALCS panel-driven installation and customization routines.

RECSIZE=([381|*r1*],{1055|*r2*},{4000|*r3*},[*r4*],[*r5*],[*r6*],[*r7*],[*r8*])
User record length for sizes L1 through L8. Sizes L1 through L3 always exist; sizes L4 through L8 exist only if the size is defined here. The minimum value for size L1 is 381, for size L2 it is 1055, and for size L3 it is 4000. The minimum value for L4 through L8 is 9.

The maximum value for RECSIZE is 32704.

ALCS and TPF compatibility:

If you want to maintain compatibility with ALCS/VSE and TPF, use default sizes for L1 through L3 and 4095 for L4. (L3 is used by ALCS, whether or not your application programs use it.)

CISIZE=([512|*c1*],{1536|*c2*},{4096|*c3*},[*c4*],[*c5*],[*c6*],[*c7*],[*c8*])
Control interval sizes for DASD records, and the internal block sizes used in ALCS, for sizes L1 through L8. They must be valid VSAM control interval sizes. Valid sizes are *n* × 512 or *n* × 2048 where *n* is 1 through 16. Each must be at least 56 bytes more than the corresponding RECSIZE. *c4* through *c8* are mandatory if record sizes L4 through L8 are specified on the RECSIZE parameter. Otherwise omit *c4* through *c8*.

PROC=(([HLASMC|*proc1*][,C|,*step1*]),({LKED|*proc2*][,LKED|,*step2*]))
Names of cataloged job control procedures, and the steps within the procedures, that the stage 2 job stream uses to execute the following:

proc1
Procedure that executes the assembler program.

step1
Step name of the step (in procedure *proc1*) that executes the assembler program.

proc2
Procedure that executes the linkage editor program.

step2
Step name of the step (in procedure *proc2*) that executes the linkage editor program.

HLASMC and LKED are IBM-supplied cataloged procedures.

Note: This parameter is meaningful only if you code STAGE2=YES on the ALCSGEN macro (for more information see "ALCSGEN macro" on page 199).

JOBCARD macro

This macro is optional. Use JOBCARD to override the default job card and to include additional job control statements.

Note for ISPF panels:

If you use ISPF panels to generate your system configuration, **do not** code this macroinstruction. Instead, update your ISPF system definitions with any changes.

"Generating sample job streams" on page 64 describes how to produce a copy of a job stream.

```
[label]  JOBCARD ['jobtext',...]
```

Where:

label
Any valid assembler label.

'jobtext',...
Text of one or more job control statements, separated by commas. Enclose each statement in single quotes. There can be a maximum of 16 parameters. The stage 1 generation process punches these statements, in the order specified, at the start of the stage 2 generation job. The quotes that enclose *jobtext* are not punched. If this macro is omitted, or no *jobtext* parameters are coded, the stage 2 job stream starts with the following job control statements:

```
//DXCGENS2 JOB
//* ALCS SYSTEM i VERSION v GENERATION - STAGE 2
```

Generating the system configuration table

Use the SCTGEN macro to generate the ALCS system configuration table. To load the system configuration table, include the name of the table in the PARM field on the EXEC statement that starts ALCS. The MEMBER parameter of SCTGEN specifies the system configuration table name.

SCTGEN macro

Use SCTGEN to specify system control parameters for ALCS.

```
[label]  SCTGEN DATE=(dd,mmm,yyyy)
                ,TIME=(hhmm[,[+|-]ggnn])
                ,{MAXENT=maxent|(MSGS=msgs}
                ,{NBRSU=([num_type_1],[0|num_type_2],[0|num_type_3])|
                   ,WSBREQ=(b1,b2,b3,...)}
                [,ACV=({av1],[av2],[2|av3],[5|av4],[av5],[80|av6])]
                [,AMODE31=(FORCE,{GLOBAL|NOTGLOBAL})]
                [,AMODE64=({[{VFA|LPSVFA}][,MSGTRACE])]
                [,APPC={NO|YES|(YES,1000|num_srbs)}]
                [,CRET={128|cret}]
                [,CTLERR=(options)]
                [,DATEFORM={'DD/MM/YY'|'format_string'}]
                [,DB2={NO|(YES[,threads])}]
                [,DB2RECONNECT={NO|YES}]
                [,DB2SSNM=ssnm]
                [,DCLECBU={0|usersize}]
                [,DCLOPTS={ALL|({IO|WAIT|MSG|SLC|PGM|ECB|CPU}[,...])}]
                [,DDTIME={0|dd|255}]
                [,DIADUMP={NO|YESdd}]
                [,DISPTIM={0|dispatcher_timeout}]
                [,DYNTCB={NO|YES}]
                [,EMAILDOMAIN=domain_name]
                [,EMAILMTA=ip_address|domain_name]
                [,EMAILPORT={25|port_number}]
                [,EMAILTIMEOUTR={30|t1}]
                [,EMAILTIMEOUTS={120|t2}]
                [,EMAILPOSTMASTER={ROC|crn}]
                [,EMAILQTIME={0|t3}]
                [,EMAILQCOUNT={0|count}]
                [,ENTERR=({0|ee1],[5|ee2])]
                [,ENTHOLD=({16|eh1],[4|eh2])]
                [,ENTLIFE=({0|el1],[60|el2])]
                [,ENTLT=({elt1],[elt2])]
                [,ENTPOOL=({255|eg1],[255|eg2])]
                [,ENTREAD=({0|er1],[0|er2])]
                [,ENTST=({est1],[est2])]
                [,ENTSTOR=({256|es1],[64|es2])]
                [,ENTSTOR1=({256|esa1],[64|esa2])]
                [,ENTSTOR2=({0|esb1],[0|esb2])]
                [,ENTSTOR3=({0|esc1],[0|esc2])]
                [,ENTWRT=({20|ew1],[5|ew2])]
                [,GLBLPROT={NO|YES}]
                [,GLBLSZE=({21|s1],[25|s2],[12|s3])]
                [,HIPER=((blks[,CASTOUT|NOCASTOUT]),...)]
                [,HOLD={128|hold}]
                [,HTTP={NO|YES|(YES,6|connections,requests|6)}
                [,JSON={NO|YES|(YES,10|instances)}
                [,ICSF={NO|YES}]
                [,LTPOOL={INUSE|FREE}]
```

```
              [,MEMBER=name]
              [,MIGSEQ={NO|YES}]
              [,MONTHABBR={(Jan, ...Dec)|(abbr_name,...)}]
              [,MONTHNAME={(January, ...December)|(name,...)}]
              [,MQM={NO|(YES[,CONNECT|NOCONNECT])}]
              [,MQMI={initiation_queue|(initiation_queue,system_state)}]
              [,MQMM=queue_manager_name]
              [,MQMQ={input_queue|(input_queue,system_state)}]
              [,MSGI=logon_response]
              [,MSGTRACE={64K|aK|bM|NO}]
              [,NBRIOB=nbriob]
              [,NOWAIT={32|n}]
              [,OPRERR=(option,...)]
              [,PAGE={NONE|ALL|(option,...)}]
              [,PDULOGSTREAM='logstream_prefix']
              [,PERFMON={NO|YES}]
              [,POOLCTL=({20|g1],[30|g2])]
              [,RACF={YES|NO}]
              [,SCRLLOG=({5000|sl1],[500|sl2])]
              [,SHOWCRAS={NO|YES}]
              [,SHOWCRN={NO|YES}]
              [,SUSIZE=({2|type_1_size],[5|type_2_size],[5|type_3_size])]
              [,SVCDUMP={NO|YES}]
              [,SYSERR=(option,...)]
              [,SYSNAME=system_name]
              [,TCPIP={NO|YES|(YES,1|threads)}]
              [,TCPLIST={NO|YES|(YES,1|threads)}
              [,TCPNAME=tcpip_address_space_name]
              [,TCPPORT={port_number_1,port_number_2,...}]
              [,TCPVIPA=virtual_ip_address]
              [,TIMEFORM={'hh:mm:ss'|'format_string'}]
              [,TPFDF={NO|(YES[,5000|ids,STATIC|(DYNAMIC,NOTRACE|TRACE,TSTAMP|
ASTAMP])}]
              [,TRACE=({320|t1],[10|t2],[ACTIVE|INACTIVE])]
              [,USRTSK1={NO|YES}]
              [,VFABUF=(v1,v2,v3,...)]
              [,WAS={NO|YES|(YES,1|threads,5|time-out)}]
```

Where:

label
Any valid assembler label.

DATE=(*dd*,*mmm*,*yyyy*)
Initial value for the ALCS system date. The date (local time) that ALCS uses when it starts executing for the first time. Where:

dd
Day of month, 1 or 2 decimal digits. The minimum value is 1, the maximum depends on the month and year.

mmm
Month, one of: `JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV,` or `DEC`.

yyyy
Year, 4 decimal digits. The minimum value is 1967.

For example, for the 13th January 1990 specify `DATE=(13,JAN,1990)`.

Note: The `ZATIM` command can only alter the ALCS system date to a **later** date. If you are in doubt, specify an early date for the initial value.

ALCS Operation and Maintenance describes the `ZATIM` command.

TIME=(*hhmm*[,[+|-]*ggnn*])
Initial value for the ALCS system time. The time that ALCS uses when it starts executing for the first time. Where:

hhmm
Initial value of the ALCS local time; 4 decimal digits representing hours and minutes in 24-hour clock format. The minimum value is 0000, the maximum is 2359.

ggnn
Difference between Greenwich Mean Time (GMT) and ALCS local time; 4 decimal digits representing hours and minutes in 24-hour clock format. Specify a positive value, optionally with a preceding plus sign (+), if local time is fast of GMT. Precede the value with a minus sign (-), if local time is slow of GMT. Specify `0000` if local time is the same as GMT.

For example, for 1 hour after midday local time and midday GMT, specify `TIME=(1300,0100)`; for 1 hour after midnight local time and 3 hours after midnight GMT, specify `TIME=(0100,-0200)`.

This time does not need to be exactly correct because the `ZATIM` command (described in *ALCS Operation and Maintenance*) can alter the ALCS time.

MAXENT=*maxent*
Maximum number of entries that can run concurrently. The minimum value is 5. The generation calculates the default from the `MSGS` parameter.

MSGS=*msgs*
Maximum expected number of input messages per second. If the `MAXENT` parameter is omitted, the generation uses this value to calculate the maximum number of entries that can run concurrently. If `MAXENT` is specified, the generation ignores `MSGS`.

NBRSU=({*num_type_1*],[0|*num_type_2*],[0|*num_type_3*])
Total number of *type_1_size*, *type_2_size*, and *type_3_size* storage units. Use the **SUSIZE**= parameter to specify the storage unit sizes.

If *num_type_1* is omitted, the generation calculates the default value from the `WSBREQ` parameter. The minimum value is 10. The workstation debug tool uses a large number of L2 storage units.

WSBREQ=(*b1,b2,b3,...*)
Mean (average) number of size L1, size L2, size L3, ... storage blocks that an entry uses at any one time. The number of subparameters must match the number of subparameters on the `CISIZE` and `RECSIZE` parameters of the `ALCS` macro. The minimum value for each subparameter is 1. If `NBRSU` is omitted, the generation uses `WSBREQ`, together with the maximum number of entries that can run concurrently, to calculate the number of storage units. If `NBRSU` is specified, the generation ignores `WSBREQ`.

Optional parameters

ACV=({*av1*],[*av2*],[2|*av3*],[5|*av4*],[*av5*],[80|*av6*])
Control values for ALCS online monitor task scheduling - the activity control variables:

av1
ALCS services the input list only if the number of active entries is less than, or equal to, *av1*. The minimum value is 1. The maximum value is 1 less than the maximum number of entries that can run concurrently in the system. The default is 80% of the maximum number of entries that can run concurrently.

av2
Controls create-type macros (`CREMC, CREDC, CRETC`). ALCS creates a new entry only if the total number of entries (active entries and entries on the defer and delay lists) is less than, or equal to, *av2*. If the total number of entries is greater than *av2*, ALCS queues the entry for retry; eventually the number of entries reduces to *av2* and the entry can execute the create type macro. The minimum value is 1, the maximum value is 1 less than the maximum number of entries that can run concurrently in the system. The default is 80% of the maximum number of entries that can run concurrently.

2|*av3*
Controls input list service, create type macros, and issuing VTAM RECEIVEs. If the number of available storage units is less than *av3*, the monitor does not service the input list or create type macros, and it stops issuing VTAM RECEIVEs. The minimum value is 1. The maximum value is 1 less than the total number of storage units defined.

5|*av4*
Controls use of IOCBs. If the number of available IOCBs is less than *av4*, the monitor does not activate some functions that use IOCBs. Note some functions that use IOCBs do not check against *av4*. The minimum value is 1. The maximum value is 1 less than the total number of IOCBs.

av5
Controls the number of entries that Recoup can create. Recoup will stop creating new entries when the number of entries that it has already created is equal to *av5*. Recoup also uses LODIC ECBCREATE (described in the *ALCS Application Programming Reference - Assembler*) to regulate the number of entries it creates, but *av5* can be used to provide a stricter control over the number of concurrent Recoup entries.

Some users find that a stricter control is needed to reduce the number of created entries to below the number of actuators they use for the ALCS database. This means that the amount of queuing in the system for devices (IOSQ time) is reduced and the response time of end user entries is improved during Recoup. The default is 80 per cent of the maximum number of entries that can run concurrently.

80|*av6*
Controls batch type processing, Recoup, file maintenance, and other transactions that use LODIC ECBCREATE (described in the *ALCS Application Programming Reference - Assembler*) to regulate the number of entries that they create. These transactions stop creating new entries when the number of entries already existing gets to *av6* percent of *av1* or *av2* (whichever is the smaller). The minimum value is 1, the maximum is 99.

AMODE31=(FORCE,{GLOBAL|NOTGLOBAL})
Force all application programs to run in 31-bit addressing mode, and allow storage units to be located above the 16MB boundary. GLOBAL further allows global areas 1 and 2 to be located above the 16MB boundary.

By default, ALCS allows application programs to run in 24-bit or 31-bit addressing mode, in order to allow ALCS/VSE and TPF programs to run. However, this makes less efficient use of storage, by requiring entry storage and the application global areas 1 and 2 to be located below the 16MB boundary, the limit of a 24-bit address.

Applications written for use with ALCS, and almost all application programs written for use with ALCS/VSE or TPF, normally work equally well in either mode. Even when migrating thousands of application programs from ALCS/VSE or TPF, it is worth identifying those few programs that do not work in 31-bit addressing mode and changing them so that they do. As soon as you can do so, you are strongly recommended to specify AMODE31=FORCE, which allows entry storage and all three global areas to be located anywhere in storage.

A few ALCS/VSE or TPF programs, though able to run in either mode, may contain code that depends on global area addresses being 24-bits. While you are still changing these programs, you can specify AMODE31=(FORCE,NOTGLOBAL) in order to keep global areas 1 and 2 below the 16MB boundary.

AMODE64=({[{VFA|LPSVFA}][,MSGTRACE])
This parameter controls whether VFA buffers or the online message trace area are above the bar or below the bar (below the bar is the default).

If you specify AMODE64=VFA or AMODE64=LPSVFA, ALCS obtains page-fixed virtual storage needed for the VFA buffers above the bar by using the z/OS 64-bit virtual storage support.

If you specify AMODE64=(VFA,MSGTRACE) or AMODE64=(LPSVFA,MSGTRACE), ALCS obtains virtual storage needed for the VFA buffers and the online message trace area above the bar by using the z/OS 64-bit virtual storage support.

4K page frames are used, when you specify AMODE64=VFA. However, ALCS tries to use 1MB page frames, when you specify AMODE64=LPSVFA. Note that when you specify LPSVFA, you also must code the LFAREA parameter in the MVS PARMLIB IEASYSxx member.

If your VFA buffers or online message trace area are above the bar, then you need to specify a z/OS MEMLIMIT for ALCS. The default of MEMLIMIT is 0, which means no storage above the bar can be allocated.

APPC={ [NO] | YES |(YES,1000|*num_srbs*)}
Enable support for APPC where:

NO
Support for APPC is not enabled. Applications cannot issue APPC, CPI-C or TPPCC calls

YES
Support for APPC is enabled. Applications can issue APPC, CP-C or TPPCC calls.

num_srbs
The number of SRBs to be generated. The default number of SRBs is 1000, the minimum is 100 and the maximum is 5000

CRET={128|*cret*}
Number of entries in the ALCS CRET table. The minimum value is 1, the maximum is 32 768. ALCS uses one CRET table entry for each outstanding CRETC request. A CRETC request is outstanding from the time that an application program issues the CRETC monitor-request macro until ALCS creates the new entry (see the *ALCS Application Programming Reference - Assembler*).

CTLERR=(*options*)
Describes the dump error options for CTL-type dumps, where *options* is one or more of the following, separated by commas:

NODUPE|DUPE
The system error routines produce (DUPE) or do not produce (NODUPE) duplicate error dumps. Duplicate errors are those where the same error condition occurs at the same displacement in the same application program. For NODUPE, the system error routines produce a dump only for the first of these errors.

TABLES|NOTABLES
System error dumps include (TABLES) or do not include (NOTABLES) ALCS monitor tables.

GLOBAL|NOGLOBAL
System error dumps include (GLOBAL) or do not include (NOGLOBAL) the global area.

SELBLKS|ALLBLKS
System error dumps include all (ALLBLKS) or selected (SELBLKS) entry storage. For SELBLKS, system error dumps include only the entry storage (ECB and associated storage blocks) associated with the entry that causes the dump. If there is no entry associated with a dump, the system error dump includes all entry storage.

MSG|NOMSG|DMPMSG
When a system error occurs, send a system error message (MSG) or do not send a system error message (NOMSG) to RO CRAS.

If DMPMSG is selected a system error message is not sent to RO CRAS for duplicate system errors.

NOVFA|VFA
System error dumps include (VFA) or do not include (NOVFA) VFA contents (VFA buffers, VFA buffer headers, VFA record locator table, and VFA control area).

DATEFORM={'DD/MM/YY'|'*format_string*'}
Installation default date format.

format_string is the character string that defines the format for the installation default date. The date format defined in the *format_string* is used:

- to service the TIMEC monitor-request macro when it has the format TIMEC DISPLAY,DATE,FORMAT=INSTALLATION.
- in ALCS command responses that include the date
- In the screen format on 3270 displays. Note that the maximum length of the date shown at the bottom of the screen is either 22 characters (when SHOWCRN=NO) or 11 characters (when SHOWCRN=YES).

When the installation default date is required for any of the above (for example, for the 3270 screen format) the USRDFMT installation-wide monitor exit is activated. The contents of the *format_string*

are passed as a parameter to the USRDFMT exit which can construct the installation default date and return it to ALCS (for example, for the date in the 3270 screen format). If you do not implement the USRDFMT exit, ALCS will construct the installation default date from the contents of the *format_string*.

There are two formats that can be used in the *format_string*. If you have implemented the USRDFMT installation-wide exit you should use the first format. If you have not implemented USRDFMT, you should use the second format. The two formats are:

- Format one - a mixture of characters and user tokens

  Characters and user tokens which provide your representation of the installation default date. The characters can be delimiters, punctuation or any other text. The user tokens will represent date values (for example, a token of ZZZZ could be your representation of the year value). The *format_string* is passed to the USRDFMT installation-wide monitor exit which must construct the installation default date using the characters and user tokens in the *format_string*.

- Format two - a mixture of characters and ALCS tokens

  Characters and ALCS tokens which provide your representation of the installation default date. The characters can be delimiters, punctuation or any other text. The ALCS tokens (as listed below) are replaced with the corresponding date value when the installation default date is constructed by ALCS. The sequence in which you define the ALCS tokens in the *format_string* is important. If the day number is required at the start of the date and the year number required at the end (with a month name in the middle), define the ALCS tokens in the *format_string* in that sequence. For example, if you code the *format_string* for `DATEFORM` as `'DD LLLLLLLLL YYYY'` the installation default date will be:

  – a 2-digit decimal day number, followed by a blank
  – an alphabetic month name up to a maximum of 9 characters, followed by a blank
  – a 4-digit decimal year number

  This would provide a date in September in the format 22 September 1997 (a date in June could be 07 June 1997). If you code the *format_string* for `DATEFORM` as `'MM/DD/YY'` the installation default date for 22 September 1997 will be 09/22/97.

The *format_string* can contain any of the following ALCS tokens:

YYYY
ALCS replaces this token with the 4-digit decimal year. For example, if the date is 7 March 1997, ALCS replaces 'YYYY' with '1997'.

YY
ALCS replaces this token with the 2-digit decimal year. For example, if the date is 7 March 1997, ALCS replaces 'YY' with '97'.

JJJ
ALCS replaces this token with the 3-digit decimal day-in-year (sometimes called the "Julian date"). For example, if the date is 7 March 1997, ALCS replaces 'JJJ' with '066'.

MM
ALCS replaces this token with the 2-digit decimal month. For example, if the date is 7 March 1997, ALCS replaces 'MM' with '03'.

SSS...
ALCS replaces this token with the abbreviated month name (as supplied in the `SCTGEN MONTHABBR` parameter). The number of 'S' characters in the token must be at least 3, and should be at least equal to the longest abbreviated month name. If the number of 'S's in the token is less than the abbreviated month name, ALCS truncates the abbreviated month name. If the number of 'S's in the token is more than the abbreviated month name, ALCS removes the extra 'S's. For example, if the date is 7 March 1997, and the `MONTHABBR` for March is three characters, ALCS replaces 'SSSS' with 'Mar'.

LLL...
ALCS replaces this token with the full month name (as supplied in the `SCTGEN MONTHNAME` parameter). The number of 'L' characters in the token must be at least 3, and should be at least

equal to the longest full month name. If the number of 'L's in the token is less than the month name, ALCS truncates the month name. If the number of 'L's in the token is more than the month name, ALCS removes the extra 'L's. For example, if the date is 7 March 1997, ALCS replaces 'LLLLLLLLL' with 'March'.

DD
ALCS replaces this token with the 2-digit decimal day-in-month. For example, if the date is 7 March 1997, ALCS replaces 'DD' with '07'.

DB2={NO|(YES[,*threads*])}
Whether this ALCS supports DB2. Where:

NO
Application programs cannot issue SQL calls.

YES
Application programs can issue SQL calls. ALCS allows up to *threads* concurrent database access threads. The default for *threads* is 1, the maximum is 2000.

The value of *threads* must not exceed the value specified in the IDBACK parameter of the DSN6SYSP macro for your DB2 installation. For information about installing DB2, refer to *DB2 for z/OS Installation Guide*.

DB2RECONNECT={NO|YES}
If DB2 is not started when ALCS tries to connect to it, this parameter controls whether ALCS automatically tries to connect again when DB2 starts. Do (YES) or do not (NO) automatically connect when DB2 starts. The default is NO.

DB2SSNM=*ssnm*
The DB2 subsystem name for initial connection. If omitted, the ALCS operator must use the ZCSQL command to establish a connection to a specified DB2 subsystem before application programs can issue SQL calls.

DCLECBU={0|*usersize*}
Size of the ECB user data collection area. The default value is 0 (no area) and the maximum size is 4096 bytes.

Application programs use the DCLAC macro (described in the *ALCS Application Programming Reference - Assembler*) to store data in the ECB user data collection area.

DCLOPTS={ALL|({IO|WAIT| MSG|SLC|PGM|ECB|CPU}[,...])}
Status of data collection when ALCS starts. If this parameter is omitted the ALCS operator must use the ZDCLR command to start data collection. If you specify only one option then you may omit the parentheses.

Where:

ALL
All available statistics are collected; that is, information about:

Input/output operations
ALCS task waits
Input/output messages, excluding SLC
SLC traffic
Application program use
Each ECB, including CPU utilization

IO
Collect statistics about input/output operations.

WAIT
Collect statistics about ALCS task waits.

MSG
Collect statistics about input and output messages.

SLC
Collect statistics about SLC traffic.

PGM
Collect statistics about application program use.

ECB
Collect statistics about each ECB.

CPU
Include 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 or monitor-request macros or C functions. It does not include CPU time when the entry loses control (for example in DASD I/O, sequential file I/O, or communication I/O).

By selecting any option except CPU, the ECB collection option will also be automatically set.

DDTIME={0|*dd*|255}
Time interval, in seconds, for suppressing duplicate system error messages. Specify a decimal number in the range 1 to 254. Normally ALCS sends an SE-NODUMP message to RO CRAS and the ALCS diagnostic file when a duplicate system error occurs. Use this parameter to reduce the frequency of these messages. After sending an SE-NODUMP message for a particular system error, ALCS does not send another until the time interval has expired.

Specify 0 if you do not want to suppress any messages.
Specify 255 if you want to suppress all messages.

DIADUMP={NO|YES}
Do (YES) or do not (NO) write a system error dump to the ALCS diagnostic file (DIA) when ALCS terminates abnormally. The dump will override the current system error options and dump as if all options are selected.

DISPTIM={0,*dispatcher_timeout*}
Specifies the ALCS dispatcher timeout threshold in seconds whereby ALCS will terminate all active ECBs in the system after the ALCS input list is shut continuously for the threshold value. Specify 0 to indicate no threshold is to be used.

DYNTCB={NO|YES}
Use this parameter to specify whether your ALCS system runs with the ALCS dynamic TCB facility enabled.

NO
Do not enable the ALCS dynamic TCB facility. This is the default.

YES
Enable the ALCS dynamic TCB facility. Refer to *ALCS Operation and Maintenance* for information about the job control statements to run ALCS and the ZCTCB command.

EMAILDOMAIN=*domain_name*
ALCS e-mail domain name, 1 through 64 characters. You can use the ZMAIL SET command to set or change the domain name when ALCS is running.

EMAILMTA=*ip_address*|*domain_name*
IP address or domain name of the local message transfer agent (MTA) for outbound e-mail. IP address specified with dotted decimal notation (for example, 9.180.147.125). Domain name can be 1 through 255 characters. You can use the ZMAIL SET command to set or change this value when ALCS is running.

Note: When a domain name is specified, the first ZMAIL send command will invoke the conversion of the domain name to an IP address. This could cause a DXC8073E message.

EMAILPORT={25|*port_number*}
Port number of the local message transfer agent (MTA) for outbound e-mail. Port number 25 is the well known port number for the simple message transfer protocol (SMTP). You can use the ZMAIL SET command to set or change this value when ALCS is running.

EMAILTIMEOUTR={30|*t1*}
Connection timeout in seconds for inbound e-mail. The minimum value is 1 second, the maximum value is 256 seconds. You can use the ZMAIL SET command to set or change this value when ALCS is running.

EMAILTIMEOUTS={120|*t2*}
Connection timeout in seconds for outbound e-mail. The minimum value is 1 second, the maximum value is 256 seconds. You can use the ZMAIL SET command to set or change this value when ALCS is running.

EMAILPOSTMASTER={ROC|*crn*}
Destination for inbound e-mail messages addressed to 'Postmaster@*domain_name*', where *domain_name* is your ALCS mail domain name. You can use the ZMAIL SET command to set or change this when ALCS is running.

EMAILQTIME={0|*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. Specify 0 to indicate no timeout. The maximum value is 256 minutes. You can use the ZMAIL SET command to set or change this value when ALCS is running.

EMAILQCOUNT={0|*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. Specify 0 to indicate no threshold. The maximum value is 256. You can use the ZMAIL SET command to set or change this value when ALCS is running.

ENTERR=({0|*ee1*],[5|*ee2*])
System error limits. ALCS terminates any entry when its requests for system error dumps reaches either of these limits. Where:

[0|*ee1*]
Absolute maximum number of system error dumps for each entry. This limit applies to every entry; entries cannot reset this limit. Specify 0 to indicate no absolute maximum number of system error dumps.

Some entries can request large numbers of system error dumps. Either specify 0 or specify a value large enough for any entry up to the maximum of 65 535.

[5|*ee2*]
Default maximum number of system error dumps. This limit applies to every entry, but entries that request large numbers of system error dumps can use the SLIMC monitor-request macro to reset this limit. Specify 0 to indicate no default maximum number of system error dumps. The maximum value is that specified for *ee1*.

ENTHOLD=({16|*eh1*],[4|*eh2*])
Entry hold limits. ALCS terminates an entry when the number of records the entry holds (at one time) reaches one of the these limits. Where:

[16|*eh1*]
Absolute record-hold count limit. The limit applies to every entry. Entries cannot reset this limit. Specify a value large enough for any entry as follows:

- From 1 to 256
- No limit (set to 0).

[4|*eh2*]
Default record-hold count limit. The limit applies to every entry, but entries that need to hold a large number of records can use the SLIMC monitor-request macro to reset this limit.

Specify 0 to indicate no default record-hold count limit.

ENTLIFE=({0|*el1*],[60|*el2*])
Entry life limits in seconds; ALCS terminates any entry that exists for longer than either limit. Where:

[0|*el1*]
Absolute maximum entry life. This limit applies to every entry; entries cannot reset this limit. Specify 0 to indicate no absolute maximum life.

Some entries can have a large entry life. For example, Recoup entries can exist for several hours. Either specify 0 or specify a value large enough for any entry, up to the maximum of 2 147 483 647.

[60|*el2*]
Default maximum entry life. This limit applies to every entry, but entries that have a large entry life (for example, Recoup entries) can use the SLIMC monitor-request macro to reset this limit. Specify 0 to indicate no default maximum life. However this is highly undesirable, since it can expose ALCS to a build-up of "stuck" entries, which can cause deadlocks and lock-outs. The maximum value is that specified for *el1*.

ENTLT=({*elt1*],[*elt2*])
Entry long-term pool file dispense limits. ALCS terminates an entry when its request for long-term pool file dispenses (GETFC monitor-request macros) reaches either of these limits. Where:

[*elt1*]
Absolute long-term pool file dispense limit. This limit applies to every entry; entries cannot reset this limit. A value of 0 indicates no absolute long-term pool file dispense limit. Do not use this value as it allows an entry to cause total depletion of long-term pool.

Some entries can require a large number of long-term pool file dispenses. Rather than specifying 0, specify a value large enough for any entry - up to the maximum of 2 147 483 647.

If you omit *elt1*, ALCS uses the value specified or defaulted for *eg1* on the ENTPOOL parameter.

[*elt2*]
Default long-term pool file dispense limit. This limit applies to every entry, but entries that require a large number of long-term pool file dispenses can use the SLIMC monitor-request macro to reset this limit. A value of 0 indicates no default long-term pool file dispense limit, which is highly undesirable. The maximum value is the same as the value specified or defaulted on *elt1*.

If you omit *elt2*, ALCS uses the value specified or defaulted for *eg2* on the ENTPOOL parameter.

ENTPOOL=({255|*eg1*],[255|*eg2*])
Entry pool file dispense limit defaults for ENTST and ENTLT. See the ENTST and ENTLT parameters. The workstation debugger uses a large number of short term pool records.

ENTREAD=({0|*er1*],[0|*er2*])
Entry read thresholds. ALCS calls the USRWAIT installation-wide monitor exit during an implied wait operation (for example WAITC, FINWC, FIPWC, or FIWHC) when the number of reads for this entry exceeds one of these thresholds (you can use this exit to indicate that the entry should be terminated, for example). Where:

[0|*er1*]
Absolute entry read threshold. This threshold applies to every entry. Entries cannot reset this threshold. Specify a threshold large enough for any entry as follows:

- From 1K to 65 535K reads (where K is 1024).
- No threshold (set to 0). This is the default value.

[0|*er2*]
Default entry read threshold. This threshold applies to every entry, but entries that need to read a large number of reads can use the SLIMC monitor-request macro to reset this threshold. Specify:

- From 1K to 65 535K reads (where K is 1024).
- No threshold (set to 0). This is the default value.

ENTST=({*est1*],[*est2*])
Entry short-term pool file dispense limits. ALCS terminates an entry when its request for short-term pool file dispenses (GETFC monitor-request macros) reaches either of these limits. Where:

[*est1*]
Absolute short-term pool file dispense limit. This limit applies to every entry; entries cannot reset this limit. A value of 0 indicates no absolute short-term pool file dispense limit. Do not use this value as it allows an entry to cause premature redispense of short-term pool records.

Some entries can require a large number of short-term pool file dispenses. Rather than specifying 0, specify a value large enough for any entry - up to the maximum of 2 147 483 647.

If you omit *est1*, ALCS uses the value specified or defaulted for *eg1* on the ENTPOOL parameter.

[*est2*]
Default short-term pool file dispense limit. This limit applies to every entry, but entries that require a large number of short-term pool dispenses can use the SLIMC monitor-request macro to reset this limit. A value of 0 indicates no default short-term pool file dispense limit, which is highly undesirable. The maximum value is the same as the value specified or defaulted on *est1*.

If you omit *est2*, ALCS uses the value specified or defaulted for *eg2* on the ENTPOOL parameter.

ENTSTOR=({<u>256</u>|*es1*],[<u>64</u>|*es2*])
Maximum entry storage limits in units of 1KB (1024 bytes). ALCS terminates any entry that requests more storage than either limit. Specify these values to include the following:

- 2KB for system overheads, 4KB for the ECB, estimated storage block requirements, estimated DECB requirements, estimated heap storage that does fit in an entry storage block rounded up to a multiple of the *type_1_size* SU size;
- a minimum of 512KB for high-level-language (HLL) applications if using LE support or a minimum of 2048KB for HLL applications if using LE support and the remote debugger, rounded up to a multiple of the *type_2_size* SU size;
- estimated heap storage requirements that do not fit in entry storage blocks rounded up to a multiple of the *type_3_size* SU size;
- a proportion (say 50%) for fragmentation.

Where:

[<u>256</u>|*es1*]
Absolute maximum storage per entry. This limit applies to every entry; entries cannot reset this limit. Specify a value large enough for any entry, up to the maximum of 65 535 with a minimum of 64.

[<u>64</u>|*es2*]
Default maximum storage per entry. This limit applies to every entry, but entries that have a large storage requirement can use the SLIMC monitor-request macro to reset this limit. The minimum value is 64. The maximum is that specified or defaulted for *es1*.

IBM recommends that you do not use ENTSTOR. Instead IBM recommends that you use ENTSTOR1, ENTSTOR2, ENTSTOR3.

Note that **ENTSTOR**=(*aa*,*bb*) is equivalent to
ENTSTOR1=(*aa*,*bb*),**ENTSTOR2**=(*aa*,*bb*),**ENTSTOR3**=(*aa*,*bb*)

ENTSTOR1=({<u>256</u>|*esa1*],[<u>64</u>|*esa2*])
Maximum entry storage limits (not including stack storage) in units of 1KB (1024 bytes). ALCS terminates any entry that requests more storage than either limit. Specify these values to include the following:

- 2KB for system overheads;
- 4KB for the ECB;
- estimated storage block requirements;
- estimated DECB requirements;
- estimated heap storage that does fit in an entry storage block;
- a proportion (say 50%) for fragmentation;

- rounded up to a multiple of the *type_1_size* SU size.

Where:

[256|*esa1*]
Absolute maximum storage per entry. This limit applies to every entry; entries cannot reset this limit. Specify a value large enough for any entry, up to the maximum of 65 535 with a minimum of 64.

[64|*esa2*]
Default maximum storage per entry. This limit applies to every entry, but entries that have a large storage requirement can use the SLIMC monitor-request macro to reset this limit. The minimum value is 64. The maximum is that specified or defaulted for *esa1*.

ENTSTOR2=({0|*esb1*],[0|*esb2*])
Maximum entry storage limits (high-level language stack storage) in units of 1KB (1024 bytes). ALCS terminates any entry that requests more storage than either limit. Specify these values to include the following:

- a minimum of 512KB for high-level-language (HLL) applications if using LE support or a minimum of 2048KB for HLL applications if using LE support and the remote debugger;
- a proportion (say 50%) for fragmentation;
- rounded up to a multiple of the *type_2_size* SU size.

Where:

[0|*esb1*]
Absolute maximum storage per entry. This limit applies to every entry; entries cannot reset this limit. Specify a value large enough for any entry, up to the maximum of 65 535 with a minimum of 0 (no limit).

[64|*esb2*]
Default maximum storage per entry. This limit applies to every entry, but entries that have a large storage requirement can use the SLIMC monitor-request macro to reset this limit. The minimum value is 0 (no limit). The maximum is that specified or defaulted for *esb1*.

ENTSTOR3=({0|*esc1*],[0|*esc2*])
Maximum entry storage limits (heap storage) in units of 1KB (1024 bytes). ALCS terminates any entry that requests more storage than either limit. Specify these values to include the following:

- estimated heap storage requirements that do not fit in any entry storage block;
- a proportion (say 50%) for fragmentation;
- rounded up to a multiple of the *type_3_size* SU size.

Where:

[0|*esc1*]
Absolute maximum storage per entry. This limit applies to every entry; entries cannot reset this limit. Specify a value large enough for any entry, up to the maximum of 65 535 with a minimum of 0 (no limit).

[0|*esc2*]
Default maximum storage per entry. This limit applies to every entry, but entries that have a large storage requirement can use the SLIMC monitor-request macro to reset this limit. The minimum value is 0 (no limit). The maximum is that specified or defaulted for *esc1*.

ENTWRT=({20|*ew1*],[5|*ew2*])

[20|*ew1*]
Maximum number of write operations (for example FILEC or FILNC) that an entry can request before it must defer, up to the maximum of 65 535. If an entry requests *ew1* writes without including a DLAYC or DEFRC macro, ALCS forces the entry to defer. That is, it adds the entry to the defer list during macro service for the *ew1*th write monitor-request macro. This gives a time delay (approximately 0.0125 seconds) that allows the writes to complete.

If the entry continues to request writes, ALCS forces it to defer again after the next *ew1* writes, and so on.

An entry can prevent these forced defers by issuing DLAYC or DEFRC more often than every *ew1* writes.

[5|*ew2*]
Control the function of the DLAYC and DEFRC monitor-request macros. Entries that request large numbers of write operations can often use the DLAYC or DEFRC monitor-request macros so that they do not exceed the limit *ew1*. The *ew2*th DLAYC or DEFRC monitor-request macro includes a time delay (approximately 0.0125 seconds) to allow the writes to complete. This process repeats so that there is a time delay after every *ew2* DLAYC or DEFRC monitor-request macro that an entry requests. The maximum is that specified for *ew1*.

While a record or resource is held, issuing DLAYC or DEFRC does not reset this limit.

GLBLPROT={NO|YES}
Storage protection for the global area. You may require ALCS to provide storage protection for global areas 1 and 3.

NO
Use the entry storage key for global areas 1 and 3.

YES
Do not use the entry storage key for global areas 1 and 3, but use a storage key that provides storage protection. Assembler application programs that modify data in global areas 1 and 3 must change their PSW protect key before modifying the data, and, when they have completed their data modifications, restore their PSW protect key. Applications use the GLMOD, KEYCC, KEYRC and FILKW macros for this (see the *22apra.*).

GLBLSZE=({21|*s1*],[25|*s2*],[12|*s3*])
The size, in units of 1KB (1024 bytes), of the three parts of the application global area (area 1, area 2, and area 3). The minimum values are as follows:

- 4 for *s1*
- 1 for *s2*
- 12 for *s3*

The maximum value for each of *s1*, *s2*, and *s3* is 4000.

Determine the size of each part of the application global area from the information printed by the G01G0 macros in the global load control programs (GOA0, GOA1, ..., and CGAF). Remember to allow space for the global directories (including 512 bytes for the directory reserved for IBM use, global directory 15). In addition allow an overhead of 256 bytes for each directory, and a further 256 bytes for the directory of directories.

HIPER=((*blks*[,CASTOUT|NOCASTOUT]),...)]
Specify the number of 4KB blocks for each record size that you want to cache in Hiperspace. Specify the record sizes in sequence L1 through L8. Use a comma for each record size that you do not want to cache in Hiperspace, where:

blks
Number of records of this record type to be created.

CASTOUT
Create a Hiperspace with CASTOUT=YES. This means that MVS can reclaim some pages when the system load warrants this.

NOCASTOUT
Create a Hiperspace with CASTOUT=NO. This means that MVS will reclaim pages less frequently. In some cases this will result in adverse performance levels.

The allocation requested in this generation could be changed by an MVS installation-wide exit. Check that there are no MVS installation-wide exits capable of doing this. For example, to allocate 8MB of Hiperspace for L2 records and none for any other record size, and to allow castout, code:

```
HIPER=(,(2000))
```

HOLD={128|*hold*}
The number of entries in the ALCS resource hold table. Application programs use the CORHC, ENQC, and EVNTC monitor-request macros to hold resources, and CORUC, DEQC, and POSTC to unhold resources. The *ALCS Application Programming Reference - Assembler* describes these monitor-request macros. ALCS uses one resource hold table entry for each resource that is held. Allocate sufficient resource hold table entries so that there are always at least half the entries unused. The minimum value is 1, the maximum is 32 767.

HTTP={NO|YES|(YES,6|*connections,requests*|6)}
Enable support for the HTTP Client:

NO
Support for the HTTP Client is not enabled. Applications cannot issue HTTP Client calls using the z/OS client web enablement toolkit.

YES|(YES,6|*connections*)
Support for the HTTP Client is enabled. Applications can send HTTP requests to and receive HTTP responses from a remote HTTP Server. Support is for up to *connections* concurrent connections to remote HTTP servers. The default for *connections* is 6, the minimum is 2 and the maximum is 100.

YES|(YES,6|*connections, requests*|6)
Support for the HTTP Client is enabled. Applications can send HTTP requests to and receive HTTP responses from a remote HTTP Server. Support is for up to *connections* concurrent connections to remote HTTP servers. The default for *connections* is 6, the minimum is 2 and the maximum is 100. Support is for up to requests concurrent HTTP requests. The default for requests is 6, the minimum is 2 and the maximum is 500. There must be at least as many requests as connections.

JSON={NO|YES|(YES,10*instances*)}

NO
Support for the JSON Parser is not enabled. Applications cannot use the z/OS JSON Parser.

YES|(YES,10|*instances*)
Support for the JSON Parser is enabled. Applications can parse, validate, and create JSON text. Support is for up to *instances* parser sessions. The default for *instances* is 10, the minimum is 10 and the maximum is 200.

ICSF={NO|YES}
Enable support for ICSF where:

NO
Support for ICSF is not enabled. Applications cannot encrypt or decrypt data using the ICSF interface.

YES
Support for ICSF is enabled. Applications can encrypt or decrypt data using the ICSF interface. One SUBTASK will be created for ICSF calls.

LTPOOL={INUSE|FREE}
Initial status of all long-term pool file records is available (FREE), or not available (INUSE). See Chapter 9, “Long-term pool space recovery - Recoup,” on page 480. **LTPOOL**=INUSE requires a run of Recoup before the system can dispense any long-term pool file records.

MEMBER=*name*
Name of the configuration dependent table that describes the ALCS system configuration. The stage 2 link-edit step uses this name for the system configuration table load module.

The default name is DXCCDSiv, where *i* and *v* are the identifier and version specified on the ALCS macro.

MIGSEQ={NO|YES}

Where:

NO

Do not allow ALCS to wait for DFHSM to recall general sequential files that have been migrated. This is the default.

YES

Allow ALCS to wait for DFHSM to recall general sequential files that have been migrated. You must select which general sequential files are handled this way by using the installation-wide monitor exit USRSEQ1. For more information about this exit, see "Migrated sequential file exit - USRSEQ1" on page 285.

MONTHABBR={(Jan, ...Dec)|(*abbr_name*,...)}

Abbreviated month names used in ALCS displays (that include abbreviated month name) and used in dates provided by the TIMEC monitor-request macro. Each abbreviated month name is a character string that can be mixed case, up to a maximum of 20 characters, but must not contain space (X'40') characters. Each abbreviated month name must be separated by a comma. You must provide the same number of abbreviated names as you provide (or default) in the MONTHNAME parameter.

If you omit this parameter, ALCS uses up to the first 3 characters of each month name that you specify (or default) in the MONTHNAME parameter. ALCS passes the abbreviated month names as a parameter to your USRDFMT installation-wide exit routine if installed. See "Date and time formatting - USRDFMT" on page 258.

MONTHNAME={(January, ...December)|(*month_name*,...)}

Month names used in ALCS displays (that include month names) and used in dates provided by the TIMEC monitor-request macro. Each month name is a character string that can be in mixed case, up to a maximum of 20 characters. It must not contain space (X'40') characters. Each month name is separated by a comma.

If you supply month names and have no installation-wide monitor exit routine for date formatting installed, (see "Date and time formatting - USRDFMT" on page 258) you must provide 12 month names. The first is the name for January, the second is the name for February, and so on.

If you have an installation-wide monitor exit routine for date formatting, then this string can contain any number of month names up to a maximum of 20. ALCS passes the month names as a parameter to your USRDFMT installation-wide exit routine.

If you omit this parameter, ALCS uses the 12 English language month names, with initial capital - 'January', 'February', and so on.

MQM={NO|(YES[,CONNECT|NOCONNECT])}

Support for WebSphere MQ for z/OS. Where:

NO

Application programs cannot issue MQI calls.

YES

Application programs can issue MQI calls.

CONNECT

ALCS connects to the WebSphere MQ for z/OS queue manager during startup.

NOCONNECT

ALCS does not connect to the WebSphere MQ for z/OS queue manager during startup. The ALCS operator can use the ZCMQI command to establish a connection with the WebSphere MQ for z/OS queue manager after ALCS is started.

MQMI={*initiation_queue*|(*initiation_queue*,*system_state*)}

ALCS initiation queue (also called the trigger queue) name, 1 to 48 alphanumeric characters. Omit this parameter if there is no initiation queue defined for ALCS on this WebSphere MQ for z/OS queue manager.

system_state is one of IDLE, CRAS, MESW, or NORM.

If MQM=(YES,CONNECT) and MQMI=(*initiation_queue*,*system_state*) are both specified, then ALCS attempts to open the initiation queue automatically:

- If *system_state* is omitted, then ALCS attempts to open the initiation queue when it connects to the WebSphere MQ for z/OS queue manager during startup.
- If *system_state* is specified, then ALCS attempts to open the initiation queue just before it reaches the specified system state. ALCS closes the initiation queue when it reaches a system state lower than this.

This parameter also defines the default initiation queue name for the ZCMQI command INITQ parameter.

See *ALCS Concepts and Facilities* for an overview on using WebSphere MQ for z/OS initiation queues with ALCS.

See the *WebSphere MQ for z/OS Application Programming Reference* for information about WebSphere MQ for z/OS initiation queues.

MQMM=*queue_manager_name*
ALCS queue manager name, 1 to 48 alphanumeric characters. If omitted, ALCS uses the default name established during the WebSphere MQ for z/OS installation and setup.

MQMQ={*input_queue*|(*input_queue*,*system_state*)}
ALCS input queue name, 1 to 48 alphanumeric characters. Omit this parameter if there is no input queue defined for ALCS on this WebSphere MQ for z/OS queue manager.

system_state is one of IDLE, CRAS, MESW, or NORM.

If MQM=(YES,CONNECT) and MQMQ=(*input_queue*,*system_state*) are both specified, then ALCS attempts to open the input queue automatically:

- If *system_state* is omitted, then ALCS attempts to open the input queue when it connects to the WebSphere MQ for z/OS queue manager during startup.
- If *system_state* is specified, then ALCS attempts to open the input queue just before it reaches the specified system state. And ALCS closes the input queue when it reaches a system state lower than this.

This parameter also defines the default input queue name for the ZCMQI command INPUTQ parameter.

See *ALCS Concepts and Facilities* for an overview on using WebSphere MQ for z/OS queues with ALCS.

MSGI=*logon_response*
Logon response message. ALCS displays this message on IBM 3270 display terminals when they log on to ALCS. Specify a maximum of 64 characters enclosed in single quotes. The default is ALCS*iv* `online` where *i* and *v* are the identifier and version as specified in the ALCS macro.

MSGTRACE={64K|*a*K|*b*M|NO}
The online message trace area size.

Specify the size in megabytes (M) or kilobytes (K). The smallest size is 64K, the largest size is 2048K (or 2M).

If you code MSGTRACE then you also must consider coding the appropriate AMODE64 and PAGE parameters. Note that if `AMODE64=(,MSGTRACE)` or `AMODE64=(VFA,MSGTRACE)` is coded then the size allocated (because memory objects are used) is in megabytes.

You indicate that there is no message trace area (and so there is no message tracing) by coding `MSGTRACE=NO` (or alternatively `MSGTRACE=0K` or`MSGTRACE=0M`).

NBRIOB=*nbriob*
Total number of I/O control blocks (IOCBs) in this ALCS system. The minimum value is 10. The generation calculates the default from the maximum number of entries that can run concurrently in the system.

NOWAIT={32|*n*}
Maximum number of implied wait operations (for example, WAITC, FINWC, FIPWC, or FIWHC) that an entry can request before it must delay, a decimal number from 32 to 256. If an entry issues *n* monitor-request macro calls with implied wait without encountering any outstanding I/O requests, and without including a DLAYC or DEFRC macro, ALCS forces the entry to delay. That is, it adds the entry to the delay list during the *n*th implied wait monitor-request macro, allowing other entries to do work.

If the entry continues to request implied wait operations without encountering any outstanding I/O requests, ALCS forces it to delay again after the next *n* implied wait monitor-request macros, and so on.

Note: If an entry issues 16 monitor-request macro calls with implied wait without encountering any outstanding I/O requests, and without including a DLAYC or DEFRC macro, ALCS forces the entry to lose control. That is, it adds the entry to the ready list during the 16th implied wait monitor-request macro. The NOWAIT parameter does not affect this mechanism.

OPRERR=(*options*)
Describes the dump error options for OPR-type dumps, where *options* is one or more of the following, separated by commas:

NODUPE|DUPE
The system error routines produce (DUPE) or do not produce (NODUPE) duplicate error dumps. Duplicate errors are those where the same error condition occurs at the same displacement in the same application program. For NODUPE, the system error routines produce a dump only for the first of these errors.

TABLES|NOTABLES
System error dumps include (TABLES) or do not include (NOTABLES) ALCS monitor tables.

GLOBAL|NOGLOBAL
System error dumps include (GLOBAL) or do not include (NOGLOBAL) the global area.

SELBLKS|ALLBLKS
System error dumps include all (ALLBLKS) or selected (SELBLKS) entry storage. For SELBLKS, system error dumps include only entry storage (ECB and associated storage blocks) associated with the entry that causes the dump. If there is no entry associated with a dump, a system error includes all entry storage.

MSG|NOMSG|DMPMSG
If a system error occurs, send a system error message (MSG) or do not send a system error message (NOMSG) to RO CRAS.

If DMPMSG is selected a system error message is not sent to RO CRAS for duplicate system errors.

NOVFA|VFA
System error dumps include (VFA) or do not include (NOVFA) VFA contents (VFA buffers, VFA buffer headers, VFA record locator table, and VFA control area).

PAGE={NONE|ALL|(*option*,...)}
Control ALCS page fixing. By default, ALCS page fixes storage that is critical to performance. This storage includes the application programs, entry storage (storage units), VFA buffers, message trace area, and the application global area. Note that communication tables are always pageable. Specify one of:

NONE
Do not allow paging; that is, page fix all storage that is critical to performance.

ALL
Allow paging of the VFA buffers, storage units, application programs, application global area, and online message trace area.

(*option*,...)
One or more of the following, separated by commas:

PROGRAM
Allow paging of application programs.

STORE*x*
Allow paging of storage units.

STORE1
Allow paging of type 1 storage units

STORE2
Allow paging of type 2 storage units

STORE3
Allow paging of type 3 storage units.

STORE
Allow paging of all storage units.

Note: IBM recommends that you use STORE1, STORE2, STORE3 instead of using STORE, which is available only for backward compatability.

GLOBAL
Allow paging of the application global area.

MSGTRACE
Allow paging of the online message trace area.

VFA
Allow paging of the VFA buffers.

The recommended use of PAGE is as follows:

- For a test system, specify PAGE=ALL.
- For a production system later than z/OS 1.3 or using WLM goal mode or a system without expanded storage such as an IBM zSeries, specify PAGE=NONE (or accept the default).
- For production systems based on older technology than above, specify PAGE=ALL and avoid paging by setting MVS storage isolation parameters for the ALCS performance group.

  Performance groups are defined in the SYS1.PARMLIB member IEAIPSxx (see the appropriate MVS *MVS Initialization and Tuning Manual* for detailed information on this subject).

  You can use PAGE=NONE when your production ALCS is the only job on the system.
- When you code AMODE64=VFA, then PAGE=VFA will be ignored as in this case the storage is always page-fixed.

PDULOGSTREAM='*logstream_prefix*'
Prefix for the MVS system logger log stream names that ALCS will use for the emergency pool recovery (PDU) function, 1-21 characters enclosed in quotes. ALCS constructs the actual log stream names as:

logstream_prefix.L*s*.*i*

Where L*s* is the corresponding long-term pool size and *i* is the ALCS system identifier.

If you include this parameter, ALCS will write released long-term pool file addresses to the appropriate log stream, and use these file addresses in case of pool depletion. If you omit this parameter, the ALCS emergency pool recovery function will not be available.

See *ALCS Concepts and Facilities* for a description of the emergency pool recovery function.

PERFMON={NO|YES}
Use this parameter to specify whether your ALCS system runs with the ALCS performance monitor enabled.

NO
Do not enable the ALCS performance monitor. This is the default.

YES
Enable the ALCS performance monitor.

POOLCTL=({20|*g1*],[30|*g2*])
Control values for ALCS type-1 short-term pool file management:

20|***g1***
Pool keypoint update count. ALCS keypoints (writes out to the real-time database) the short-term pool file directory record after dispensing *g1* records from that pool. The minimum value is 1; the maximum is 999.

30|***g2***
Pool restart skip count. When ALCS restarts after an unplanned shutdown, it skips (that is, it does not dispense) *g2* records from each short-term pool. The minimum value is 2; the maximum is 999.

The pool keypoint update count (*g1*) must be less than the pool restart skip count (*g2*).

RACF={YES|NO}
External security manager. Where:

YES
The external security manager for this ALCS is RACF. This is the default.

NO
The external security manager for this ALCS is not RACF.

Specifying RACF=YES means that the maximum length of installation-defined data that the AUTHC monitor-request macro can retrieve is 254 bytes. This is because of a SAF restriction on field ACEEINST in the ACEE control block.

SCRLLOG=({5000|*sl1*],[500|*sl2*])
ALCS scroll log mode limits. Where:

[5000|*sl1*]
Absolute maximum number of lines that can be saved in the scroll log. The minimum value is 100 and the maximum is 5 000.

[500|*sl2*]
Default maximum number of lines that can be saved in the scroll log. The minimum value is 100 and the maximum is 5 000.

Use the ZSCRL command to control the scroll log mode. The *ALCS Operation and Maintenance* describes the ALCS ZSCRL command.

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.

SHOWCRAS={NO|YES}
Do (YES) or do not (NO) show the CRAS status or authority on the last line (row) of IBM 3270 display terminals logged on to ALCS. When the terminal has both Prime and alternate CRAS status or authority, then CRAS=PRC is shown. The CRAS status or authority field follows the system name or user text field at the bottom of the screen and affects the maximum user text length.

- When SHOWCRAS=NO (the default) is set, the maximum length of the user text shown is 45 characters.
- When SHOWCRAS=YES is set, the maximum length of the user text shown is 32 characters.

SHOWCRN={NO|YES}
Do (YES) or do not (NO) display the CRN on the last line (row) of IBM 3270 display terminals logged on to ALCS. The CRN field follows the date field at the bottom of the screen and affects the maximum length of the date.

- When SHOWCRN=NO (the default) is set, the maximum length of the date shown is 22 characters.
- When SHOWCRN=YES is set, the maximum length of the date shown is 11 characters.

SUSIZE=({2|*type_1_size*],[5|*type_2_size*],[5|*type_3_size*])
Storage unit (SU) size index. Use *type_1_size* to specify the SU size for assembler applications.

Use *type_2_size* to specify the SU size for high-level-language (HLL) requirements. Specify a value of 9 when using the workstation debugger.

Use *type_3_size* to specify the SU size for heap storage requirements that do not fit in an entry storage block. Specify a value so that the SU size exceeds the largest possible heap storage request.

The following table shows the values for this parameter and the resulting storage unit size:

SU size index	SU size		SU size index	SU size
2	12KB		7	508KB
3	28KB		8	1020KB
4	60KB		9	2044KB
5	124KB		10	4092KB
6	252KB		11	8188KB

SVCDUMP={NO|YES}
Do (YES) or do not (NO) dump all the private storage address space along with system trace and summary data using the z/OS dump facility, when ALCS terminates abnormally.

SYSERR=(*options*)
System error options, where *options* is one or more of the following, separated by commas:

NODUPE|DUPE
The system error routines produce (DUPE) or do not produce (NODUPE) duplicate error dumps. Duplicate errors are those where the same error condition occurs at the same displacement in the same application program. For NODUPE, the system error routines produce a dump only for the first of these errors.

TABLES|NOTABLES
System error dumps include (TABLES) or do not include (NOTABLES) ALCS monitor tables and the application global area.

SELBLKS|ALLBLKS
System error dumps include all (ALLBLKS) or selected (SELBLKS) entry storage. For SELBLKS, system error dumps include only entry storage (ECB and associated storage blocks) associated with the entry that causes the dump. If there is no entry associated with a dump, the system error dump includes all entry storage.

MSG|NOMSG
If a system error occurs, the system error routines send (MSG) or do not send (NOMSG) a system error message to RO CRAS.

NOVFA|VFA
System error dumps include (VFA) or do not include (NOVFA) VFA buffers.

Note: The options in SYSERR apply to both the CTL-type and OPR-type dumps. If you have included either the CTLERR or OPRERR parameters then options coded on this parameter are ignored. You should normally use CTLERR and OPRERR in preference to SYSERR.

SYSNAME=*system-name*
ALCS system name. ALCS displays this system name on the last line (row) of IBM 3270 display terminals that are logged on to ALCS. It also includes this system name in some messages. Specify a maximum of 32 characters enclosed in single quotes. The default is ALCS*iv* where *i* and *v* are the identifier and version as specified in the ALCS macro.

TCPIP={NO|YES|(YES,1|*threads*)}
Enable support for TCP/IP where:

NO
TCP/IP communication resources can not be started. Application programs can not issue TCP/IP sockets calls.

YES|(YES,1|*threads*)
TCP/IP communication resources can be started. Application programs can issue TCP/IP sockets calls. Support is for up to *threads* concurrent sockets threads. The default for *threads* is 1, the maximum is 2000.

TCPLIST={NO|YES|(YES,1|*threads*)}
Support for TCP/IP concurrent server (Listener) where:

NO
No support for the concurrent server.

YES|(YES,1|*threads*)
Support for up to *threads* concurrent server sockets threads. The default for *threads* is 1, the maximum is 256. Up to eight concurrent servers can be started at the same time, but there is a single set of threads which is shared by all concurrent servers.

TCPNAME=*tcpip_address_space_name*
The TCP/IP address space name for initial connection. If omitted, the ALCS operator must use the ZCTCP command to establish a connection to a specified TCP/IP address space before application programs can issue TCP/IP sockets calls and TCP/IP communication resources can be started. Specify the name of the TCP/IP job or started task. 1 to 8 alphanumeric characters.

TCPPORT={*port_number_1*,*port_number_2*,...}
The TCP/IP port numbers which the concurrent server (Listener) will use. If omitted, the ALCS operator can use the ZCTCP command to specify the TCP/IP port numbers for the concurrent server. Each port number is a decimal number between 0 and 65 535.

TCPVIPA=*virtual_ip_address*
By default, ALCS issues BIND(INADDR_ANY) when it creates TCP/IP sockets for use by the concurrent server (Listener) and by TCP/IP communication resources. This results in Communication Server accepting incoming requests over all available network interfaces.

There are situations in which you might want ALCS to use BIND (specified IP Address) instead. Consider the case of an IP stack having been configured with two virtual IP addresses (VIPA). One VIPA address is returned by the name server when clients resolve one host name, and the other VIPA address is returned by the name server when clients resolve the other host name. In fact, both host names represent the same IP stack, but the host names can be used to represent two different ALCS systems on that MVS host.

This feature may be useful when ALCS is restarted on a different LPAR or different machine.

This feature may also be useful when two ALCS systems connect to the same IP stack.

You can use either the `DEVICE VIRTUAL` and `LINK VIRTUAL` statements (static VIPA) or you can use the `VIPADYNAMIC VIPADEFINE` statements (multiple application-instance dynamic VIPA).

TIMEDATEFORM={'hh:mm:ss DD/MM/YY'|'*format_string*'}
Installation default time and date format.

format_string
Character string that defines the format for the installation default time and date. The combined time and date defined in the *format_string* is used to service the TIMEC monitor-request macro when it has the format
`TIMEC DISPLAY,TIMEDATE,FORMAT=INSTALLATION`

See the explanation for *format_string* that accompanies the description of the `DATEFORM` and `TIMEFORM` parameters. If you code *format_string* for `TIMEDATEFORM` you should consider coding format strings for the `DATEFORM` and `TIMEFORM` parameters as well.

TIMEFORM={'hh:mm:ss'|'*format_string*'}
Installation default time format.

format_string
Character string that defines the format for the installation default time. The time format defined in the *format_string* is used to service the TIMEC monitor-request macro when it has the format `TIMEC DISPLAY,TIME,FORMAT=INSTALLATION`.

When the installation default time is required, the USRDFMT installation-wide monitor exit is activated. The contents of the *format_string* are passed as a parameter to the USRDFMT exit which can construct the installation default time and return it to ALCS. If you do not implement the USRDFMT exit, ALCS will construct the installation default time from the contents of the *format_string*.

There are two formats that can be used in the *format_string*. If you have implemented the USRDFMT installation-wide exit you should use the first format. If you have not implemented USRDFMT, you should use the second format. The two formats are:

- Format one - a mixture of characters and user tokens

  Characters and user tokens which provide your representation of the installation default time. The characters can be delimiters, punctuation or any other text. The user tokens will represent time values (for example, a token of NN could be your representation of the minute value). The *format_string* is passed to the USRDFMT installation-wide monitor exit which must construct the installation default time using the characters and user tokens in the *format_string*.

- Format two - a mixture of characters and ALCS tokens

  Characters and ALCS tokens which provide your representation of the installation default time. The characters can be delimiters, punctuation or any other text. The ALCS tokens (as listed below) are replaced with the corresponding time value when the installation default time is constructed by ALCS. The sequence in which you define the ALCS tokens in the *format_string* is important. If the time is to be in the sequence of hours, minutes, seconds, define the ALCS tokens in the *format_string* in that sequence. For example, if you code the *format_string* for `TIMEFORM` as `'hh mm ss'` the installation default time will be:

  – a 2-digit decimal hour number, followed by a blank
  – a 2-digit decimal minutes number, followed by a blank
  – a 2-digit decimal seconds number

The *format_string* can contain any of the following ALCS tokens:

hh
ALCS replaces this token with the 2-digit decimal hour (24-hour clock format). For example, if the time is 16:20:30, ALCS replaces 'hh' with '16'.

mm
ALCS replaces this token with the 2-digit minute. For example, if the time is 16:20:30, ALCS replaces 'mm' with '20'.

ss
ALCS replaces this token with the 2-digit second. For example, if the time is 16:20:30, ALCS replaces 'ss' with '30'.

am
Time of day before or after 12:00h. ALCS replaces this token with either 'am' or 'pm' depending on whether the time is before midday or after. For example, if this token is used and the time is 16:20:30, ALCS displays the time as 04:20:30pm.

TPFDF={NO|(YES[,5000|*ids*,STATIC|(DYNAMIC,NOTRACE|TRACE,TSTAMP|ASTAMP])}
Specify that TPFDF is installed, set the number of record identifiers for TPFDF data collection, and set TPFDF options. Set the appropriate TPFDF options for your ALCS system. If you select the DYNAMIC and TRACE options, they provide additional TPFDF functionality, and also invoke additional processing for each TPFDF macro call.

Where:

NO
TPFDF is not installed.

YES
TPFDF is installed.

ids
Maximum number of IDs collectable by TPFDF data collection. The default is 5000.

STATIC
Calculate the storage address of each TPFDF program during ALCS restart and place each address in the TPFDF static fastlink table.

If you use the ALCS ZPCTL command to load new versions of TPFDF programs, these programs are NOT invoked by TPFDF macro calls.

DYNAMIC
Create a TPFDF dynamic fastlink table for each entry that issues TPFDF macro calls. The storage address of a TPFDF program is placed in the table on the first invocation of that program (by a TPFDF macro call issued by that entry).

If you use the ALCS ZPCTL command to load new versions of TPFDF programs, those programs are invoked by subsequent TPFDF macro calls. When this option is active, additional processing is performed for each TPFDF macro call.

NOTRACE
Do not trace TPFDF macro calls in application programs.

TRACE
Trace TPFDF macro calls in application programs. Use this option when you require TPFDF macro trace data in ALCS system error dumps (in the entry macro trace) and during conversational and diagnostic trace. When this option is active, additional processing is performed for each TPFDF macro call.

You can control TPFDF trace online.

See the description of the ZTRAC CEP command in *ALCS Operation and Maintenance*.

ASTAMP
When the ASTAMP parameter is specified in your system configuration, then the name of the application program which requested TPFDF to find, file, dispense or release the record is used as program stamp for the physical record (last file, dispense, and release stamps), for the ST Events log (file, fine, dispense, and release events), and for the pool file error messages. (Physical records program stamps and ST event logs can be displayed with the help of ZDFIL I. Pool file error messages are logged on the diagnostic file and are printed by the ALCS diagnostic file processor.)

TSTAMP
When the TSTAMP parameter is specified in your system configuration (TSTAMP is the default), then the name of the TPFDF program which found, filed, dispensed or released the record is used as program stamp for the physical record (last file, dispense, and release stamps), for the ST event logs (file, find, dispense and release events), and for the pool file error messages. (Physical record program stamps and ST event logs can be displayed with the help of ZDFIL I. Pool file error messages are logged on the diagnostic file and are printed by the ALCS diagnostic file processor.)

If you specify TPFDF=YES you must include the TPFDF macro libraries in the SOURCE= parameter of the ALCS generation macro.

TRACE=({320|*t1*],[10|*t2*],[ACTIVE|INACTIVE])

[320|*t1*]
Number of entries in the system macro trace block. The minimum value is 1; the maximum is 32 000.

[10|*t2*]
Maximum number of end users who can use the ALCS conversational trace facility at the same time. The minimum value is 1; the maximum is 128.

[ACTIVE|INACTIVE]
End users can (ACTIVE) or cannot (INACTIVE) use the ALCS conversational trace facility. This is the initial state of conversational trace following an ALCS start or restart. The ALCS operator can use the ZTRAC command to change the status. *ALCS Operation and Maintenance* describes the ZTRAC command.

USRTSK1=(NO|YES)
This parameter controls whether ALCS does (YES) or does not (NO) call the third party initialization exit USRTSK1.

If you do not have any third party software which uses the USRTSK1 exit then specify USRTSK1=NO or allow it to default.

For more information about this parameter contact the ALCS Support Group (email alcs@uk.ibm.com).

VFABUF=(*v1*,*v2*,*v3*,...)
The number of size L1, size L2, size L3, ... VFA data buffers.

VFA buffers significantly improve ALCS performance, so allocate as much storage as possible to VFA. A production system needs at least 4MB for VFA buffers.

Note: When VFA buffers are fixed in real storage, ensure that enough real storage is available.

Permanently resident (PR) records remain in VFA buffers permanently. When increasing the number of PR records in a production system, increase the number of VFA buffers accordingly.

Also, time-initiated (TI) records tend to remain longer in VFA buffers. When increasing the number of TI records, increase the number of VFA buffers accordingly.

The number of subparameters must match the number of subparameters on the CISIZE= and RECSIZE= parameters of the ALCS generation macro. The minimum value is 10 for each subparameter. If a subparameter is omitted, the generation calculates a default based on the maximum number of entries that can run concurrently.

You should also consider using Hiperspace backing when planning VFA buffers.

If you specify AMODE64=VFA or AMODE64=LPSVFA, ALCS obtains page-fixed virtual storage needed for the VFA buffers above the bar by using the z/OS 64-bit virtual storage memory object support, hence the amount of allocated storage will be in MEGABYTES. To avoid wasting space choose a value of VFABUF*CISIZE (for each record size) as close as possible to a MB boundary.

WAS={NO|YES|(YES,1|*threads*,5|*time-out*)}
Enable support for WAS where:

NO
WASbridge communication resources can not be started. Application programs can not issue WAS OLA callable services.

YES|(YES,1|*threads*,5|*time-out*)
WASbridge communication resources can be started. When there is no response within the *time-out* period for a WASbridge receive/response connection, then ALCS will use the WASbridge send connection. The *time-out* value is in fifths of a second, the range is 1-150 and the default is 5 (1 second).

Application programs can issue WAS OLA callable services. ALCS allows up to *threads* concurrent OLA calls from application programs. The default for *threads* is 1 and the maximum is 2000.

Generating a communication load module

Use the ALCS communication generation macros to generate a communication load module. This load module can be either of the following:

- A base load module; that is, a load module that contains information to build a communication table.
- An update load module; that is, a load module that contains information to update an existing communication table.

The generation procedure is the same for both.

You can prepare several communication load modules that together define the communication network to ALCS, with a separate stage 1 generation deck for each of these load modules.

Relationships between communication generation macros

There are three ALCS communication generation macros. They are:

COMGEN
Specifies general information about the communication subsystem. Include only one COMGEN macroinstruction in each generation.

If you prepare several communication base and update modules that together define the communication network to ALCS, then ALCS uses information on the COMGEN macro for the first load module in the load list; information on the COMGEN macro for subsequent load modules is ignored.

"COMGEN macro" on page 98 describes the COMGEN macro.

COMDEF
Specifies information about each ALCS communication resource. Include one COMDEF macroinstruction for each communication resource.

With two exceptions, a COMDEF macroinstruction defines a single communication resource. The exceptions are:

- LDTYPE=WTTY,TERM=FDX, where one instruction defines both the send and receive resources of a full-duplex WTTY line.
- LDTYPE=X25ALC, where one instruction can define a group of terminals.

Figure 125 on page 555 describes the COMDEF macro.

COMDFLT
Specifies default values for COMDEF macro parameters. Include as many COMDFLT macroinstructions as required.

"Setting defaults for COMDEF parameters - COMDFLT macro" on page 105 describes the COMDFLT macro.

Many of the parameters for these macros are the same. You should understand how, and in what sequence, ALCS interprets the communication defaults that you set up. The macros are interpreted in the order:

COMGEN, including system defaults
COMDFLT, to override system defaults
COMDEF, to override COMDFLT defaults (if any)

However, this does not mean that you should always use all three macros. You should use the macros to apply, selectively, default values to sets of communication resources.

You may, for example, wish to apply a default to all the printers which form part of your installation. In this case, code the necessary parameter on the COMGEN macro.

You may want to change the defaults for a particular class of device; for example the timeout defaults for 1980 printers which are used for ticket printing, without changing the timeout values for all the other printers. To do this use the COMDFLT macro.

Finally, if you want to change some defaults for a few named devices without altering the defaults for everything else, you can change them using the COMDEF macro.

This is called **cascading defaults**. It gives you great flexibility in defining your communication subsystem. You should code the macros in the order COMGEN, COMDFLT, and finally COMDEF to take advantage of this feature.

Storing communication table data offline

ALCS provides an offline facility that writes a copy of the ALCS communication table to a sequential data set; this data set is called the **communication report file**. It provides the information from which to generate reports about the communication network.

Sequence of communication generation macros

"Generating the initial ALCS configuration" on page 65 shows you the correct order for coding generation macroinstructions within the ALCS stage 1 generation deck. Additional rules apply to the order of macroinstructions within the part of the ALCS stage 1 generation deck that defines the communication configuration.

Note that you can prepare several communication load modules that together define the communication network to ALCS, with a separate stage 1 generation deck for each of these load modules.

Within the combined stage 1 generation deck, the following rules apply:

- For ALC terminals that are connected to ALCS through an SLC link (`LDTYPE=SLCALC`), define the SLC link before defining the terminal or group of terminals that is connected through that SLC link.
- For VTAM parallel sessions within a communication link to another system (`LDTYPE=ALCSLINK`), define the communication link before defining the parallel session or group of parallel sessions to that communication link.
- For ALC terminals that are connected to ALCS through an X.25 PVC (`LDTYPE=X25ALC`), define the X.25 PVC before defining the terminal or group of terminals that is connected through that X.25 PVC.
- For ALC terminals that are connected to ALCS through TCP/IP (`LDTYPE=TCPIPALC`), define the TCP/IP connection before defining the terminal or group of terminals that is connected through that TCP/IP connection.
- For terminals that are connected to ALCS through MQSeries queues (`LDTYPE=MQTERM`), define the MQSeries connection before defining the terminal or group of terminals that is connected through that MQSeries connection.
- For terminals that are connected to ALCS through WAS (`LDTYPE=WASTERM`), define the WAS connection before defining the terminal or group of terminals connected through that WAS connection.

COMGEN macro

Use `COMGEN` to specify general information about the communication subsystem.

Include one `COMGEN` macro in each generation, before any other communication generation macro statement.

```
[label}  COMGEN ACBNAME=acbname
                ,COMID=communication_id
                ,LOGON=({YES|NO},DEFAULTID=userid)
                [,BUFSZE={97|size}]
                [,CRIRANGE=({cri1,cri2],
                [cri1,cri2],[cri1,cri2],[cri1,cri2],.....[cri1,cri2])]
                [,ENTRIES=({n1],[n2],[n3],[n4],[n5],[n6],[n7])]
                [,EXPAND={10|percnt}]
                [,LOGOFF={60||seconds}]
                [,LRSENT=({50|slots],[30|seconds])]
                [,MAXORD={65535|dec_num}]
                [,MEMBER=name]
                [,MQCONVERT={NO|YES}]
                [,NEFLU={RESET|NORESET}]
                [,NOP3270={NO|YES}]
                [,OCTM={NO|YES}]
                [,ORDRANGE=({ord1,ord2],
                [ord1,ord2],[ord1,ord2],[ord1,ord2],.....[ord1,ord2])]
                [,PASSWORD={UPPER|MIXED}]
                [,PPINAME=({ALCSAUTO|nvr1],[ALCSAUTO|nvr2])]
                [,QWARNING=({0|printer_queue_threshold],[0|printer_queue_frequency])]
                [,RCVSZE={L3|MAX}]
                [,RCVSZEIP={L3|MAX}]
                [,REUSE={YES|NO}]
                [,SAFOPTION=(option,...)]
                [,TIMEOUT=({50|time1],[300|time2],[3|count1])]
                [,TIMEOUTV={0|time3}]
                [,USERLEN={0|nnn}]
```

Where:

label
Any valid assembler label.

ACBNAME=*acbname*
The ACB name of this ALCS system (see "Defining ALCS as a logical unit" on page 30).

COMID=*communication_id*
The identification of this ALCS system to message router[4]. The ALCS message router can use this ID to communicate with resources on another system. It is not necessarily the same as the ALCS system identifier. Specify 1 alphabetic character A through Z.

LOGON=({YES|NO},DEFAULTID=*userid*)
Defaults for ALCS access control, where:

YES
End users must provide user ID and password (logon) before using the terminal. Note that ALCS ignores this default for communication resources where ALCS does not support an end user logon.

NO
End users can use the terminal without logging on.

userid
Default user ID for communication resources that do not require or support a logon. ALCS, and optionally customer-written applications, use this user ID to determine what functions and facilities can be activated from these resources.

BUFSZE={97|*size*}
Size in bytes of the printer buffer. A decimal number in the range 0 through 4000. The default is 97.

This parameter specifies a system-wide default; you can use the BUFSZE= parameter in a COMDFLT or COMDEF macro to override this default.

[4] This is equivalent to the TPF CPU-ID. Note that TPF allows only A through P.

CRIRANGE=({*cri1,cri2*],[*cri1,cri2*],[*cri1,cri2*],[*cri1,cri2*],.....[*cri1,cri2*])

A range of communication resource identifier (CRI) addresses for the exclusive use of the ALCS communication generation function. If the ALCS system is using the OCTM facility (OCTM=YES on the COMGEN macro), the CRI addresses in this range will not be used by the OCTM facility.

Each CRI address (*cri1* and *cri2*) must be six hexadecimal digits. The first CRI in each range *cri1* must be lower than the last CRI in each range *cri2*. Specify up to 10 ranges of CRI addresses.

If you are using the OCTM facility, ALCS copies the CRIRANGE defined on COMGEN in the initial base communication configuration load module to the OCTM anchor record. You can use the OCTM offline support program DXCCTMOL to validate and implement new CRIRANGE values in the OCTM anchor record. See *ALCS Operation and Maintenance* for details on running the ALCS OCTM offline support program. In case of an emergency, you can reset the CRIRANGE (and ORDRANGE) values in the OCTM anchor record by omitting them from the base communication configuration load module.

Note: The CRI address ranges defined in this parameter may be extended by the ALCS communication generation to include some additional CRI addresses. For example, if the *cri1* subparameter is coded as 050002 and the *cri2* subparameter is coded as 050017, ALCS will extend this so that the range is from CRI 050000 to 050018. Refer to the configuration report provided by the communications generation to verify the actual ranges of CRI addresses that ALCS has allocated for the exclusive use of the ALCS communication generation function.

ENTRIES=({*n1*],[*n2*],[*n3*],[*n4*],[*n5*],[*n6*],[*n7*])

Note: This parameter is not required with ALCS Version 2 Release 4.1,

since you can add new entries into the communication tables at any time.

The number of initial entries in the communication tables for each type of resource, irrespective of the number of COMDEF macroinstructions. This parameter applies only for a base load module; it is ignored for an update load module. ALCS increases the number of entries in the communication tables if the number of COMDEF macroinstructions exceeds this initial number of entries.

n1
Number of terminal entries. Include any NEF2 or ALCI LUs, NetView and X.25 PVCs, and all terminals attached through them.

n2
Number of SLC link and WTTY entries.

n3
Number of ALCS application entries.

n4
Number of LU 6.1 link and LU 6.2 (APPC) entries.

n5
Number of TCP/IP entries.

n6
Number of MQ entries.

n7
Number of WAS entries.

EXPAND={10|*percnt*}

Note: This parameter is not required with ALCS Version 2 Release 4.1, since you can add new entries into the communication tables at any time.

ALCS increases the number of initial spare entries in the communication table for each type of resource by *percnt* per cent. This allows the subsequent generations to add communication resources without regenerating the entire communication network. *percnt* is a decimal number; the minimum value is 0, the maximum value is 100. It is a percentage of the total COMDEF macroinstructions, or of the number of initial entries (specified by the ENTRIES parameter), whichever is greater. This parameter applies only for a base load module; it is ignored for an update load module.

LOGOFF={60|*seconds*}
The minimum time, in seconds, ALCS requires the device to be available (that is, not in use by ALCS) before ALCS terminates the session if requested to do so by VTAM. This parameter applies only to printer devices of LDTYPE=VTAM3270 where ISTATUS=SHARED is defined in the COMDFLT or COMDEF macros. The default value is 60 seconds, and the possible range is 1 through 600 seconds.

LRSENT=({50|*slots*],[30 |*seconds*])
Suppress duplicate attention messages. When ALCS receives a message from an ALC terminal that is not defined in the communication table, it issues an attention message (one of DXC2412, DXC2413 or DXC2414). Use this parameter to prevent ALCS from generating many of these attention messages, for example when an error occurs in the ALCS network. The options are:

slots
Number of unknown ALC terminal addresses that ALCS records. ALCS suppresses duplicate attention messages for these terminal addresses. If ALCS receives messages from more than this number of unknown terminal addresses then it replaces the oldest recorded address.

seconds
Time interval, in seconds, before ALCS clears its record of an unknown terminal address. If ALCS receives another message from the unknown terminal address within this time interval then it suppresses the attention message.

MAXORD={65535 | *dec_num*}
The maximum ordinal number allowed for a resource. A decimal number in the range 1 through 16 777 216. Use the MAXORD= parameter value in the first communication generation to be loaded (the base load module) to specify the maximum ordinal that ALCS can use.

ALCS uses the MAXORD value to check the ordinal of each resource loaded. The resource ordinal is defined:

- Implicitly (ALCS allocates the ordinals sequentially at load time)
- Explicitly using:
  - The ORD= parameter of the COMDEF macro
  - The IORD= parameter of the COMDFLT macro.

MEMBER=*name*
A name to identify this communication generation. The stage 2 link-edit step uses *name* for the communication load module.

The default name is DXCCDC*iv*, where *i* and *v* are the identity and version specified on the ALCS macro.

MQCONVERT={NO|YES}
Do (YES) or do not (NO) use MQSeries options for converting application message data on the ALCS MQSeries bridge.

- For inbound messages on the MQSeries bridge:

  MQCONVERT=YES causes ALCS to specify the MQGMO_CONVERT option on MQGET calls. This option can be used (with a user-written exit if necessary) to request MQSeries to convert the message data into the queue manager's character set as part of the processing of the MQGET call. You must ensure that the *Format* field specified when the message was put correctly identifies the nature of the data in the message.

  MQCONVERT=NO (the default) causes ALCS to omit the MQGMO_CONVERT option on MQGET calls.
- For outbound messages on the MQSeries bridge:

  MQCONVERT=YES causes ALCS to specify MQFMT_STRING as the format name of the message data on MQPUT calls.

  MQCONVERT=NO (the default) causes ALCS to specify the default format name MQFMT_NONE.

NEFLU={RESET|NORESET}
Use this parameter, in conjunction with the COMDEF macro NEFLU parameter and the ZACOM command, to manage ownership of the terminals that connect to ALCS through ALCI.

RESET
ALCS accepts an ALCI configuration report for any inactive terminals whose LEIDs match those in the report. ALCS updates the owning ALCI LU for those terminals, even if they were previously connected through a different LU.

NORESET
ALCS only accepts an ALCI configuration report for terminals when:

The terminal is inactive, and

The terminal's LEID matches one of the LEIDs in the report, and

The owning ALCI LU for the terminal is either not set or it matches the LU from which the report originated.

NOP3270={NO|YES}
Do (YES) or do not (NO) allow the ALCS no-operation character to be part of the displayable character set in messages to IBM 3270 display terminals. The no-operation character is assembler symbol #NOP, hexadecimal value X'6C'. It is the percent symbol in the EBCDIC character set.

NOP3270=NO (the default) causes ALCS to treat the no-operation character as a new-line character when it occurs after the first 64 characters in any line of an output message to IBM 3270 display terminals.

OCTM={NO|YES}
Use this parameter to specify whether your ALCS system uses the Online Communication Table Maintenance facility (OCTM).

NO
The Online Communication Table Maintenance facility is not used by your ALCS system. During ALCS startup the OCTM database will not be accessed. This is the default.

YES
The Online Communication Table Maintenance facility is used by your ALCS system. During ALCS startup the OCTM database will be used to build the online communication table.

ORDRANGE=({*ord1,ord2*],[*ord1,ord2*],[*ord1,ord2*],[*ord1,ord2*],.....[*ord1,ord2*])
A range of communication resource ordinal numbers for the exclusive use of the ALCS communication generation function. If the ALCS system is using the OCTM facility (OCTM=YES on the COMGEN macro), the ordinal numbers in this range will not be used by the OCTM facility.

The first ordinal number in each range *ord1* must be less than the last ordinal number in each range *ord2*. Specify up to 10 ranges of communication resource ordinal numbers.

If you are using the OCTM facility, ALCS copies the ORDRANGE defined on COMGEN in the initial base communication configuration load module to the OCTM anchor record. You can use the OCTM offline support program DXCCTMOL to validate and implement new ORDRANGE values in the OCTM anchor record. See *ALCS Operation and Maintenance* for details on running the ALCS OCTM offline support program. In case of an emergency, you can reset the ORDRANGE (and CRIRANGE) values in the OCTM anchor record by omitting them from the base communication configuration load module.

Note: The ordinal number ranges defined in this parameter may be extended by the ALCS communication generation to include some additional ordinal numbers. For example, if the *ord1* subparameter is coded as 97 and the *ord2* subparameter is coded as 111, ALCS will extend this so that the range is from ordinal 96 to ordinal 112. Refer to the configuration report provided by the communications generation to verify the actual ranges of ordinal numbers that ALCS has allocated for the exclusive use of the ALCS communication generation function.

PASSWORD={UPPER|MIXED}
Enable support for mixed case passwords.

UPPER
Passwords must be upper case. ALCS will convert passwords to upper case before calling the ESM for password validation.

MIXED
Passwords can be a combination of upper and lower case characters. ALCS will not convert the password to upper case before calling the ESM for validation The ESM must be enabled for mixed case passwords. For example, if RACF is the ESM, specify SETROPTS PASSWORD(MIXEDCASE).

PPINAME=({ALCSAUTO|*nvr1*],[ALCSAUTO|*nvr2*])
Where:

nvr1
Name of the NetView PPI receiver within the NetView started task that processes messages from ALCS. Normally this will be the only NetView started task, in which case the NetView PPI passes all messages and alerts from ALCS to it. The default name is ALCSAUTO.

This name must match the PPI name that is specified in the NetView AMOTLIST member for the NetView started task.

nvr2
Name of the NetView PPI receiver within the NetView started task that processes alerts from ALCS. If ALCS sends many alerts to the NetView PPI (for example SITA NCBs) then you may want to establish a second NetView started task in order to process them separately from ALCS messages. The default name is ALCSAUTO.

This name must match the PPI name that is specified in the NetView AMOTLIST member for the started task.

QWARNING=({0|*printer_queue_threshold*],[0|*printer_queue_frequency*])
The options are:

printer_queue_threshold
This parameter specifies the greatest number of messages on queue for this printer before installation-dependent action is taken. When the number of messages on queue reaches this threshold, ECB-controlled exit programs APRB and APRC are called if they are installed. These exit programs enable you to implement your installation-dependent actions; for example, issue a warning message. A decimal number in the range 10 through 65535 or 0 (default). The default value 0 indicates that there is no threshold specified.

printer_queue_frequency
After the "printer_queue_threshold" has been reached, this parameter specifies the number of additional messages that can be added to the queue for this printer before installation-dependent action is taken. Each time the number of messages on queue increases by this threshold, ECB-controlled exit programs APRB and APRC are called if they are installed. These exit programs enable you to implement your installation-dependent actions; for example, issue a warning message. A decimal number in the range 10 through 1000 or 0 (default). The default value 0 indicates that there is no frequency specified.

RCVSZE={L3|MAX}
Size of the buffers that ALCS uses to receive input messages from IBM 3270 display devices.

L3
ALCS uses a buffer size that is large enough to contain an input message for a size L3 storage block. This is the default.

MAX
ALCS uses a buffer size that is large enough to contain an input message for a size LX storage block.

RCVSZEIP={L3|MAX}
Size of the buffers that ALCS uses to receive input messages from TCP/IP communication devices.

L3
ALCS uses a buffer size that is large enough to contain an input message for a size L3 storage block. This is the default.

MAX
ALCS uses a buffer size that is large enough to contain an input message for a size LX storage block.

This parameter also controls the maximum value of the buffer size parameter for some of the TCP/IP sockets calls that can be issued by application programs: READ, RECV, RECVFROM. By default, ALCS restricts the maximum value of the buffer size parameter to 4000 bytes. If your installation uses storage blocks that are larger than 4000 bytes, you can specify RCVSZEIP=MAX to increase the maximum value of the buffer size parameter to match your largest block size. *ALCS Application Programming Guide* describes how to include TCP/IP sockets calls in application programs.

REUSE={YES|NO}
Re-use CRIs and ordinals.

REUSE=YES
ALCS re-uses CRIs and ordinal numbers from deleted communication resources. This is the default.

REUSE=NO
ALCS does not re-use CRIs and ordinal numbers from deleted communication resources.

SAFOPTION=(*option*,...)
Optional features of ALCS access control. These features may not work with security products other than IBM's RACF. The *options* currently supported are:

EXPIRYWARN
The response to a log on with user ID and password includes, if applicable, a warning that the password is close to its expiry date.

TIMEOUT=({50|*time1*],[300|*time2*],[3|*count1*])
Controls how ALCS performs error recovery for ALC printer printer terminals (printer terminals with LDTYPE=VTAMALC, LDTYPE=SLCALC, LDTYPE=X25ALC, LDTYPE=TCPIPALC, LDTYPE=MQTERM, or LDTYPE=WASTERM).

This parameter specifies a system-wide default; you can use the TIMEOUT= parameter in a COMDFLT or COMDEF macro to override this default.

ALC printer terminals do not generate negative acknowledgements for error conditions.

[50|*time1*]
Answerback timeout in seconds. A decimal number in the range 1 through 300.

If you specify 0 for *time1*, ALCS waits indefinitely for an answerback.

[300|*time2*]
Test message transmission interval in seconds. A decimal number in the range 1 through 600.

If you specify 0 for *time2* or if you specify the SYSSEND=NO parameter in the COMDFLT or COMDEF macros, ALCS does not send test messages.

[3|*count1*]
Recovery retry count. A decimal number in the range 1 through 30.

If you specify 0 for *count1* or if you specify the SYSSEND=NO parameter in the COMDFLT or COMDEF macros, ALCS does not retry the transmission.

After transmitting a message to an ALC printer, ALCS waits for an answerback (a positive acknowledgement). If ALCS does not receive an answerback within *time1* seconds (plus or minus half a second). It then retries (retransmits the message) up to a maximum of *count1* times. Following each of these retransmissions, it again waits for an answerback for up to *time1* seconds.

If ALCS still does not receive an answerback after retrying *count1* times, it transmits a test message (DXC2555) every *time2* seconds. If the printer terminal responds to the test message with an answerback, ALCS resumes normal transmission by resending the original message.

TIMEOUTV={0|*time3*}
Controls error recovery for VTAM supported devices (devices with LDTYPE=VTAM3270, LDTYPE=X25PVC, and LDTYPE=VTAMALC).

0|***time3***
VTAM SEND timeout in seconds. A decimal number in the range 1 through 299. If you specify 0 for *time3*, or allow it to default, ALCS uses the printer answerback timeout that is specified on the COMGEN TIMEOUT parameter instead.

After ALCS issues a VTAM SEND macro to send a message to a logical unit (LU), ALCS waits for VTAM to initiate the SEND. If VTAM does not initiate the SEND within *time3* seconds (plus or minus half a second) then ALCS terminates the session for this LU and issues message DXC2421E. ALCS does not issue message DXC2421E if the system load is too high.

USERLEN={0|*nnn*}
Length in bytes of the user area in each entry in the communication table. The user area is appended to each entry. *nnn* is a decimal value. If *nnn* is not a multiple of 8, COMGEN rounds it up to a multiple of 8. Enter data into this area by using the USERDAT= or USER1= through USER16= parameters on the COMDEF macro.

Setting defaults for COMDEF parameters - COMDFLT macro

Use COMDFLT to specify default values for COMDEF parameters. If a **required parameter** is omitted from COMDEF, a default previously specified by COMDFLT is used. If an **optional parameter** is omitted from COMDEF, a default previously specified by COMDFLT is used. If none is found ALCS uses the system default. Figure 29 on page 105 summarizes the defaults for COMDEF parameters.

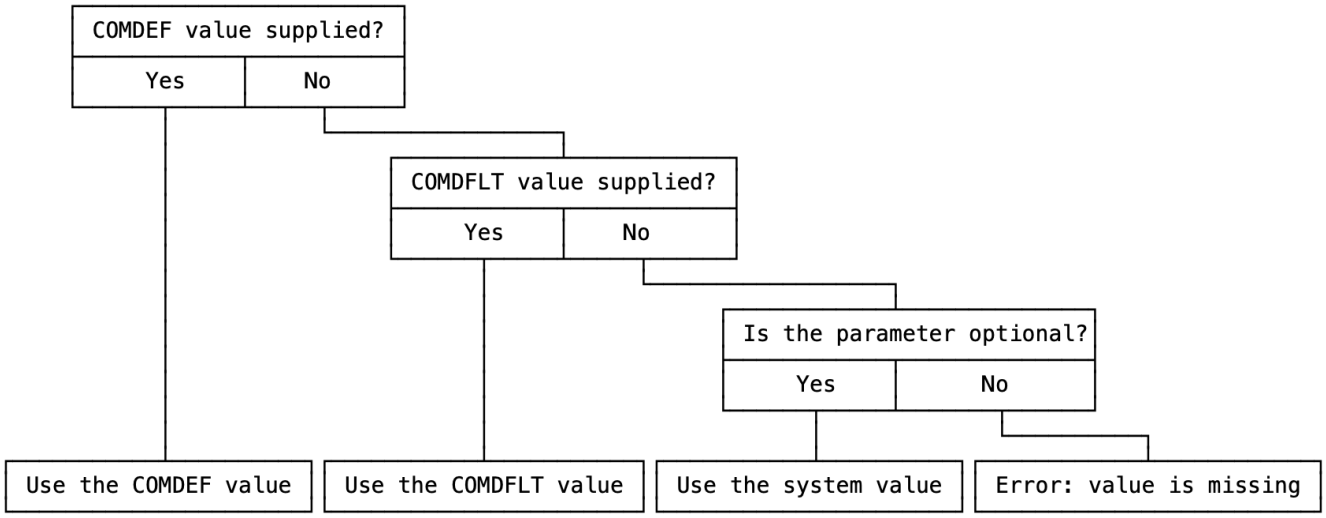


Figure 29. How ALCS evaluates COMDEF parameters

The COMDFLT macro allows you to specify a parameter value once only, instead of coding the same value in many successive COMDEF macros. You can include as many keywords as you like on each COMDFLT macroinstruction, and you can specify COMDFLT as often as you like. See Figure 125 on page 555 for an example of its use.

No syntax checking is performed until a COMDEF macro picks up the default set by COMDFLT.

COMDFLT parameters (except CLEAR=) correspond to the COMDEF parameters described in Figure 125 on page 555.

The positional parameter CLEAR= resets all the existing COMDFLT default values to nulls; that is, COMDEF uses the system default. You can establish new defaults in the same COMDFLT macro, following the CLEAR= parameter.

You can set the following COMDEF parameters with COMDFLT:

```
[label]  COMDFLT[CLEAR]
                          [,APPL=default]
                          [,ASSDEV=default]
                          [,BAUD=default]
```

```
                              [,BPALCCHK=default]
                              [,BUFSZE=default]
                              [,CML=default]
                              [,CODE=default]
                              [,COMID=default]
                              [,COUNTER=default]
                              [,DBCS=default]
                              [,ENQRSP=default]
                              [,FMSG=default]
                              [,HEN=default]
                              [,HEX=default]
                              [,ICRI=init_val]
                              [,IORD=init_val]
                              [,IPMGSZ=default]
                              [,ISTATUS=default]
                              [,LDTYPE=default]
                              [,LINK=default]
                              [,LOGON=default]
                              [,LOOPBIT=default]
                              [,MFROM=default]
                              [,NEFLU=default]
                              [,PRI=default]
                              [,PVC=default]
                              [,RETRANS=default]
                              [,RETRV=default]
                              [,SCRCOL=default]
                              [,SCRSZE=default]
                              [,SFROM=default]
                              [,STPALL=default]
                              [,SYSSEND=default]
                              [,TAB=default]
                              [,TCID=default]
                              [,TERM=default]
                              [,TIMEOUT=default]
                              [,TIMER=default]
                              [,UNSLIFE=default]
                              [,UPDATE=default]
                              [,USERDAT=default]
                              [,USER1=default]
                              ⋮
                              [,USER18=default]
```

The ICRI= and IORD= parameters set initial values as follows:

ICRI
Sets a **starting** value for a **range** of CRIs within a resource type. The CRIs are then allocated sequentially within that type unless COMDEF CRI= explicitly overrides the sequential value.

IORD
Sets a **starting** value for a **range** of ordinals. This allows ranges of ordinals to be allocated according to the communication resource function. The ordinals are then allocated sequentially unless COMDEF ORD= explicitly overrides the sequential value.

Note: A COMDFLT CLEAR clears the IORD and ICRI. The IORD and ICRI are not used when they are clear; instead ALCS uses an implicit value.

How ALCS allocates CRIs

The CRI comprises three bytes as follows:

- A 1-byte **logical device index** (LDI).

  The first (high order) byte contains the LDI.

  Unless you explicitly override it with the COMDEF CRI= parameter or the COMDFLT ICRI= parameter, the ALCS generation allocates LDIs consecutively as it encounters each new logical device type in the sequence of generation macroinstructions.

  ALCS also allocates a new LDI if and when the next logical REI would exceed X'FFFF'.

  For devices which are defined in the base communication load module, either by COMDEF macroinstructions or by COMGEN ENTRIES parameter, ALCS allocates LDIs in the following sequence of logical device type:

- Terminals (including NEF2 or ALCI LUs, NetView and X.25 PVCs, and terminals attached through them).
- SLC links and WTTY links.
- ALCS applications.
- LU 6.1 links and LU 6.2 (APPC) links.
- TCP/IP links.
- MQ queues.
- WAS connections.

If you prefer to choose your own LDI values for each logical device type, then you should build the base communication load module with no devices defined in it, and ensure that:

- The `COMGEN ENTRIES` parameter is omitted from the base load module.
- The `COMGEN MAXORD` parameter in the base load module is at least as large as the total number of devices you are defining in the update load modules.

Note: ALCS allocates LDIs starting with **X'02'** because LDIs X'00' and X'01' are reserved for CRAS CRIs.

ALCS Concepts and Facilities gives an overview of CRIs and ALCS CRAS terminals.

- A 2-byte **resource entry index** (REI).

  The second two bytes (the low-order bytes) of the CRI contain the REI.

  Unless you explicitly override it with the `COMDEF CRI=` parameter or the `COMDFLT ICRI=` parameter, the ALCS generation allocates REIs consecutively (starting with **X'0001'** for each LDI) as each new resource with the same logical device type is encountered in the sequence of generation macroinstructions.

COMDEF macro

Use `COMDEF` to specify information about each communication resource which connects to the ALCS system. The logical device type (`LDTYPE=`) parameter specifies the main characteristics of the communication resource. Figure 30 on page 108 shows which `LDTYPE` to specify in the `COMDEF` macro, Table 12 on page 108 shows where to find more detailed information.

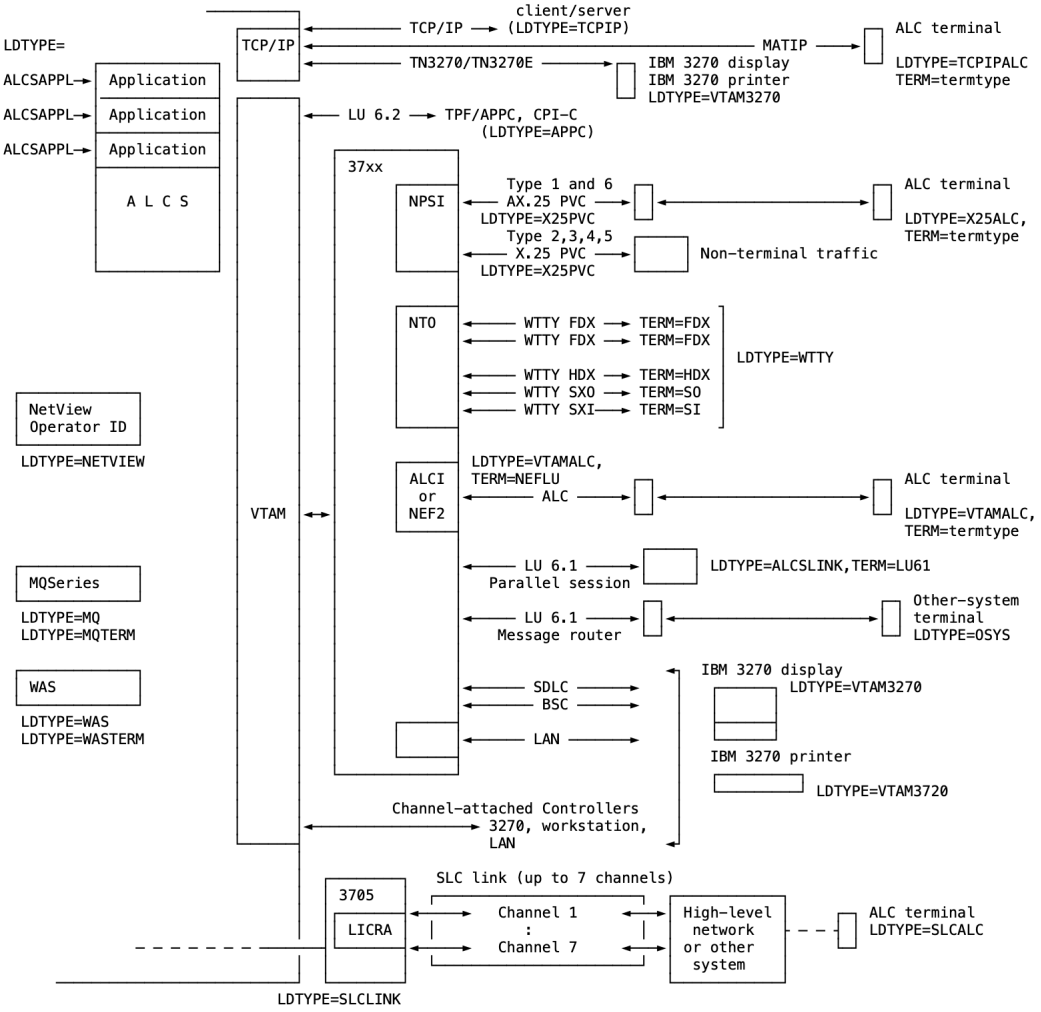


Figure 30. The basic COMDEF parameters for different ALCS communication resources

The formats of the COMDEF macroinstruction for the various ALCS communication resource types are specified in the following sections:

Table 12. Where to find more information about each LDTYPE specification

LDTYPE	
ALCSAPPL	“COMDEF parameters for a local application” on page 123
ALCSLINK	“COMDEF parameters for an LU 6.1 link” on page 125
APPC	“COMDEF parameters for APPC connections” on page 127
MQ	“COMDEF parameters for a connection using MQSeries queues” on page 129
MQTERM	“COMDEF parameters for a terminal connected through MQSeries queues” on page 130
NETVIEW	“COMDEF parameters for NetView” on page 132
OSYS	“COMDEF parameters for an other-system terminal” on page 133
SLCALC	“COMDEF parameters for an SLC ALC terminal” on page 135
SLCLINK	“COMDEF parameters for an SLC link” on page 137
TCPIP	“COMDEF parameters for TCP/IP” on page 142
TCPIPALC	“Parameters for an ALC device on TCP/IP” on page 149

Table 12. Where to find more information about each LDTYPE specification (continued)	
LDTYPE	
VTAMALC	“COMDEF parameters for a NEF or ALCI ALC terminal” on page 152
VTAM3270	“COMDEF parameters for a VTAM 3270 terminal” on page 153
WAS	“COMDEF parameters for a connection to WAS” on page 155
WASTERM	“COMDEF parameters for a terminal connected through WAS” on page 156
WTTY	“COMDEF parameters for a WTTY link” on page 158
X25ALC	“COMDEF parameters for AX.25 ALC terminals” on page 159
X25PVC	“COMDEF parameters for an X.25 permanent virtual circuit (PVC)” on page 162

With three exceptions, a COMDEF macroinstruction defines one communication resource. The exceptions are:

- LDTYPE=WTTY,TERM=FDX, where one instruction defines both the send and receive resources of a full-duplex WTTY link.
- LDTYPE=X25ALC, where one instruction defines a group of terminals.
- LDTYPE=TCPIP,TERM=SERVER, where one instruction defines a group of TCP/IP conections.

The parameters are summarized in Table 13 on page 110.

You do not need to specify all parameters in COMDEF, since you can often let COMDEF take the defaults you have specified in COMDFLT (see “Setting defaults for COMDEF parameters - COMDFLT macro” on page 105).

If you want to define a test ALC terminal see Appendix H, “Generating a test ALC terminal,” on page 575.

The column headed "Set" shows whether the value can be specified in the COMDFLT macro. The column headed "System default" shows any default, when the parameter is not specified either by COMDEF or by COMDFLT.

Table 13. Parameters of the COMDEF macro

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
APPL	Yes	ALCSLINK (TERM=LU61), APPC, MQTERM, OSYS, SLCALC, SLCLINK (PRTCOL=TYPE1 and PRTCOL=TYPE3), TCPIP, TCPIPALC, VTAMALC, VTAM3270, NETVIEW, WAS, WASTERM, WTTY (except simplex out), X25ALC, X25PVC (except PRTCOL=TYPE1, PRTCOL=TYPE6, and PRTCOL=TYPE7)	ALCSLINK (TERM=PARSESS)	1-4 chars	None	**APPL**=*application_name* Name of the application defined by COMDEF LDTYPE=ALCSAPPL to which ALCS routes input messages from this resource.
			SLCLINK (PRTCOL=TYPE2)	1-4 chars	ROUTER	**APPL**={ROUTER | *application_name*} See "COMDEF parameters for an SLC link" on page 137.
ASSDEV	Yes		MQTERM, NETVIEW, OSYS, SLCALC, TCPIPALC, VTAMALC, VTAM3270, WASTERM, X25ALC	1-8 chars	None	**ASSDEV**=*name* The CRN of the associated resource, which may be any resource defined by COMDEF. For CRAS terminals ALCS expects it to be a printer, since it sends some command responses to the associated resource of a CRAS; for other terminals it is normally a printer. The same resource can be associated with many resources. The CRN of the associated resource must not be the CRN of the resource being defined.
BAUD	Yes		SLCLINK	0-65535	4800	**BAUD**={4800 | *bps*} Line speed. See "COMDEF parameters for an SLC link" on page 137.
BPALCCHK	Yes		MQTERM		**NO**	**BPALCCHK**={NO | YES} Indicates whether full ALC checking is done. See "COMDEF parameters for a terminal connected through MQSeries queues" on page 130
BUFSZE	Yes		MQTERM, OSYS, SLCALC, TCPIPALC, VTAMALC, WASTERM, X25ALC	0-4000	97 or 1920	**BUFSZE**=*size* Size in bytes of printer buffer. System defaults for: **TERM=** Buffer size (bytes) **1977** 97 **1980** 97 **3270PRT** 1920

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
CHANS	No	SLCLINK			None	**CHANS**=((*address1*,*address2*), ...) One to seven pairs of MVS I/O device addresses. See "COMDEF parameters for an SLC link" on page 137.
CML	Yes		SLCLINK		Depends on protocol	CML=([NO|YES],[NO|YES],[NO|YES]) Control exchange of CMLs. See "COMDEF parameters for an SLC link" on page 137.
CODE	Yes		X25PVC, SLCLINK, TCPIP (using MATIP)		ALC	**CODE**={ALC|IATA7|IATA5|NONE} For X25PVC - Translation code for input and output messages. For SLCLINK - Translation code for output messages. For TCPIP - Translation code for MATIP sessions opened by ALCS.
			SLCALC, TCPIPALC (using MATIP)		ALC	**CODE**={ALC|IATA7|IATA5|NONE} Translation code for output messages. For SLCALC - ALCS translates input messages from this terminal according to the code specified in the MCI byte of the SLC link envelope. For TCPIPALC - ALCS translates input and output messages for this terminal according to the code specified in the MATIP session.
The following cause ALCS to translate input and output messages to and from transmission code indicated: **ALC** Padded ALC 6-bit **IATA7** ATA/IATA 7-bit (CCITT#5 ASCII) **IATA5** Padded ATA/IATA 5-bit (CCITT#2) With NONE, ALCS does not translate output messages. ATA/IATA *Systems and Communications Reference Manual (Vols 1-7)* describes the ATA/IATA 7-bit and 5-bit transmission codes.						
COMID	Yes	ALCSLINK (TERM=LU61), OSYS, SLCLINK (APPL=ROUTER)	ALCSAPPL, SLCLINK (PRTCOL=TYPE2)	A-Z	None	**COMID**=*communication_id* Communication ID of the system concerned. See appropriate description.
CONN	No		WAS	1-5	CONN=(10, 0)	The number of concurrent receive and send connections

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
CONV	No		APPC		Depends on protocol	CONV={BASIC|MAPPED|SPECIAL} See "COMDEF parameters for APPC connections" on page 127.
COUNTER	Yes		SLCLINK		Depends on protocol	**COUNTER**=(*c1*,*c2*,*c3*,*c4*,*c5*,*c6*,*c7*,*c8*) Values of the counters that control the SLC link. See "COMDEF parameters for an SLC link" on page 137.
CRAS	No		MQTERM, OSYS, SLCALC, TCPIPALC, VTAMALC, WASTERM, X25ALC	17-255	None	**CRAS**={AT*nnn*|AP*nnn*} The resource is an alternate CRAS, AT17-255. The resource is an alternate printer CRAS AP17-255.
			MQTERM (TERM=3270DSP), MQTERM (TERM=3270PRT), WASTERM (TERM=3270DSP), WASTERM (TERM=3270PRT)	1-255	None	**CRAS**={AT*nnn*|AP*nnn*} The resource is an alternate CRAS AT1-AT255. The resource is an alternate printer CRAS AP1-AP255
			NETVIEW, VTAM3270	1-255	None	**CRAS**={PRC|ROC|({AT*nnn*|AP*nnn*} [,NF|F])} CRAS type of the resource. See "COMDEF parameters for a VTAM 3270 terminal" on page 153 and "COMDEF parameters for NetView" on page 132.
CRI	No		All	6 hex digits	Allocated online	**CRI**=*cri* Explicitely specifies a CRI for this communication resource only. CRIs are normally allocated implicitly (and sequentially) within a resource type. The CRI= parameter does not affect this sequence for the next **implicit** CRI. The ICRI= parameter of the COMDFLT macro sets a **starting** value for a **range** of CRIs within a resource type. The CRIs are then allocated sequentially within that type unless COMDEF CRI= explicitly overrides the sequential value. The *ALCS Concepts and Facilities* describes the ALCS CRI.
CSID	No	OSYS	SLCALC, TCPIPALC, VTAMALC, VTAM3270, X25ALC	1-6 hex digits	None	**CSID**=*terminal_id* Cross-system ID, the ID that the **other system** (which owns this terminal) uses to identify this terminal. Values less than 6 digits are padded with zeros.
DBCS	Yes		MQTERM, VTAM3270, WASTERM		NO	**DBCS**={NO|YES} Whether the resource supports the double byte character set or not.

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
ENQRSP	Yes		SLCLINK (PRTCOL=TYPE2)		RSM	**ENQRSP**={RSM|ACK} Type of response for an ENQ or ILB. See "COMDEF parameters for an SLC link" on page 137.
FMSG	Yes		ALCSAPPL		NO	**FMSG**={NO|YES} Whether to pass commands to the application, or not. See "COMDEF parameters for a local application" on page 123.
FLOWID	No		TCPIP (using MATIP)	2 hex digits	None	**FLOWID**=*flw* MATIP flow ID. See "COMDEF parameters for TCP/IP" on page 142.
HDR	No		TCPIP (using MATIP)	2 binary digits	None	**HDR**=*hdr* MATIP header type. See "COMDEF parameters for TCP/IP" on page 142.
HEN	Yes	SLCLINK (PRTCOL=TYPE1 and PRTCOL=TYPE3)	TCPIP (using MATIP)	4 hex digits	None	**HEN**=*HLN_entry_address* HLN address of ALCS's connection to the HLN. See "COMDEF parameters for an SLC link" on page 137 and "COMDEF parameters for TCP/IP" on page 142.
HEX	Yes	SLCALC	TCPIP (using MATIP), TCPIPALC (using MATIP)	4 hex digits	None	**HEX**=*HLN_exit_address* HLN address of the terminal's or link's connection to the HLN. This HLN address can attach many terminals - additional addressing information (TCID, IA, TA) is required to identify a particular terminal. See "COMDEF parameters for an SLC ALC terminal" on page 135 and "COMDEF parameters for TCP/IP" on page 142.
IA	No	SLCALC, TCPIPALC (using MATIP), X25ALC (if its owning X25PVC is defined as PRTCOL=TYPE6 or PRTCOL=TYPE7)		2 hex digits 00 through 3F	None	**IA**=*interchange_address* HLN owner's interchange address of the terminal control unit connecting this terminal.
INQNAME	No	MQ		1-48 chars	None	**INQNAME**=*MQSeries_queue_name* Name of an MQSeries queue for incoming messages. See "COMDEF parameters for a connection using MQSeries queues" on page 129.
IPMGSZ	Yes		TCPIP	32K-2048K 1M-2M		Maximum input or output message size for large (extended format) TCP/IP messages.

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
ISTATUS	Yes		APPC, ALCSLINK, MQ, MQTERM, OSYS, SLCALC, SLCLINK (except virtual SLC link), TCPIP, TCPIPALC, VTAMALC, VTAM3270, WTTY, WAS, WASTERM, X25ALC, X25PVC		ACTIVE	**ISTATUS**={ACTIVE|INACTIVE|SHARED} Status of the resource when ALCS starts: **ACTIVE** ALCS initiates communication with the resource during system restart when ALCS cycles up to a system state above standby state. **INACTIVE** ALCS starts this resource only in response to an external request (for example, an ALCS command). **SHARED** For VTAM3270 printers only, this resource is shared with one or more other ALCSs or other MVS subsystems. ALCS does not attempt to establish a session with a shared device until there is something to transmit.
			ALCSAPPL		ACTIVE	**ISTATUS**=({ACTIVE|INACTIVE} [,PERMANENT]) See "COMDEF parameters for a local application" on page 123.
LDTYPE	Yes	All			None	**LDTYPE**=*resource_type* Type of resource, one of: ALCSAPPL TCPIP ALCSLINK TCPIPALC APPC VTAMALC MQ VTAM3270 MQTERM WAS NETVIEW WASTERM OSYS WTTY SLCALC X25ALC SLCLINK X25PVC
LEID	No	VTAMALC (except TERM=NEFLU)		1-6 hex digits	None	**LEID**=*leid* NEF/ALCI logical end-point ID of the terminal.
LINK	Yes	ALCSLINK (TERM=PARSESS), MQTERM, SLCALC, SLCLINK (virtual SLC link), TCPIPALC (using MATIP), WASTERM	TCPIPALC (not using MATIP)	1-8 chars	None	**LINK**=*name* CRN of the link. See appropriate description

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
LLUNAME	No		APPC	1-8 chars	None	**LLUNAME**=*local_LU_name* APPC local LU name. See "COMDEF parameters for APPC connections" on page 127.
LOGON	Yes		MQTERM, TCPIPALC, VTAM3270, VTAMALC, WASTERM, X25ALC		None	**LOGON**=({YES|NO[,**USERID**=*userid*]}) Logon options for SAF authority checking. If YES is specified, the resource is required to log on with a user ID and password. If NO is specified, then no log on is required. If NO is specified with USERID, the *userid* is the default user ID for the communication resource. If USERID is not specified in COMDEF or COMDFLT, the value on the COMGEN parameter DEFAULTID is used. ALCS, and optionally, user-written applications, use the user ID or default user ID to determine what functions and facilities can be activated from these resources.
			ALCSLINK, APPC, MQ, OSYS, SLCALC, SLCLINK, TCPIP, WAS, WTTY, X25PVC		None	**LOGON**=(NO[,**USERID**=*userid*]) Specifies that the resource does not support a log on. If USERID is specified, it is the default user ID. If USERID is not specified in COMDEF or COMDFLT, the value on the COMGEN parameter DEFAULTID is used. ALCS, and optionally, customer-written applications, use the default user ID to determine what functions and facilities can be activated from these resources.
LOOPBIT	Yes		SLCLINK		**0**; Depends on protocol	**LOOPBIT**=([0|1],[NOTEST|TEST]) See "COMDEF parameters for an SLC link" on page 137
LPORT	No	TCPIP (TERM=SERVER)		1-65534	None	**LPORT**=*port_number* Local port number. See "COMDEF parameters for TCP/IP" on page 142

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
MAXCONN	No		TCPIP (TERM=SERVER)	1-4000	None	**MAXCONN**={256 | *n*} Maximum number of concurrent connections. A decimal number in the range 1 through 4000. See "COMDEF parameters for TCP/IP" on page 142.
MFORM	Yes		ALCSAPPL		IMSG	**MFORM**={IMSG | OMSG | AMSG | XMSG} Expected message format of input messages. See "COMDEF parameters for TCP/IP" on page 142.
MODE	No		APPC	1-8 chars	None	**MODE**=*mode_name* APPC mode name. See "COMDEF parameters for APPC connections" on page 127.
MPX	No	TCPIP (using MATIP)	2 binary digits	None	None	**MPX**=*mpx* MATIP multiplex type. See "COMDEF parameters for TCP/IP" on page 142.
NAME	No	All		Varies	None	**NAME**=*name* Name (usually CRN) of the resource. The *ALCS Concepts and Facilities* describes the ALCS CRN.
NEFLU	Yes		VTAMALC (except TERM=NEFLU)	1-8 chars	None	**NEFLU**=*name* CRN of the ALCI LU through which ALCS accesses this terminal. If this is specified, and NEFLU=NORESET is specified on the COMGEN macro, then ALCS only accepts an input message or ALCI configuration report for this terminal if both the LEID and ALCI LU name match the values defined in the generation.
ORD	No		All	Any decimal number up to the value specified on the COMGEN MAXORD= parameter.	Allocated on line	**ORD**=*ordinal* Explicitly specifies the communication resource ordinal for this communication resource only. Communication resource ordinals are normally allocated implicitly (and sequentially) for each communication resource. The IORD= parameter of the COMDFLT macro sets a **starting** value for a **range** of ordinals. This allows ranges of ordinals to be allocated according to the communication resource function. The ordinals are then allocated sequentially unless a COMDEF ORD= explicitly overrides the sequential value.

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
OUTQNAME	No		MQ	1-48 chars	None	**OUTQNAME**=*MQSeries_queue_name* Name of an MQSeries queue for outgoing messages. See "COMDEF parameters for a connection using MQSeries queues" on page 129.
PLUNAME	No		APPC	1-17 chars	None	**PLUNAME**=*partner_LU_name* APPC partner LU name. See "COMDEF parameters for APPC connections" on page 127.
PLUNAME2	No		APPC	1-17 chars	None	**PLUNAME2**=*partner_LU_name* APPC partner LU name. See "COMDEF parameters for APPC connections" on page 127.
PRI	Yes		ALCSLINK (TERM=LU61), VTAM3270, X25PVC	0-255	0	**PRI**={0|*nnn*} Priority of the resource. During communication initialization, ALCS acquires VTAM LU resources in priority order, starting with priority 0.
PROG	No	ALCSAPPL		4 chars	None	**PROG**=*program_name* Name of program or transfer vector. See "COMDEF parameters for a local application" on page 123.
PRTCOL	Yes		APPC, SLCLINK (except virtual SLC link), X25PVC, and WAS.	Varies	Depends on resource type	PRTCOL=TYPE*n* Protocol of the link. See appropriate description.
PVC	Yes	X25ALC		1-8 chars	None	**PVC**=*name* CRN of the permanent virtual circuit through which ALCS accesses the terminal. See "COMDEF parameters for AX.25 ALC terminals" on page 159.
REGNAME	No	WAS		1-12 chars	None	**REGNAME**=*application_name* *Application_name* is the name of the ALCS - WAS set of local connections for this WAS resource. This must be a GRS complex unique name and must be a maximum of twelve alphanumeric characters. See "COMDEF parameters for a connection to WAS" on page 155.
RETRV	Yes		MQTERM, OSYS, NETVIEW, SLCALC, TCPIPALC, VTAMALC, VTAM3270, WASTERM, X25ALC		NO	**RETRV**={NO|YES} Whether the retrieve facility is (YES) or is not (NO) automatically activated for the resource. See *ALCS Operation and Maintenance* for a description of the ZRETR command.

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
RETRANS	Yes		SLCLINK (PRTCOL=TYPE2)		PART	**RETRANS**=={PART|ALL} Type of message retransmission after an I/O error. See "COMDEF parameters for an SLC link" on page 137.
RHOST	No	TCPIP (TERM=CLIENT)	TCPIPALC	1-15 chars in dotted decimal notation	None	**RHOST**=*IP_address* Remote IP address or **RHOST**=([*IP1*],[*IP2*],[*IP3*],[*IP4*]) Remote IP address(es). See "COMDEF parameters for TCP/IP" on page 142 and see "Parameters for an ALC device on TCP/IP" on page 149.
RHSID	No		ALCSLINK (TERM=PARSESS)		None	**RHSID**=*rname* CICS and IMS name of the parallel session. See "COMDEF parameters for an LU 6.1 link" on page 125.
RPORT	No	TCPIP (TERM=CLIENT)		1-65534	None	**RPORT**=*port_number* Remote port number. See "COMDEF parameters for TCP/IP" on page 142.
RPORTMATCH	No		TCPIP (TERM=CLIENT)		NO	**RPORTMATCH**={NO|YES} Whether the remote port number is (YES) or is not (NO) used to check an inbound connection. See "COMDEF parameters for TCP/IP" on page 142.
SCROLL	Yes			TCPIPALC, WASTERM	Yes	**SCROLL**={YES|NO} Allow scrolling (YES) to different parts of those responses that are too large to fit on a single screen. See "SCROLL={YES|NO}]" on page 150.
SCRSZE	Yes		MQTERM, OSYS, SLCALC, TCPIPALC, VTAMALC, WASTERM, X25ALC		12, 24, or 30	**SCRSZE**=*rows* Number of lines on the display terminal, system defaults for: **TERM=** Screen size (lines) **2915** 12 **4505** 30 **3270DSP** 24

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
SERVNAME	No	WAS			None	**SERVNAME**=(*r1*,*r2*) Where: *r1* is *requestservicename_1* and *r2* is *requestservicename_2*. See "COMDEF parameters for a connection to WAS" on page 155.
SESSION	No		ALCSLINK (TERM=PARSESS)		SEND	**SESSION**={RECEIVE|SEND} Whether ALCS acts as a secondary LU (SEND) or a primary LU (RECEIVE). "COMDEF parameters for an LU 6.1 link" on page 125 describes this parameter in more detail.
SFORM	Yes		ALCSAPPL		YES	**SFORM**= ([{YES|NO}][,MIXED][,NOFID]) How to process input messages. See "COMDEF parameters for a local application" on page 123.
SPORT	No		TCPIP (TERM=CLIENT)	1 - 65534	None	**SPORT**=*port_number* Local port number for server. See "COMDEF parameters for TCP/IP" on page 142.
SPORTMATCH	No		TCPIP (TERM=CLIENT)		NO	**SPORTMATCH**={NO|YES} Whether the local port number for the server is (YES) or is not (NO) used to check an inbound connection. See "COMDEF parameters for TCP/IP" on page 142.
SQNAME	No		MQ	1-48 chars	None	**SQNAME**=*MQSeries_queue_name* Name of an MQSeries queue for service messages. See "COMDEF parameters for a connection using MQSeries queues" on page 129.
STPALL	Yes		SLCLINK (PRTCOL=TYPE2)		NO	**STPALL**={NO|YES} Whether to send STPALL to stop all channels on the link. See "COMDEF parameters for an SLC link" on page 137.
SYMDEST	No		APPC	1-8 chars	None	**SYMDEST**=*symbolic_destination_name* APPC symbolic destination name. See "COMDEF parameters for APPC connections" on page 127.
SYSSEND	Yes		MQTERM, NETVIEW, OSYS, SLCALC, TCPIPALC, VTAMALC, VTAM3270, WASTERM, X25ALC			**SYSSEND**={YES|NO} Whether to send system generated messages to the designated device (which can include any printer). System generated messages include system error messages and unsolicited messages.

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
SYSSTATE	No		ALCSAPPL		NORM	**SYSSTATE**={NORM|MESW|CRAS|IDLE} The lowest system state in which the application accepts input messages
TA	No	SLCALC, TCPIPALC (using MATIP), X25ALC		2 hex digits X'00' through X'3F'	None	**TA**=*terminal_address* | **TA**=(*ta*,...) Terminal address(es).
TAB	Yes		MQTERM, OSYS, SLCALC, TCPIPALC, VTAMALC, WASTERM, X25ALC		NO	**TAB**={NO|YES} Whether the printer has the tabular skip feature or not. It only applies where TERM=1977.
TCID	Yes	SLCALC, TCPIPALC (using MATIP), X25ALC (if its owning X25PVC is defined as PRTCOL=TYPE7)		2 hex digits	None	**TCID**=*hex_number* HLN owner's transmission circuit identifier of the terminal.
TERM	Yes	ALCSLINK, MQTERM, OSYS, SLCALC, TCPIPALC, VTAMALC, VTAM3270 (TEST=YES), WASTERM, WTTY, X25ALC	NETVIEW		None	**TERM**=*terminal-type* Terminal or other device type. See appropriate description.
		TCPIP			None	**TERM**=(CLIENT|SERVER[,*protocol*[,*hth*]]) Protocol of the connection. See "COMDEF parameters for TCP/IP" on page 142.
TEST	Yes		VTAMALC, VTAM3270, WTTY, X25ALC, X25PVC		NO	**TEST**={NO|YES:} **NO** Not a test resource. **YES** Test resource. STV supplies input messages and receives output.
TIMER	Yes		SLCLINK (except virtual SLC link)		Depends on protocol	**TIMER**=(*i1*,*i2*,*i3*,*i4*,*i5*,*i6*,*i7*,*i8*,*i9*,*i10*,*i11*,*i12*) Values of time intervals that control the SLC link. See "COMDEF parameters for an SLC link" on page 137.

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
TIMEOUT	Yes		MQTERM, SLCALC, TCPIPALC, VTAMALC, WASTERM, X25ALC			**TIMEOUT**=([50|*time1*],[300|*time2*],[3|*count1*]) Controls how ALCS performs error recovery for printer terminals. For a full description refer to the TIMEOUT parameter on the COMGEN macro ("COMGEN macro" on page 98).
			TCPIP			**TIMEOUT**=([0|*time1*],[0|*time2*],[SSQ]) Controls how ALCS performs error recovery for TCP/IP connections. See "COMDEF parameters for TCP/IP" on page 142.
TPNAME	No		APPC	1-48 chars	None	**TPNAME**=*partner_TP_name* APPC partner TP name. See "COMDEF parameters for APPC connections" on page 127.
TRANSLATE	No		TCPIP		YES	**TRANSLATE**={YES|NO} Whether message data is (YES) or is not (NO) translated by ALCS. See "COMDEF parameters for TCP/IP" on page 142.
UNSLITE	Yes		MQTERM, OSYS, SLCALC, TCPIPALC, VTAMALC, WASTERM, and X25ALC		NO	**UNSLITE**={NO|YES} Whether the terminal has the unsolicited light feature. It only applies where TERM=1977.
UPDATE	Yes		All		ADD	**UPDATE**={ADD|DELETE|REPLACE} Type of update to the existing communication configuration table.

Note:

DELETE

If DELETE is specified:

The only other parameters required are the resource name, the terminal address (for LDTYPEs SLCALC and X25ALC), and the interchange address (for LDTYPE SLCALC). Other parameters are ignored.

The communication resource ordinal number and the CRI become available for re-use by other resources. This may affect the ordinal numbers and CRIs that are allocated to other communication resources in the communication generation. You should use the ALCS communication report generator (DXCCOMOL) to check which ordinal numbers and CRIs have been allocated to each resource. You are recommended to use the COMDEF ORD= and CRI= parameters (and the COMDFLT IORD= and ICRI= parameters) to control which ordinal numbers and CRIs are allocated to specific communication resources.

Table 13. Parameters of the COMDEF macro (continued)

Parameter	Set	Required with LDTYPEs	Optional with LDTYPEs	Length or range	System default	Description
USERDAT USER1 USER2 . USER16	Yes Yes Yes . Yes		All All All . All	1-255 chars 1-255 chars 1-255 chars . 1-255 chars	None None None . None	**USERDAT**=*user_data* **USER1**=*user_data* **USER2**=*user_data* ⋮ **USER16**=*user_data* User data associated with the resource. COMDEF passes this data to a user-written macro, DXCZCUSR. DXCZCUSR processes the data and returns it to the communication generation process. The processed data is placed in the user area of the communication table. See "ALCS DXCZCUSR user macro" on page 444.
WASNAME	No	WAS		1 - 8 characters		**WASNAME**=(*dgname*,*nname*,*sname*) Where: *dgname* is *daemongroupname*, *nname* is *nodename*, and *sname* is *servername* See "COMDEF parameters for a connection to WAS" on page 155.
X25NAME	No	SLCLINK (virtual SLC link)			None	**X25NAME**=*xname* CRN of the X.25 PVC associated with the link. See "COMDEF parameters for a virtual SLC link used to address an X.25 PVC or TCP/IP resource." on page 142.

COMDEF parameters for a local application

Use this COMDEF format to define an application to which ALCS can route messages. The application can be:

- Local, running in this ALCS system
- Remote, running in another ALCS or TPF system.

Figure 31 on page 123 gives an overview of COMDEF parameters to specify a local application.

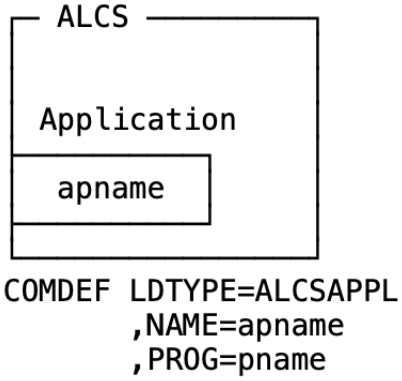


Figure 31. Defining a local application

```
[label}  COMDEF  LDTYPE=ALCSAPPL
                        ,NAME=application_name
                        ,PROG=program_name
                        [,COMID=communication_id]
                        [,CRI=explicit_cri]
                        [,FMSG={NO|YES}]
                        [,ISTATUS=([ACTIVE|INACTIVE}[,PERMANENT])]
                        [,MFORM={IMSG|OMSG|AMSG|XMSG}]
                        [,ORD=explicit_ordinal]
                        [,SFORM=({[{YES|NO}][,MIXED][,NOFID])]
                        [,SYSSTATE={NORM|MESW|CRAS|IDLE}]
                        [,UPDATE={ADD|DELETE|REPLACE}]
                        [,USERDAT=user_data]
                        [,USER1=user_data]
                        ⋮
                        [,USER16=user_data]
```

Where:

NAME=*application_name*
CRN of the application. An alphanumeric string of 1 to 4 characters; the first character must be alphabetic. *application_name* must be unique within ALCS. The ALCS router uses *application_name* to route to the application.

Note: *application_name* is not an LU name. This is because to VTAM, ALCS is the application. Applications that run under ALCS are transparent to VTAM.

PROG=*program_name*
The 4-character name of the program or transfer vector to which ALCS passes messages for this application. This is the input message editor for the application. The name must conform to the specifications for program names within ALCS. The same program can be specified for more than one application.

COMID=*communication_id*
For a local application, this must be the same value specified by the COMID= parameter of the COMGEN macro.

FMSG={NO|YES}
Specifies how to process ALCS commands (input messages beginning with the letter Z).

NO
ALCS does not pass commands to the application. The ALCS command processor processes all commands.

YES
ALCS passes all messages to the application (except those with first 5 characters ZROUT or ZLOGF).

The FMSG= parameter has no effect on messages from:

- Prime CRAS or CRAS AT1 through AT16. ALCS always passes commands from these CRASs to the ALCS command processor.
- WTTY links. ALCS never passes input messages from WTTY links to the ALCS command processor (input messages from WTTY links cannot be ALCS commands).

ISTATUS=([ACTIVE|INACTIVE}[,PERMANENT])
Initial status of the application, where:

ACTIVE
ALCS can route input messages to the application.

INACTIVE
ALCS cannot route input messages to the application.

PERMANENT
The ZACOM command (see the *ALCS Operation and Maintenance*) cannot change the initial status of the application. Omit PERMANENT to allow the ALCS operator to activate and deactivate the application.

MFORM={IMSG|OMSG|AMSG|XMSG}
The application expects input messages in the corresponding message format. See the *ALCS Application Programming Reference - Assembler* for descriptions of these message formats. AMSG is interpreted as the AMSG input message format.

SFORM=({[{YES|NO}][,MIXED][,NOFID])
Specifies special input message formatting requirements for the application, where:

YES|NO
For messages from ALC terminals, after translating from line code, do (YES) or do not (NO) process the messages as follows:

- Check for end-of-message, backspace, erase, and non valid characters in the text.
- Convert any action key characters in the text into the appropriate program function (PF) key value.
- Convert any field identifier (FID) characters in the text into the appropriate application command (subject to the NOFID parameter, see below).

NOFID
Do not convert any field identifier (FID) characters in the text.

MIXED
Do not translate input message text to upper case.

SYSSTATE={NORM|MESW|CRAS|IDLE}
The lowest system state in which the application program accepts input messages.

For other parameters, see Table 13 on page 110; and see also “Setting defaults for COMDEF parameters - COMDFLT macro” on page 105.

COMDEF parameters for a remote application

Figure 32 on page 125 gives an overview of COMDEF parameters used to specify a remote application.

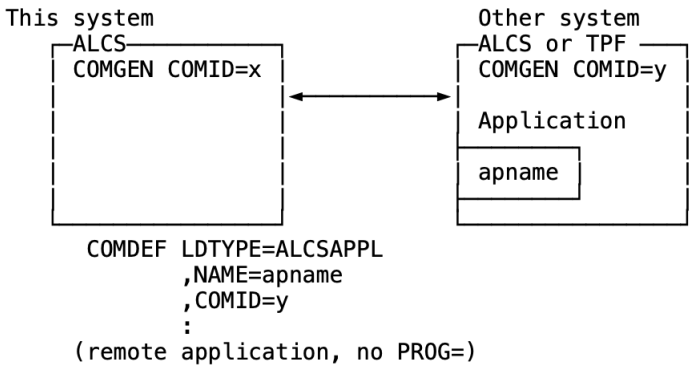


Figure 32. Defining an application owned by another system

```
[label}  COMDEF  LDTYPE=ALCSAPPL
                          ,NAME=application_name
                          ,COMID=communication_id
                          [,CRI=explicit_cri]
                          [,ORD=explicit_ordinal]
                          [,UPDATE={ADD|DELETE|REPLACE}]
                          [,USERDAT=user_data]
                          [,USER1=user_data]
                          ⋮
                          [,USER16=user_data]
```

NAME=*application_name*
CRN of the application on the system that owns the application. An alphanumeric string of 1 to 4 characters; the first character must be alphabetic. *application_name* must be unique within ALCS. The ALCS router uses *application_name* to route to the application.

Note: *application_name* is not an LU name. This is because to VTAM ALCS is the application. Applications that run under ALCS are transparent to VTAM.

COMID=*communication_id*
Communication ID of the system that owns this resource. It is 1 alphabetic character (A through Z).

COMDEF parameters for an LU 6.1 link

Use these COMDEF formats to define an LU 6.1 communication link to another system.

Note: The ALCS message router can use LU 6.1 links to communicate with **other-system** resources.

Figure 33 on page 125 gives an overview of COMDEF parameters used to specify an LU 6.1 link to another system.

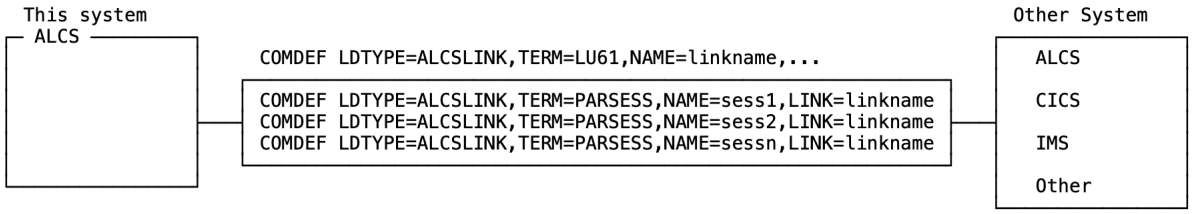


Figure 33. Defining an LU 6.1 link and parallel sessions

Note: Figure 33 on page 125 only shows the different usage of the NAME= parameter, Figure 125 on page 555 shows a more practical example of their usage.

For the LU 6.1 link:

```
[label}  COMDEF  LDTYPE=ALCSLINK
                           ,TERM=LU61
                           ,NAME=link_name
                           ,APPL=application_name
                           [,COMID=communication_id]
                           [,CRI=explicit_cri]
                           [,ISTATUS={ACTIVE|INACTIVE}]
                           [,LOGON=(NO[,USERID=userid])]
                           [,ORD=explicit_ordinal]
                           [,PRI={0|nnn}]
                           [,UPDATE={ADD|DELETE|REPLACE}]
                           [,USERDAT=user_data]
                           [,USER1=user_data]
                           ⋮
                           [,USER16=user_data]
```

For a parallel session:

```
[label}  COMDEF  LDTYPE=ALCSLINK
                           ,TERM=PARSESS
                           ,NAME=sess_name
                           ,LINK=link_name
                           [,APPL=application_name]
                           [,CRI=explicit_cri]
                           [,ISTATUS={ACTIVE|INACTIVE}]
                           [,LOGON=(NO[,USERID=userid])]
                           [,ORD=explicit_ordinal]
                           [,RHSID=rname]
                           [,SESSION={SEND|RECEIVE}]
                           [,UPDATE={ADD|DELETE|REPLACE}]
                           [,USERDAT=user_data]
                           [,USER1=user_data]
                           ⋮
                           [,USER16=user_data]
```

Where:

TERM=LU61
An SNA LU 6.1 link.

TERM=PARSESS
A parallel session within an LU 6.1 link.

NAME=*link_name*
CRN of the link. An alphanumeric string of 1 to 8 characters; the first character must be alphabetic. *link_name* must be unique within ALCS. With TERM=LU61,, *application_name* is the LU name of the system at the remote end of this link.

NAME=*sess_name*
CRN of the parallel session. An alphanumeric string of 1 to 8 characters; the first character must be alphabetic. *sess_name* must be unique within ALCS. With TERM=PARSESS,, *sess_name* is the local half-session name of this parallel session.

APPL=*application name*
Name of the application to which ALCS routes messages received on this link or parallel session. If you omit the **APPL** parameter for an LU6.1 parallel session, ALCS uses the default application name previously specified by a COMDFLT macroinstruction. If there is no default application name, then ALCS uses the application name specified on the COMDEF macroinstruction for the LU6.1 link to which the parallel session belongs.

When the ALCS message router uses the link, messages (received on this link) are not passed to this application; they are passed to the destination application that is identified within the message (in PPMSG format).

This parameter is ignored if the link is used only by the message router.

COMID=*communication_id*
Communication ID of the system at the remote end of the link. It is 1 alphabetic character, A through Z.

LINK=*link_name*
CRN of the LU 6.1 link to which this parallel session belongs. This is the *link_name* specified in a COMDEF statement TERM=LU61,**NAME**=*link_name*.

RHSID=*rname*
Specify this parameter when the parallel session communicates with IMS or CICS. *rname* is the same as the value specified in the NAME parameter in the IMS SUBPOOL macroinstruction, or in the SESSNAME= parameter of the CICS SESSIONS definition.

SESSION={SEND|RECEIVE}

RECEIVE
ALCS issues a VTAM SIMLOGON request to start the session, and ALCS acts as the primary LU. Specify SESSION=RECEIVE for:

- ALCS-to-ALCS LU 6.1 parallel session
- ALCS-to-non-ALCS LU 6.1 parallel session where ALCS is in receive mode, and the non-ALCS system is in send mode.

SEND
ALCS issues a VTAM REQSESS request to start the session, and ALCS acts as the secondary LU. Specify SESSION=SEND for:

- ALCS-to-non-ALCS LU 6.1 parallel session where the non-ALCS system is in receive mode, and ALCS is in send mode.

For other parameters, see Table 13 on page 110; and see also "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

COMDEF parameters for APPC connections

Use this COMDEF format to define an APPC connection between this ALCS and another APPC transaction program (TP).

```
[label}  COMDEF  LDTYPE=APPC
                          ,NAME=crn
                          ,APPL=appl
                          [,CONV={BASIC|MAPPED|SPECIAL}]
                          [,CRI=explicit_cri]
                          [,ISTATUS={ACTIVE|INACTIVE}]
                          [,LLUNAME=local_LU_name]
                          [,LOGON=(NO[,USERID=userid])]
                          [,MODE=mode_name]
                          [,ORD=explicit_ordinal]
                          [,PLUNAME=partner_LU_name]
                          [,PLUNAME2=partner_LU_name]
                          [,PRTCOL={TYPE1|TYPE2|TYPE3}]
                          [,SYMDEST=symbolic_destination_name]
                          [,TPNAME=partner_TP_name]
                          [,UPDATE={ADD|DELETE|REPLACE}]
                          [,USERDAT=user_data]
                          [,USER1=user_data]
                          :
                          [,USER16=user_data]
```

Where:

NAME=*crn*
CRN of this APPC resource, 1 to 8 alphanumeric characters. It must be unique within this ALCS system. This name must correspond to the transaction scheduler section of an entry in the APPC/MVS TP profile file, since it is used with an inbound allocate request to start an APPC conversation with ALCS.

CONV={BASIC|MAPPED|SPECIAL}
Type of conversation, where:

BASIC
The conversation type is basic.

On output:
ALCS converts the output message data (starting at field CM1CMW in the CM1CM message block) into a single logical record before sending it on the conversation. A logical record consists of a 2-byte field that contains the length of the message data, followed by the data.

On input:
ALCS converts the input message data from logical records after receiving it on the conversation. The input message data is passed to the application input edit program (starting at field CM1TXI in the CM1CM message block).

MAPPED
Conversation type is mapped.

On output:
ALCS does not convert the output messages into logical records.

On input:
ALCS does not convert the input messages from logical records. The input message data is passed to the application input edit program (starting at field CM1TXI in the CM1CM message block).

This is the default when PRTCOL=TYPE2 or PRTCOL=TYPE3.

SPECIAL
Conversation type is basic.

On output:
ALCS converts the output message data (starting at field CM1TXT+2 in the CM1CM message block) into a single logical record before sending it on the conversation.

On input:
ALCS does not convert the input messages from logical records. Input message data is passed to the application input edit program (starting at field CM1TXI+2 in the CM1CM message block).

This is the default when PRTCOL=TYPE1.

LLUNAME=*local_LU_name*
The APPC local LU name for conversations between ALCS and another APPC TP. 1 to 8 alphanumeric characters. If this parameter is specified, ALCS uses it for allocating outbound conversations.

MODE=*mode_name*
The APPC mode name for conversations between ALCS and another APPC TP. 1 to 8 alphanumeric characters. If this parameter is specified, ALCS uses it for allocating outbound conversations.

PLUNAME=*partner_LU_name*
The APPC partner LU name for conversations between ALCS and another APPC TP. 1 to 17 alphanumeric characters. If you specify this parameter, ALCS uses it for allocating outbound conversations.

PLUNAME2=*partner_LU_name*
The APPC partner LU name for conversations between ALCS and another APPC TP. 1 to 17 alphanumeric characters. If you specify this parameter, ALCS uses it for allocating inbound conversations when the TP profile for an inbound allocate request specifies CRN=*. In this case, ALCS scans all the APPC connections defined in the communication generation, until it finds one whose partner LU name matches the partner LU name for the inbound allocate.

To find a match, ALCS checks the PLUNAME2 parameter (if it is specified) otherwise it checks the PLUNAME parameter. Specify PLUNAME2 for a type 1 APPC connection where the partner LU for the inbound conversation is different from the partner LU for the outbound conversation.

PRTCOL={TYPE1|TYPE2|TYPE3}
Type of APPC connection, where:

TYPE1
The connection comprises two conversations with the APPC partner defined for this resource. ALCS uses one conversation for sending data and the other conversation for receiving data.

TYPE2
The connection comprises one conversation with the APPC partner defined for this resource. ALCS uses this conversation for sending data.

TYPE3
The connection comprises one conversation with the APPC partner defined for this resource. ALCS uses this conversation for receiving data.

SYMDEST=*symbolic_destination_name*
The APPC symbolic destination name for conversations between ALCS and another APPC TP. 1 to 8 alphanumeric characters. This name must correspond to an entry in the APPC/MVS side information file.

TPNAME=*partner_TP_name*
The APPC partner TP name for conversations between ALCS and another APPC TP. 1 to 48 alphanumeric characters. If this parameter is specified, ALCS uses it for allocating outbound conversations.

For other parameters see Table 13 on page 110; and see also "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

If you specify the SYMDEST parameter, then you can omit the PLUNAME, MODE, and TPNAME parameters.

If the APPC partner is TPF, you must specify the SYMDEST parameter and omit the PLUNAME, MODE, and TPNAME parameters.

Refer to *MVS Programming: Writing Transaction Programs for APPC/MVS* for information about allocating sessions and conversations between TPs. The PLUNAME, LLUNAME, MODE, TPNAME, and SYMDEST parameters of the COMDEF macro correspond to parameters on the APPC/MVS Allocate callable service:

PLUNAME
corresponds to Partner_LU_name

LLUNAME
corresponds to Local_LU_name

MODE
corresponds to Mode_name

TPNAME
corresponds to TP_name

SYMDEST
corresponds to Sym_dest_name

Refer to *MVS Planning: APPC Management* for information about planning, configuring, activating, customizing, and maintaining APPC/MVS.

COMDEF parameters for a connection using MQSeries queues

Use this COMDEF format to define a connection using MQSeries queues.

```
[label}  COMDEF  LDTYPE=MQ
                           ,NAME=name
                           ,INQNAME=MQSeries-queue-name
                           [,CRI=explicit_cri]
                           [,ISTATUS={ACTIVE|INACTIVE}]
                           [,LOGON=(NO[,USERID=userid])]
                           [,ORD=explicit_ordinal]
                           [,OUTQNAME=MQSeries_queue_name]
                           [,SQNAME=MQSeries_queue_name]
                           [,UPDATE={ADD|DELETE|REPLACE}]
                           [,USERDAT=user_data]
                           [,USER1=default]
                           ⋮
                           [,USER16=default]
```

Where:

NAME=*name*
CRN of this connection, 1 to 8 characters. It must be unique within this ALCS system.

INQNAME=*MQSeries_queue_name*
Name of an MQSeries queue, 1 to 48 alphanumeric characters. This is the name of a local queue. This queue is used for incoming messages from a remote system.

ISTATUS={ACTIVE|INACTIVE}
Initial status of the connection, where:

ACTIVE
ALCS sets the resource active in the relevant communication table entry after ALCS restart is complete. If there are any messages on the MQSeries request queue, ALCS starts processing them. You must specify the SCTGEN parameter `MQM=(YES,CONNECT)` in your system configuration table in order to make this happen.

INACTIVE
ALCS starts this resource only in response to an external request (for example, an ALCS command).

OUTQNAME=*MQSeries_queue_name*
Optional name of an MQSeries queue, 1 to 48 alphanumeric characters. This is the name of a local queue, or the local name (alias) of a remote queue. This queue is used for outgoing messages to a remote system.

SQNAME=*MQSeries_queue_name*
Optional name of an MQSeries queue, 1 to 48 alphanumeric characters. This is the name of a local queue, or the local name (alias) of a remote queue. This queue can be used for service messages to a remote system (for example when the MQSeries bridge is started).

For other parameters, see Table 13 on page 110; and see also "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

COMDEF parameters for a terminal connected through MQSeries queues

Use this `COMDEF` format to define a terminal connected through MQSeries queues.

```
[label}  COMDEF  LDTYPE=MQTERM
                 ,NAME=crn
                 ,APPL=application_name
                 ,LINK=MQ_queue_resource_name
                 ,TERM={1977|1980|2915|4505|3270DSP|3270PRT}
                 [,ASSDEV=name]
                 [,BPALCCHK={NO|YES}]
                 [,BUFSZE=size]
                 [,CRAS={ATnnn|APnnn}]
                 [,CRI=explicit_cri]
                 [,DBCS={NO|YES}]
                 [,ISTATUS={ACTIVE|INACTIVE}]
                 [,LOGON=(YES|NO{,USERID=userid})
                 [,ORD=explicit_ordinal]
                 [,SCRCOL=cols]
                 [,SCRSZE=rows]
                 [,SYSSEND={YES|NO}]
                 [,TAB={NO|YES}]
                 [,TIMEOUT=({50|time1],[300|time2],[3|count1])]
                 [,UPDATE={ADD|DELETE|REPLACE}]
                 [,USERDAT=user_data]
                 [,USER1=user_data]
                                ⋮
                 [,USER16=user_data]
```

Where:

NAME=*crn*
CRN of this resource, 1 to 8 alphanumeric characters. It must be unique within this ALCS system.

LINK=*MQ_queue_resource_name*
The CRN of an MQ queue resource, through which ALCS accesses the terminal. Use COMDEF LDTYPE=MQ to define the MQ queue resource.

TERM={1977|1980|2915|4505|3270DSP|3270PRT}
Terminal device type, where:

Table 14. TERM= parameter values and system defaults for LDTYPE=MQTERM

TERM	Function	SCRSZE	SCRCOL	BUFSZE
1977	ALC keyboard printer			97 bytes
1980	ALC receive-only printer			97 bytes
2915	ALC display	12 lines	64 cols	
4505	ALC display	30 lines	64 cols	
3270DSP	3270-type display	24 lines	80 cols	
3270PRT	3270-type printer			1920 bytes

Use SCRSZE and SCRCOL to override the display size and BUFSZE to override the buffer size.

You can only override the number of columns with TERM=3270DSP.

Note that ALCS does not convert messages to or from 3270 datastream format for TERM=3270DSP or TERM=3270PRT.

For printer devices (TERM=1977, TERM=1980, TERM=3270PRT), the device must send printer acknowledgments to ALCS. These can be either the default ALC format (single EOI character) or another format which can be detected by your installation-wide monitor exit USRCOM8.

ALC display devices (TERM=2915 and TERM=4505) can use the field identifier (FID) sequence. The default format is a 1-byte command character followed by a FID character (hexadecimal 7A).

ALC display devices (TERM=2915 and TERM=4505) can send action key messages to ALCS. The default format is a 1-byte action key character followed by an EOP character.

3270-type display devices (TERM=3270DSP) can send function key messages to ALCS. The default format is a 2-byte function key value plus optional text characters followed by an EOP character.

BPALCCHK={NO|YES}
Indicates whether full ALC character checking is done. If BPALCCHK=NO, full ALC character checking is done. If BPACLCHK=YES, only ACTION KEY processing is done and invalid ALC characters are allowed.

DBCS={NO|YES}
Whether the resource supports the double byte character set or not. DBCS=YES is only allowed with TERM=3270DSP or TERM=3270PRT. Default is DBCS=NO.

SCRCOL=cols
Number of columns on the display terminal. System defaults are:

Table 15. SCRCOL= system defaults for LDTYPE=MQTERM

TERM	Screen size (columns)
2915	64
4505	64
3270DSP	80

You can only override the number of columns with TERM=3270DSP.

SCRSZE=rows
Number of lines on the display terminal. System defaults are:

Table 16. SCRSZE= system defaults for LDTYPE=MQTERM

TERM	Screen size (lines)
2915	12
4505	30
3270DSP	24

TIMEOUT=({50|*time1*],[300|*time2*],[3|*countl*])
Use only for printers where TERM=1977 or TERM=1980.

For other parameters, see Table 13 on page 110; and see also "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

COMDEF parameters for NetView

Use this COMDEF format to define a NetView operator ID to ALCS.

```
[label] COMDEF LDTYPE=NETVIEW
                         ,APPL=application_name
                         ,NAME=operator_ID
                         [,ASSDEV=name]
                         [,CRAS={PRC|ROC|(ATnnn|APnnn[,|NOFALLBACK|FALLBACK])}]
                         [,CRI=explicit_cri]
                         [,ORD=explicit_ordinal]
                         [,RETRV={NO|YES}]
                         [,SYSSEND={YES|NO}]
                         [,TERM={DISPLAY|PRINTER}]
                         [,UPDATE={ADD|DELETE|REPLACE}]
                         [,USERDAT=user_data]
                         [,USER1=user_data]
                         :
                         [,USER16=user_data]
```

Where:

NAME=*operator_ID*
NetView operator ID. Note that ALCS uses this as the CRN of the resource.

CRAS={PRC|ROC|(AT*nnn*|AP*nnn*[,|FALLBACK|NOFALLBACK])}
The terminal has CRAS status. Omit this parameter if the terminal does not have CRAS status. Otherwise specify one of:

PRC
Prime CRAS

ROC
RO CRAS

AT*nnn*
Alternate CRAS, *nnn* is decimal 1 through 255

AP*nnn*
Alternate CRAS printer, *nnn* is decimal 1 through 255

For alternate CRASs and alternate CRAS printers, an optional second subparameter specifies whether or not the terminal is a CRAS fallback candidate, as follows:

FALLBACK
For AT1 through AT16 and AP1 through AP16 only. The terminal is a fallback candidate. Can be abbreviated to F. This is the default for alternate CRASs AT1 through AT16.

NOFALLBACK
For AT1 through AT16 and AP1 through AP16 only. The terminal is not a fallback candidate. Can be abbreviated to NF. This is the default for alternate CRAS printers AP1 through AP16.

TERM={DISPLAY|PRINTER}
The NetView operator ID functions as an input/output terminal (DISPLAY) or as an output only terminal (PRINTER).

For other parameters, see Table 13 on page 110; and see also "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

COMDEF parameters for an other-system terminal

Use this COMDEF format to define a terminal that another system owns and controls.

The system that owns and controls this terminal must:

- Connect to ALCS with a link LDTYPE of either:
  - SLCLINK with PRTCOL=TYPE2 and APPL=ROUTER
  - ALCSLINK
- Be an ALCS or TPF system, or another system which supports a message-router facility compatible with ALCS.

The other system's message router works cooperatively with the ALCS message router to allow communication between this ALCS and the terminal. Figure 34 on page 134 gives an overview of COMDEF parameters used in this section.

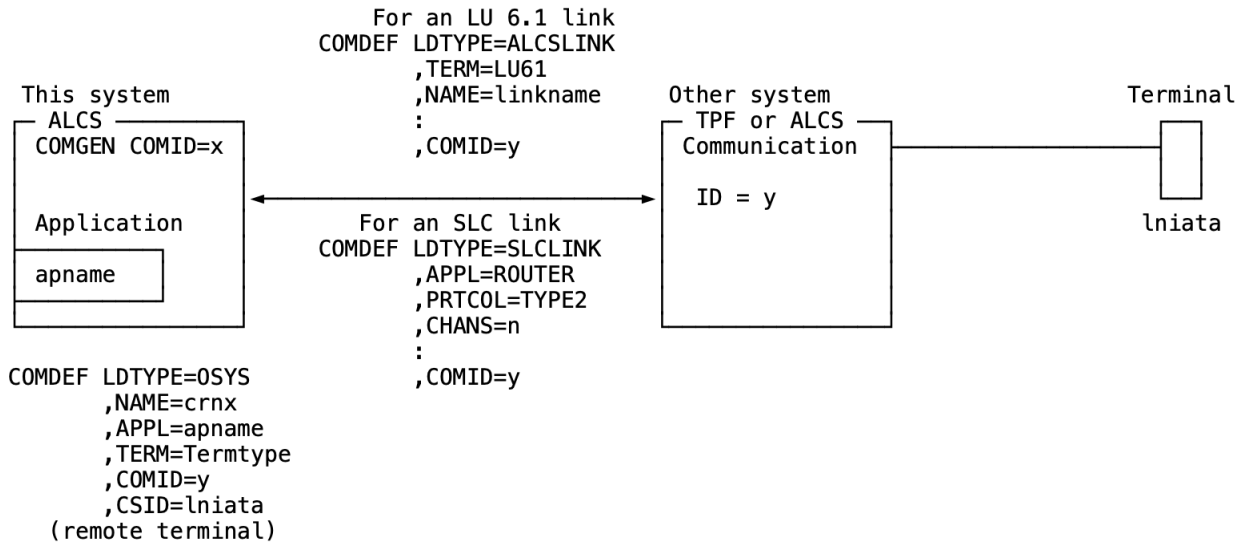


Figure 34. Defining a terminal on another system

ALCS allocates its own unique 3-byte address (CRI) for the terminal. Normally this is not the same as the 3-byte address that the other system uses for the terminal. Use the CSID= parameter to specify the address that the **other system** (which owns this terminal) uses to identify this terminal. The ALCS message router converts the CSID to the ALCS CRI when it receives messages from the terminal, and converts the CRI to the CSID when it sends messages to the terminal.

```
[label}  COMDEF  LDTYPE=OSYS
                          ,NAME=name
                          ,APPL=application_name
                          ,TERM={1977|1980|2915|4505|3270DSP|3270PRT}
                          ,CSID=terminal_id
                          ,COMID=communication_id
                          [,ASSDEV=name]
                          [,BUFSZE={size}]
                          [,CRAS=ATnnn|APnnn]
                          [,CRI=explicit_cri]
                          [,ISTATUS={ACTIVE|INACTIVE}]
                          [,LOGON=(NO[,USERID=userid])]
                          [,ORD=explicit_ordinal]
                          [,RETRV={NO|YES}]
                          [,SCRSZE=rows]
                          [,SYSSEND={YES|NO}]
                          [,TAB={NO|YES}]
                          [,UPDATE={ADD|DELETE|REPLACE}]
                          [,UNSLITE={NO|YES}]
                          [,USERDAT=user_data]
                          [,USER1=user_data]
                          :
                          [,USER16=user_data]
```

Where:

NAME=*name*
CRN of the terminal. An alphanumeric string of 1 to 8 characters; the first character must be alphabetic. *name* must be unique within ALCS. *name* is not the terminal's LU name because the terminal is not on this VTAM network.

TERM={1977|1980|2915|4505|3270DSP|3270PRT}
Terminal device type, where:

Table 17. TERM= parameter values and system defaults for LDTYPE=OSYS			
TERM=	**Function**	**SCRSZE**	**BUFSZE**
1977	ALC keyboard printer		97 bytes
1980	ALC receive-only printer		97 bytes
2915	ALC display	12 lines	
4505	ALC display	30 lines	
3270DSP	ALCS-supported 3270-type display	24 lines	
3270PRT	ALCS-supported 3270-type printer		1920 bytes

Use SCRSZE to override the display size and BUFSZE to override the buffer size.

CSID=
Cross-system ID, the three-byte ID that the **other system** (which owns this terminal) uses to identify this terminal.

It is typically:

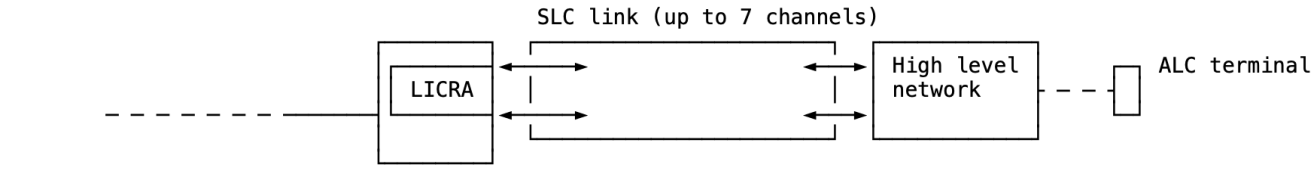
- The CRI on an ALCS system
- The LNIATA on a TPF system.

COMID=*communication_id*
Communication ID of the system that owns this resource. It is 1 alphabetic character, A through Z[5]. This is the same communication ID that you specify on the COMDEF that defines the link (LDTYPE=SLCLINK or LDTYPE=ALCSLINK) connecting the other system.

For other parameters, see Table 13 on page 110; and see also "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

COMDEF parameters for an SLC ALC terminal

Use this COMDEF format to define an ALC terminal (or compatible device) attached to an ATA/IATA high-level network (HLN) that connects to ALCS through an SLC link, PRTCOL=TYPE1 or PRTCOL=TYPE3.




```
COMDEF LDTYPE=SLCLINK,CHANS=n
       ,PRTCOL=TYPE1 or ,PRTCOL=TYPE3
       :
COMDEF LDTYPE=SLCALC
       :
       ,HEX=exitaddr,TCID=hexnum
       ,IA=intaddr,TA=termaddr
```

Figure 35. The COMDEF parameters to define an ALC terminal on an SLC link

The HLN uses a terminal identifier (called an SLC ID in ALCS) to address the terminal. Use the HEX=, TCID=, IA=, and TA= parameters to specify the SLC ID. You must obtain this addressing information from the owner of the HLN. ALCS SLC support converts the SLC ID to the ALCS CRI when it receives messages from the terminal, and converts the CRI to the SLC ID when it sends messages to the terminal.

[5] TPF allows only A through P for its communication ID, which it calls the "CPU-ID".

```
[label}  COMDEF  LDTYPE=SLCALC
                           ,NAME=base_name
                           ,APPL=application_name
                           ,TERM={1977|1980|2915|4505}
                           ,LINK=name
                           ,HEX=HLN_exit_address
                           ,TCID=hex_number
                           ,IA=interchange_address
                           ,TA=terminal_address
                           [,ASSDEV=name]
                           [,BUFSZE={size}]
                           [,CODE={ALC|IATA7|IATA5|NONE}]
                           [,CRAS={ATnnn|APnnn}]
                           [,CRI=explicit_cri]
                           [,CSID=terminal_id]
                           [,ISTATUS={ACTIVE|INACTIVE}]
                           [,LOGON=(NO[,USERID=userid])]
                           [,ORD=explicit_ordinal]
                           [,RETRV={NO|YES}]
                           [,SCRSZE=rows]
                           [,SYSSEND={YES|NO}]
                           [,TAB={NO|YES}]
                           [,TIMEOUT=({50|time1],[300|time2],[3|count1])]
                           [,UNSLITE={NO|YES}]
                           [,UPDATE={ADD|DELETE|REPLACE}]
                           [,USERDAT=user_data]
                           [,USER1=user_data]
                           ⋮
                           [,USER16=user_data]
```

Where:

NAME=*base_name*
Base CRN for the terminal. An alphanumeric string of 1 to 4 characters; the first character must be alphabetic. The stage 1 generation builds the full CRN for the terminal. The terminal CRN is made up of:

- Base name (NAME)
- Two-character interchange address (IA)
- Two-character terminal address (TA).

The full CRN for each terminal must be unique within ALCS.

TERM={1977|1980|2915|4505}
Terminal device type, where:

Table 18. TERM= parameter values and system defaults for LDTYPE=SLCALC

TERM=	Function	SCRSZE	BUFSZE
1977	ALC keyboard printer		97 bytes
1980	ALC receive-only printer		97 bytes
2915	ALC display	12 lines	
4505	ALC display	30 lines	

Use SCRSZE to override the display size and BUFSZE to override the buffer size.

LINK=*name*
name is the CRN of the SLC link through which ALCS accesses the terminal. This link must be PRTCOL=TYPE1 or PRTCOL=TYPE3.

HEX=*HLN_exit_address*
HLN address of the terminal's connection to the HLN, 4 hexadecimal digits. Obtain it from the owner of the HLN.

Note that "HLN exit address" refers to the point where a message leaves the HLN. This parameter specifies the address which is the HLN exit address for outbound (to the terminal) messages. The same address is called the "HLN entry address" when it appears in inbound (from the terminal) messages.

TA=*terminal_address*
Terminal address (TA) of the terminal. It is a 2-digit hexadecimal number. Obtain it from the owner of the HLN.

TIMEOUT=({50|*time1*],[300|*time2*],[3|*countl*])
Use this parameter only for printers where TERM=1977 or TERM=1980. See “COMGEN macro” on page 98 for further information.

For other parameters, see Table 13 on page 110; and “Setting defaults for COMDEF parameters - COMDFLT macro” on page 105.

COMDEF parameters for an SLC link

Use these COMDEF formats to define an SLC link. Use the PRTCOL= parameter to define which **type** of link specification to use. There are three formats, depending on whether the link operates according to the specifications of:

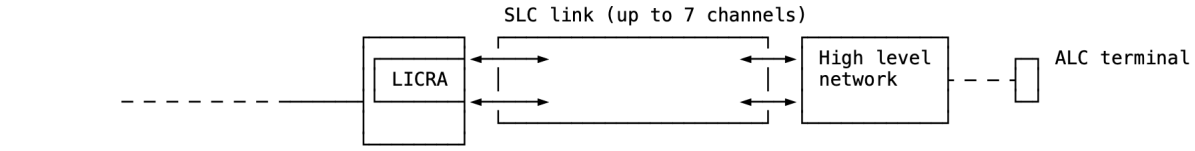
TYPE1
Synchronous Link Control Procedure , SITA document SITA Document P.1124.

TYPE2
Character orientated synchronous link control, ATA/IATA *Systems and Communications Reference Manual (Vols 1-7)*.

Note: The ALCS message router can use SLC type 2 links to communicate with **other-system** resources.

TYPE3
Synchronous Link Control Procedure, SITA document P.1124

Defining a P.1024 or P.1124 SLC link with terminal traffic




```
COMDEF LDTYPE=SLCLINK,CHANS=n
       ,PRTCOL=TYPE1  or ,PRTCOL=TYPE3
       :
```

Figure 36. The COMDEF parameters to define an SLC link with terminal traffic (P.1024 or P.1124)

```
[label]  COMDEF  LDTYPE=SLCLINK
                        ,APPL=application_name
                        ,CHANS=((address1,address2),...)
                        ,HEN=HLN_entry_address
                        ,NAME=name
                        [,PRTCOL={TYPE1|TYPE3}]
                        [,BAUD={4800|bps}]
                        [,CML=({YES},[NO|YES],[NO|YES])]
                        [,CODE={ALC|IATA7|IATA8|NONE}]
                        [,COUNTER=(c1,c2,c3,c4,c5,c6,c7,c8)]
                        [,CRI=explicit_cri]
                        [,ISTATUS={ACTIVE|INACTIVE}]
                        [,LOGON=(NO[,USERID=userid])]
                        [,LOOPBIT=({0|1],[TEST|NOTEST])]
                        [,ORD=explicit_ordinal]
                        [,TIMER=(i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12)]
                        [,UPDATE={ADD|DELETE|REPLACE}]
                        [,USERDAT=user_data]
                        [,USER1=user_data]
                        :
                        [,USER16=user_data]
```

Where:

PRTCOL={TYPE1|TYPE3}
The link operates according to the SITA P.1024 (TYPE1) or P.1124 (TYPE3) specifications. The following parameters are mandatory for these protocols:

APPL=*application_name*
Name of the application to which ALCS routes Type B messages. Use COMDEF LDTYPE=ALCSAPPL to define the application. See "COMDEF parameters for a local application" on page 123. ALCS routes Type A messages to the application specified for the input terminal (see "COMDEF parameters for an SLC ALC terminal" on page 135).

HEN=*HLN_entry_address*
HLN address of ALCS's connection to the HLN. Obtain it from the owner of the HLN.

Note that "HLN entry address" refers to the point where a message enters the HLN. This parameter specifies the address which is the HLN entry address for outbound (from ALCS) messages. The same address is called the "HLN exit address" when it appears in inbound (to ALCS) messages.

Defining an SLC link with no terminal traffic

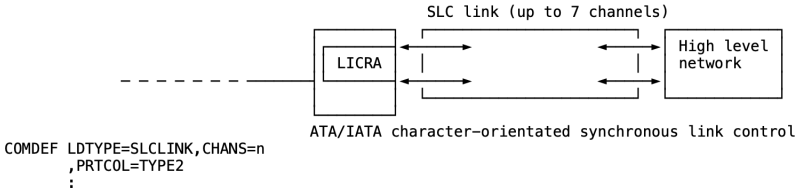


Figure 37. The COMDEF parameters to define an SLC link with no terminal traffic

```
[label]  COMDEF  LDTYPE=SLCLINK
                        ,CHANS=((address1,address2),...)
                        ,NAME=name
                        ,PRTCOL=TYPE2
                        [,APPL={ROUTER|application_name}]
                        [,BAUD={4800|bps}]
                        [,CML=({YES|NO},[NO|YES],[NO|YES])]
                        [,CODE={ALC|IATA7|IATA5|NONE}]
                        [,COMID=communication_id]
                        [,COUNTER=(c1,c2,c3,c4,c5,c6,c7,c8)]
                        [,CRI=explicit_cri]
                        [,ENQRSP={RSM|ACK}]
                        [,ISTATUS={ACTIVE|INACTIVE}]
                        [,LOGON=(NO[,USERID=userid])]
                        [,LOOPBIT=(0|1],[NOTEST|TEST])]
                        [,ORD=explicit_ordinal]
                        [,RETRANS={PART|ALL}]
                        [,STPALL={NO|YES}]
                        [,TIMER=(i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12)]
                        [,UPDATE={ADD|DELETE|REPLACE}]
                        [,USERDAT=user_data]
                        [,USER1=user_data]
                        :
                        [,USER16=user_data]
```

PRTCOL=TYPE2
The link operates according to the character orientated synchronous link control as defined by ATA/IATA *Systems and Communications Reference Manual (Vols 1-7)*. Specify PRTCOL=TYPE2 if the message router uses the link.

The following parameters are optional for this protocol:

APPL={ROUTER|*application_name*}

ROUTER
Only the message router component of ALCS uses this link.

application_name
Name of the application to which ALCS routes input messages.

COMID=*communication_id*
Identification of the system to which this link connects (for message router). It is 1 alphabetic character, A through Z[6]. This parameter is mandatory when APPL=ROUTER is specified or defaulted.

[6] TPF only allows A through P as its CPU-ID.

ENQRSP={RSM|ACK}
Type of LCB used as response for an ENQ or ILB:

RSM
ALCS accepts either RSM or STP.

ACK
ALCS accepts ACK, RSM, or STP.

ISTATUS={ACTIVE|INACTIVE}
Status of the SLC link when ALCS starts:

ACTIVE
ALCS opens and starts the SLC link when it changes system state above idle state. If you want to use the alternative channel addresses for one or more channels in an SLC link, you must open the SLC link in idle state with the appropriate command before commencing state changes (see the *ALCS Operation and Maintenance*).

INACTIVE
ALCS opens and starts the SLC link only in response to ALCS commands.

Note: ALCS stops and closes all SLC links when it changes system state below normal state.

RETRANS={PART|ALL}
Type of message retransmission after an I/O error occurs during the transmission of a message (if another SLC channel on the same link is available):

PART
Retransmit only the blocks that are not yet successfully transmitted.

ALL
Retransmit the complete message.

STPALL={NO|YES}

NO
ALCS does not send STA to stop all channels on a link. Instead, it sends an individual STP for each channel.

YES
ALCS can send STA to stop all channels of a link.

Use the following parameters as required:

CHANS=((*address1,address2*), ...)
1 to 7 pairs of MVS I/O device addresses, where each address is an EP subchannel address. A pair of addresses *(address1,address2)* specifies the addresses of one full-duplex SLC channel:

address1
Address of the output (transmit) SLC line

address2
Address of the input (receive) SLC line.

Both *address1* and *address2* can have either of the following formats:

address
Transmit or receive address

(prime_address,alternate_address)
Prime address followed by the alternate address (transmit or receive).

NAME=*name*
CRN of the link. An alphanumeric string of 1 to 7 characters; the first character must be alphabetic. *name* must be unique within ALCS. Unlike other resource types, an SLC link CRN is restricted to 7 characters. The CRNs of the channels in this link are created by adding the channel number (1 through 7) to the end of the CRN of the link.

The following parameters are optional for all protocols:

BAUD={4800|*bps*}
Speed, in bits per second, of the lines that constitute this link. All channels in a link are assumed to run at the same rate. *bps* is a decimal number. Default values for the TIMER parameter depend on this parameter.

CML=([NO|YES],[NO|YES],[NO|YES])
The three subparameters let you specify whether or not to exchange clear message label (CML) LCBs:

1. On the SLC link (NO is an error with PRTCOL=TYPE1 and PRTCOL=TYPE3)
2. For single block Type A messages
3. For single block Type B messages.

```
For PRTCOL=TYPE1 the default is CML=(YES,NO,NO)
For PRTCOL=TYPE2 the default is CML=(YES,NO,NO)
For PRTCOL=TYPE3 the default is CML=(YES,NO,YES),
```

COUNTER=(*c1,c2,c3,c4,c5,c6,c7,c8*)
Value, in decimal, of various counters that control the link. These values correspond to either:

- The P.1024 and P.1124 Z*n* counters (see *Synchronous Link Control Procedure*, SITA document P.1x24)
- The ATA/IATA N*n* counters (see the ATA/IATA *Systems and Communications Reference Manual (Vols 1-7)*).

Table 19 on page 140 shows the relationship between ALCS, SITA, and ATA/IATA counter values:

Table 19. Counters used for SLC links

Value	Purpose	Default	P.1x24	ATA/IATA
c1	Minimum number of character synchronization characters (SYNs) preceding any block. Specify a value between 2 and 16.	2	Z1	N1
c2	Number of negative-acknowledgment LCBs (NAKs) before declaring a channel down. The minimum value is 1.	3	Z4	N2
c3	Reserved. ALCS does not implement this counter; it ignores the value specified.	0	-	-
c4	Reserved. ALCS does not implement this counter; it ignores the value specified.	0	-	-
c5	Number of enquiry LCBs (ENQs) before declaring a channel down. The minimum value is 1.	3	Z3	N5
c6	Maximum number of outstanding data blocks. Specify a value between 1 and 16.	5	Z2	-
c7	Number of stop LCB (STP) repetitions. The minimum value is 1.	3	Z5	-
c8	Maximum number of blocks sent or received before an acknowledge LCB (ACK) is expected or sent. This corresponds to the ATA/IATA "frequency of positive acknowledgment of protected information blocks". Omit this value for PRTCOL=TYPE1 and PRTCOL=TYPE3.	1	-	-

LOOPBIT=({0|1],[NOTEST|TEST])

Value and use of the looptest bit in the envelope of LCBs. Decide this by agreement between the two communicating parties.

```
For PRTCOL=TYPE1 the default is LOOPBIT=(0,TEST)
For PRTCOL=TYPE2 the default is LOOPBIT=(0,NOTEST)
For PRTCOL=TYPE3 the default is LOOPBIT=(0,TEST),
```

0

ALCS sets the looptest bit to 0 in LCBs that it transmits.

1

ALCS sets the looptest bit to 1 in LCBs that it transmits.

TEST

ALCS tests the looptest bit in LCBs that it receives. If ALCS sets the bit to 1 in output LCBs, the bit must be 0 in input LCBs; if ALCS sets it to 0 in output LCBs, it must be 1 in input LCBs.

NOTEST

ALCS does not test the looptest bit in LCBs that it receives.

TIMER=(*i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12*)

Value, in seconds and tenths of a second, of various time intervals that control the link. The default values depend on the value specified for the BAUD parameter. ALCS rounds down the specified value to the nearest one fifth of a second. These values correspond to the P.1024 S*n* timers and the P.1124 S*n* and TT*n* timers (see the *Synchronous Link Control Procedure*, SITA document P.1x24), and to the ATA/IATA T*n* timers (see the ATA/IATA *Systems and Communications Reference Manual (Vols 1-7)*. Table 20 on page 141 shows the relationship between ALCS, SITA, and ATA/IATA timer values:

Table 20. Timer values used for SLC links

Value	Purpose	Default Slow/Fast	P.1x24	ATA/IATA
i1	"Idle output line" timer value.	3 2	S6	T1
i2	"Idle link control block repetition" timer value.	3 2	-	T2
i3	"NAK repetition" timer value.	3.5 2.5	S2	T3
i4	"Enquiry link control block repetition" timer value.	3 2	S3	T4
i5	"No ACK received" timer value.	3 2	S1	T5
i6	"No CML received" timer value.	9 6	S9, TT1	T6
i7	Reserved. ALCS does not implement this timer; it ignores the value specified.	0 0	-	-
i8	"Multiblock message" timer value.	60 60	S10, TT2	T8
i9	"No block received" timer value.	4 3	S7	T9
i10	"Stop link control block" repetition timer value.	3 2	S5	-
i11	"Resume link control block" repetition timer value.	3 2	S4	-
i12	"Channel down" timer value.	15 10	S8	-

The fast default applies when BAUD is greater than 2400.
Use the recommended values in the column headed "P.1x24." when the SLC link is connected to the SITA high-level network.

For other parameters, see Table 13 on page 110; and see “Setting defaults for COMDEF parameters - COMDFLT macro” on page 105.

COMDEF parameters for a virtual SLC link used to address an X.25 PVC or TCP/IP resource.

Use this COMDEF format to define a virtual SLC link that is used to address an X.25 PVC or TCP/IP resource for transmitting and receiving Type B traffic.

You can also use COMDEF if you have a host-to-host application, originally designed for use with an SLC link, but which you now want to use with an X.25 link or TCP/IP resource. In this case, the host-to-host application uses the address (CRI) of this virtual SLC link.

Applications such as the IPARS Message Switching Package address communication resources by means of a symbolic line number (SLN) instead of a CRI. A virtual SLC link allows these applications to convert an SLN address to a CRI address of the associated X.25 PVC or TCP/IP resource, using the ALCS COMIC monitor-request macro. Note that the SLN of an SLC link is in the low-order byte of its CRI.

See Appendix C, “Sample symbolic line number conversion,” on page 550 for samples.

```
[label}  COMDEF  LDTYPE=SLCLINK
                           ,NAME=name
                           ,X25NAME=xname|(LINK=tname
                           [,CRI=explicit_cri]
                           [,LOGON=(NO[,USERID=userid])]
                           [,ORD=explicit_ordinal]
                           [,UPDATE={ADD|DELETE|REPLACE}]
                           [,USERDAT=user_data]
                           [,USER1=user_data]
                           ⋮
                           [,USER16=user_data]
```

Where:

NAME=*name*
CRN of the link. An alphanumeric string of 1 to 7 characters; the first character must be alphabetic. *name* must be unique within ALCS.

X25NAME=*xname*
CRN of the X.25 PVC that is associated with this virtual SLC link. The X.25 PVC must be PRTCOL=TYPE2 or PRTCOL=TYPE3.

LINK=*tname*
CRN of the TCP/IP resource that is associated with this virtual SLC link. The TCP/IP resource must be TERM=(CLIENT,MATIPB).

For other parameters, see Table 13 on page 110; and see “Setting defaults for COMDEF parameters - COMDFLT macro” on page 105.

COMDEF parameters for TCP/IP

Use these COMDEF formats to define a TCP/IP connection between this ALCS and another TCP/IP host. ALCS obtains TCP/IP services from the Communications Server IP product which resides on the same MVS image as ALCS.

z/OS Communications Server IP API Guide publications provide information about planning, configuring, activating, customizing, and maintaining Communications Server IP.

For a TCP/IP connection where the ALCS application is the client:

```
[label}  COMDEF LDTYPE=TCPIP
                ,NAME=crn
                ,APPL=appl
                ,RHOST=({IP_address1],[IP_address2],[IP_address3],[IP_address4])
                ,RPORT=remote_port_number
                ,TERM={CLIENT|(CLIENT,application_protocol[,hth])}
                [,CODE={ALC|IATA7|IATA5|NONE}]
                [,CRI=explicit_cri]
                [,FLOWID=flw]
                [,HDR=hdr]
                [,HEN=high-level_designator]
                [,HEX=high-level_designator]
                [,IPMGSZ={a|K|bM}]
                [,ISTATUS={ACTIVE|INACTIVE}]
                [,LOGON=(NO[,USERID=userid])]
                [,MPX=mpx]
                [,ORD=explicit_ordinal]
                [,RPORTMATCH={NO|YES}]
                [,SPORT=server_port_number]
                [,SPORTMATCH={NO|YES}]
                [,TIMEOUT=({0|time1],[0|time2],[SSQ])]
                [,TRANSLATE={YES|NO}]
                [,UPDATE={ADD|DELETE|REPLACE}]
                [,USERDAT=user_data]
                [,USER1=user_data]
                ⋮
                [,USER16=user_data]
```

Where:

NAME=*crn*
CRN of this TCP/IP resource, 1 to 8 alphanumeric characters. It must be unique within this ALCS system.

RHOST=({*IP_address1*],[*IP_address2*],[*IP_address3*],[*IP_address4*])
The TCP/IP remote IP address where your application obtains services. ALCS connects to the server using this IP address. A string of 1 to 15 characters in dotted decimal notation, for example 9.180.214.187 .

If you specify more than one IP address then ALCS tries to connect using each address each in turn; this may be useful if the remote system uses primary and alternate IP addresses for the service.

For TCP/IP connections that implement MATIP Type A or Type B protocol, ALCS compares each IP address with the remote host for inbound connection requests; this may be useful if the remote system is responsible for starting the connection and it uses primary and alternate IP addresses.

RPORT=*remote_port_number*
The TCP/IP remote port number where your application obtains services. ALCS connects to the server using this port number. A decimal number from 1 to 65 534.

TERM={CLIENT|(CLIENT,*application_protocol*[,*hth*])}
ALCS applications use this connection to request services from a remote server application.

Specify *application_protocol* if you want ALCS to check whether a complete message has been received before passing it to the application.

Specify TERM=(CLIENT,MATIPB) for TCP/IP connections that implement MATIP (Mapping of Airline Traffic over IP) Type B.

- If ALCS should act as the MATIP Type B client then use the ZACOM command to start this TCP/IP client resource.
- If ALCS should act as the MATIP Type B server then define another TCP/IP resource with COMDEF TERM=(SERVER,MATIPB). When a MATIP Type B host connects to ALCS then ALCS automatically starts this TCP/IP client resource if its RHOST parameter matches the IP address of the remote host, its RPORT parameter matches the port number of the remote host (if RPORTMATCH=YES), and its SPORT parameter matches the local port number of the TCP/IP server resource (if SPORTMATCH=YES).

Specify **TERM**=(CLIENT,MATIPA,*hth*) for TCP/IP connections that implement MATIP (Mapping of Airline Traffic over IP) Type A host-to-host. *hth* is one of:

HTHI
IATA host-to-host (default).

HTHS
SITA host-to-host.

- If ALCS should act as the MATIP Type A client then use the ZACOM command to start this TCP/IP client resource.
- If ALCS should act as the MATIP Type A server then define another TCP/IP resource with COMDEF TERM=(SERVER,MATIPA). When a remote MATIP Type A host connects to ALCS then ALCS automatically starts this TCP/IP client resource if its RHOST parameter matches the IP address of the remote host, its RPORT parameter matches the port number of the remote host (if RPORTMATCH=YES), and its SPORT parameter matches the local port number of the TCP/IP server resource (if SPORTMATCH=YES).

Specify TERM=(CLIENT,MATIPA) for TCP/IP connections that implement MATIP (Mapping of Airline Traffic over IP) Type A conversational traffic.

- If ALCS should act as the MATIP Type A client then use the ZACOM command to start this TCP/IP client resource.
- If ALCS should act as the MATIP Type A server then define another TCP/IP resource with COMDEF TERM=(SERVER,MATIPA). When a remote MATIP Type A host connects to ALCS then ALCS automatically starts this TCP/IP client resource if its RHOST parameter matches the IP address of the remote host, its RPORT parameter matches the port number of the remote host (if RPORTMATCH=YES), and its SPORT parameter matches the local port number of the TCP/IP server resource (if SPORTMATCH=YES).

CODE={ALC|IATA7|IATA5|NONE}
Translation code for MATIP sessions whenever ALCS opens the session.

FLOWID=*flw*
MATIP flow ID for the session. It is 2 hexadecimal digits. This TCP/IP connection must be TERM=(CLIENT,MATIPA,HTHI) or TERM=(CLIENT,MATIPA,HTHS).

HDR=*hdr*
MATIP header type for the sessions.
This TCP/IP connection must be TERM=(CLIENT,MATIPA),,TERM=(CLIENT,MATIPA,HTHI) or TERM=(CLIENT,MATIPA,HTHS).

For Type A conversational sessions, one of:

10
No ASCU header

01
ASCU Header is A1 + A2

00
ASCU header is H1 + H2 + A1 + A2 (default)

For Type A host-to-host sessions, one of:

10
No header (default)

01
Header is flow ID

00
Header is session ID + flow ID

HEN=*high-level_designator*
For TCP/IP connections that implement MATIP Type B, the high-level designator of ALCS's connection to the HLN. Obtain it from the owner of the HLN. It is 4 hexadecimal digits.

HEX=*high-level_designator*
For TCP/IP connections that implement MATIP Type B, the high-level designator of the remote Message Switching host system. Obtain it from the owner of the HLN.

For TCP/IP connections that implement MATIP Type A host-to-host, this is the MATIP H1,H2 address value that identifies the remote host system. It must be allocated bilaterally.

It is 4 hexadecimal digits.

IPMGSZ=*a*|K|*b*M
The maximum input or output message size for large messages on a TCP/IP connection defined with TERM=CLIENT or TERM=(CLIENT,MATIPA). Specify the size in megabytes (M) or kilobytes (K). The smallest size for large messages is 32K. The largest size is 2048K or 2M for TERM=CLIENT and 63K for TERM=(CLIENT,MATIPA). If you omit this parameter then large messages are not allowed for this TCP/IP connection.

When IPMGSZ is specified for a TCP/IP connection, ALCS allows input and output messages which are larger than any of the storage block sizes defined in your system. It does this by using heap storage instead. To enable this, you must allocate type 3 storage units using the SCTGEN NBRSU and SUSIZE parameters in your system configuration. Specify a type 3 storage unit size which is greater than or equal to the value of IPMGSZ plus 21 bytes (to allow for a message header). The number required will depend on factors such as the rate of large messages sent and received by your applications, and the existence time of entries.

When IPMGSZ is specified for a TCP/IP connection, you must also specify a MEMLIMIT for the ALCS job or started task. See Chapter 2, “Installing ALCS,” on page 6 for details.

MPX=*mpx*
MATIP multiplex type for the session.
This TCP/IP connection must be TERM=(CLIENT,MATIPA),,TERM=(CLIENT,MATIPA,HTHI) or TERM=(CLIENT,MATIPA,HTHS).
For Type A conversational sessions, one of:

10
Single ASCU

01
Group of ASCUs with 2-byte identification

00
Group of ASCU with 4-byte identification (default)

For Type A host-to-host sessions, one of:

10
Single flow (default)

01
Multiple flows

This TCP/IP connection must be TERM=(CLIENT,MATIPA,HTHI) or TERM=(CLIENT,MATIPA,HTHS).

RPORTMATCH={NO|YES}
Use this parameter for TCP/IP client resources that implement MATIP Type A or Type B protocol and the connection can be started by a remote system.

When the TCP/IP client resource is started by a remote host, the RPORT and RPORTMATCH parameters can help to distinguish between services on the remote host which use the same protocol (for example MATIP Type A) and share the same remote host IP address.

When the TCP/IP client resource is started by ALCS (using the ZACOM command or COMDEF ISTATUS=ACTIVE), the RPORTMATCH parameter has no effect.

“Type B message support using TCP/IP” on page 42 and “Type A host-to-host message support using TCP/IP” on page 44 provide more information about managing TCP/IP connections.

SPORT=*server_port_number*
The TCP/IP local port number where your application provides services. A decimal number from 1 to to 65 534. Use this parameter - in conjunction with the SPORTMATCH parameter - for TCP/IP client resources that implement MATIP Type A or Type B protocol and the connection can be started by a remote system.

SPORTMATCH={NO|YES}
Use this parameter - in conjunction with the SPORT parameter - for TCP/IP client resources that implement MATIP Type A or Type B protocol and the connection can be started by a remote system.

When the TCP/IP client resource is started by a remote host, the SRPORT and SRPORTMATCH parameters can help to distinguish between services on the remote host which use the same protocol (for example MATIP Type A) and share the same remote host IP address.

When the TCP/IP client resource is started by ALCS (using the ZACOM command or COMDEF ISTATUS=ACTIVE), the SPORT and SPORTMATCH parameters have no effect.

"Type B message support using TCP/IP" on page 42 and "Type A host-to-host message support using TCP/IP" on page 44 provide more information about managing TCP/IP connections.

TIMEOUT=({0|*time1*],[0|*time2*],[SSQ])
Controls how ALCS performs error recovery for TCP/IP connections.

[0|*time1*]
Idle connection timeout. If there is no more input data from a TCP/IP connection during this time interval, ALCS closes the connection. (But see the description of the SSQ parameter below.) A decimal number of seconds in the range 1 through 32 767. 0 (or NONE) means there is no idle connection timeout; this is the default.

[0|*time2*]
Blocked send timeout. If the TCP/IP stack does not accept any output data during this time interval, ALCS conditionally calls installation-wide monitor exit USRTCPA.

If USRTCPA is not loaded or USRTCPA returns with condition R15=0, ALCS restarts the blocked send timeout and continues trying to transmit data. If USRTCPA returns with condition R15=4, ALCS closes the connection and puts an information message on the diagnostic sequential file. A decimal number of seconds in the range 1 through 32 767. 0 (or NONE) means there is no blocked send timeout; this is the default.

SSQ
MATIP session status support. Intended for TCP/IP connections that implement MATIP Type A or Type B. When the idle connection timeout expires, and SSQ is specified, ALCS transmits a MATIP Session Status Query (SSQ) packet and restarts the idle connection timeout. After transmitting three SSQ packets without receiving any input data, ALCS closes the connection.

TRANSLATE={YES|NO}
Intended for TCP/IP connections that implement MATIP Type A host-to-host. Specify TRANSLATE=NO on the connection defined as **TERM**=(CLIENT,MATIPA,*hth*) in order to bypass any data translation for inbound and outbound messages on this connection. This can be useful if the MATIP session specifies a non-EBCDIC coding, but the data translation must be done by your application programs rather than by ALCS.

Internet RFC (Request for Comments) number 2351 describes the MATIP protocol.

For a TCP/IP connection where the ALCS application is the server:

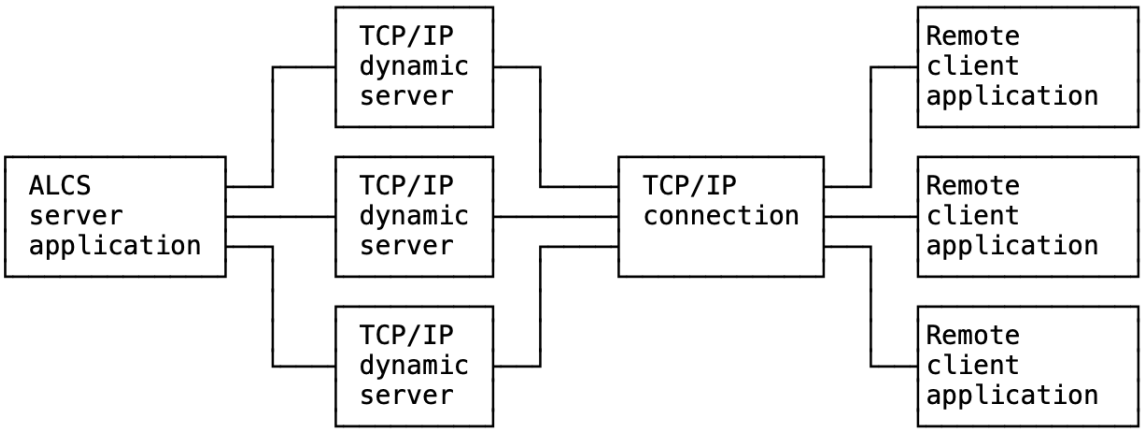


Figure 38. Defining a TCP/IP server connection

```
[label}  COMDEF  LDTYPE=TCPIP
                ,NAME=crn
                ,APPL=appl
                ,TERM={SERVER|(SERVER,application_protocol)}
                ,LPORT=local_port_number
                [,CRI=explicit_cri]
                [,IPMGSZ={a|K|bM}]
                [,ISTATUS={ACTIVE|INACTIVE}]
                [,LOGON=(NO[,USERID=userid])]
                [,MAXCONN={256|n}]
                [,ORD=explicit_ordinal]
                [,TIMEOUT=({0|time1],[0|time2],[SSQ])]
                [,UPDATE={ADD|DELETE|REPLACE}]
                [,USERDAT=user_data]
                [,USER1=user_data]
                :
                [,USER16=user_data]
```

Where:

NAME=*crn*
CRN of this TCP/IP resource. 1 to 8 alphanumeric characters. It must be unique within this ALCS system.

TERM={SERVER|(SERVER,*application_protocol*)}
An ALCS application uses this connection to provide services to remote client applications.

Specify *application_protocol* if you want ALCS to check whether a complete message has been received before passing it to the application.

Specify TERM=(SERVER,HTTP) for TCP/IP connections that implement HTTP (Hypertext Transfer Protocol).

APPL=*appl*
CRN of an ALCS application. 4 alphanumeric characters. ALCS passes any messages received over this TCP/IP connection to the input edit program for this application.

Specify TERM=(SERVER,MATIPA) for TCP/IP connections that implement MATIP (Mapping of Airline Traffic over IP) Type A.

APPL=*appl*
ALCS ignores this parameter. ALCS routes Type A messages to the application specified for the input terminal (see "Parameters for an ALC device on TCP/IP" on page 149).

Specify TERM=(SERVER,MATIPB) for TCP/IP connections that implement MATIP (Mapping of Airline Traffic over IP) Type B.

APPL=*appl*
ALCS ignores this parameter. ALCS sends and receives Type B messages using a TCP/IP connection defined with `COMDEF TERM=(CLIENT,MATIPB)`.

When ALCS loads a communication load module containing a `TERM=SERVER` definition, ALCS dynamically creates one or more new communication table entries ("TCP/IP dynamic servers") to manage the client connections. The number of dynamic server communication table entries is determined by the `COMDEF MAXCONN` parameter.

Each TCP/IP dynamic server communication resource has a unique CRN, CRI, and ordinal number. Its CRN is made up of:

- 1 non-alphabetic character:

  "0" if `TERM=SERVER`
  "1" if `TERM=(SERVER,HTTP)`
  "3" if `TERM=(SERVER,MATIPA)`
  "4" if `TERM=(SERVER,MATIPB)`
- 1 character "I"
- 6 hexadecimal digits from its CRI.

For example, a TCP/IP dynamic server with a CRI 060123 can have a CRN 0I060123.

LPORT=*local_port_number*
The TCP/IP local port number where your application provides services. Clients use this port number when attempting to connect to ALCS. A decimal number from 1 to 65 534.

In order to use TCP/IP connections, you must:

1. Enable the ALCS support for TCP/IP (see "SCTGEN macro" on page 72).
2. Establish a connection between ALCS and a TCP/IP address space in the same MVS system (see ZCTCP in *ALCS Operation and Maintenance*).

IPMGSZ=*a*|K|*b*M
The maximum input or output message size for large messages on a TCP/IP connection defined with `TERM=SERVER` or `TERM=(SERVER,MATIPA)`. Specify the size in megabytes (M) or kilobytes (K). The smallest size for large messages is 32K. The largest size is 2048K or 2M for `TERM=SERVER` and 63K for `TERM=(SERVER,MATIPA)`. If you omit this parameter then large messages are not allowed for this TCP/IP connection.

When `IPMGSZ` is specified for a TCP/IP connection, ALCS allows input and output messages which are larger than any of the storage block sizes defined in your system. It does this by using heap storage instead. To enable this, you must allocate type 3 storage units using the `SCTGEN NBRSU` and `SUSIZE` parameters in your system configuration. Specify a type 3 storage unit size which is greater than or equal to the value of `IPMGSZ` plus 21 bytes (to allow for a message header). The number required will depend on factors such as the rate of large messages sent and received by your applications, and the existence time of entries.

When `IPMGSZ` is specified for a TCP/IP connection, you must also specify a MEMLIMIT for the ALCS job or started task. See Chapter 2, "Installing ALCS," on page 6 for details.

ISTATUS={ACTIVE|INACTIVE}
Initial status of the TCP/IP resource, where:

ACTIVE
ALCS will accept connection requests from clients after ALCS restart is complete.

INACTIVE
ALCS will not accept connection requests from clients (the ALCS operator must use the `ZACOM ACTIVATE` command to allow connection requests to be accepted).

MAXCONN={256|*n*}
Maximum number of concurrent connections allowed for a TCP/IP resource defined with `TERM=SERVER`. A decimal number in the range 1 through 4000.

TIMEOUT=({0|*time1*],[0|*time2*],[SSQ])

Controls how ALCS performs error recovery for TCP/IP connections.

[0|*time1*]

Idle connection timeout. If there is no more input data from a TCP/IP connection during this time interval, ALCS closes the connection. (But see the description of the SSQ parameter below.) A decimal number of seconds in the range 1 through 32 767. 0 (or NONE) means there is no idle connection timeout; this is the default.

[0|*time2*]

Blocked send timeout. If the TCP/IP stack does not accept any output data during this time interval, ALCS conditionally calls installation-wide monitor exit USRTCPA.

If USRTCPA is not loaded or USRTCPA returns with condition R15=0, ALCS restarts the blocked send timeout and continues trying to transmit data. If USRTCPA returns with condition R15=4, ALCS closes the connection and puts an information message on the diagnostic sequential file. A decimal number of seconds in the range 1 through 32 767. 0 (or NONE) means there is no blocked send timeout; this is the default.

SSQ

MATIP session status support. Intended for TCP/IP connections that implement MATIP Type A or Type B. When the idle connection timeout expires, and SSQ is specified, ALCS transmits a MATIP Session Status Query (SSQ) packet and restarts the idle connection timeout. After transmitting three SSQ packets without receiving any input data, ALCS closes the connection.

Parameters for an ALC device on TCP/IP

Use this COMDEF format to define an ALC terminal (or compatible device) that connects to ALCS through TCP/IP.

If the terminal is attached to an ATA/IATA high-level network (HLN) then use the HEX, TCID, IA, and TA parameters to specify the terminal address. You must obtain this addressing information from the owner of the HLN. Use the optional RHOST parameter to specify the IP address of a remote gateway for the terminal.

When the terminal is used with MATIP Type A conversational traffic, the HEX parameter corresponds to the MATIP H1 and H2 address bytes and the TCID and IA parameters correspond to the MATIP A1 and A2 address bytes.

```
[label}  COMDEF  LDTYPE=TCPIPALC
                          ,NAME=crn
                          ,APPL=application_name
                          ,TERM={1977|1980|2915|4505}
                          [,ASSDEV=name]
                          [,BUFSZE=size]
                          [,CODE={ALC|IATA7|IATA5||NONE}]
                          [,CRAS={ATnnn|APnnn}]
                          [,CRI=explicit_cri]
                          [,CSID=terminal_id]
                          [,HEX=HLN_exit_address]
                          [,IA=interchange_address]
                          [,ISTATUS={ACTIVE|INACTIVE}]
                          [,LINK=name]
                          [,LOGON=({YES|(NO[,USERID=userid]})]
                          [,ORD=explicit_ordinal]
                          [,RETRV={NO|YES}]
                          [,RHOST=remote_IP_address|
                           RHOST=([IP_address1],
                          [IP_address2],[IP_address3],[IP_address4])]
                          [,SCROLL={YES|NO}]
                          [,SCRSZE=rows]
                          [,SYSSEND={YES|NO}]
                          [,TA=terminal_address]
                          [,TAB={NO|YES}]
                          [,TCID=hex_number]
                          [,TIMEOUT=({50|time1],[300|time2],[3|count1])]
                          [,UNSLITE={NO|YES}]
                          [,UPDATE={ADD|DELETE|REPLACE}]
                          [,USERDAT=user_data]
                          [,USER1=user_data]
                          ⋮
                          [,USER16=user_data]
```

Where:

NAME=*crn*
CRN of the terminal. 1 to 8 alphanumeric characters. It must be unique within this ALCS system.

TERM={1977|1980|2915|4505}
Terminal device type, where:

Table 21. TERM= parameter values and system defaults for LDTYPE=X25ALC

TERM=	Function	SCRSZE	BUFSZE
1977	ALC keyboard printer		97 bytes
1980	ALC receive-only printer		97 bytes
2915	ALC display	12 lines	
4505	ALC display	30 lines	

Use SCRSZE to override the display size and BUFSZE to override the buffer size.

SCROLL={YES|NO}]

Yes
Allow scrolling to different parts of those responses that are too large to fit on a single screen. This is the default, when the ALCS scrolling package is used.

No
Inhibit scrolling and send the complete unformatted response to this communication resource. When connected to ALCS using a MATIP TYPE A TCP/IP connection, the response can be one or multiple blocks. When multiple blocks are returned, EOMp characters terminate the intermediate blocks and an EOMc character terminates the final block. The application that constructs the response may only use the DISPC ADD and SEND macros. The DISPC SEND DEST parameter is not allowed and the DISPC SEND START, NUM, SPACE, BOTTOM, LINE, CONFIRM,CLEAR parameters

are ignored. Heap storage is used for the response. You must allocate type 3 storage units to allow for the additional heap storage needed. You should specify a type 3 storage unit size greater than or equal the largest response plus 50 bytes (to allow for message headers and trailers).

RHOST=*remote_IP_address*
RHOST=(*{IP_address1},[IP_address2],[IP_address3],[IP_address4]*)
The IP address of the gateway or addresses of gateways to which this terminal is attached. A string of 1 to 15 characters in dotted decimal notation, for example 9.180.214.187 or (9.180.214,187,9.180.214.188).

You can specify a single IP address as **RHOST**=*IP_address*.

If you specify more than one IP address then ALCS compares each address one after then other. This comparison can be useful if the remote gateway uses primary and alternate IP addresses.

LINK=*name*
The CRN of the TCP/IP connection through which ALCS accesses the terminal. This connection must be TERM=SERVER or TERM=CLIENT.

For other parameters, see Table 13 on page 110; and see also "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

COMDEF parameters for a NEF or ALCI LU

Use this COMDEF format to define a NEF[7] or ALCI LU.

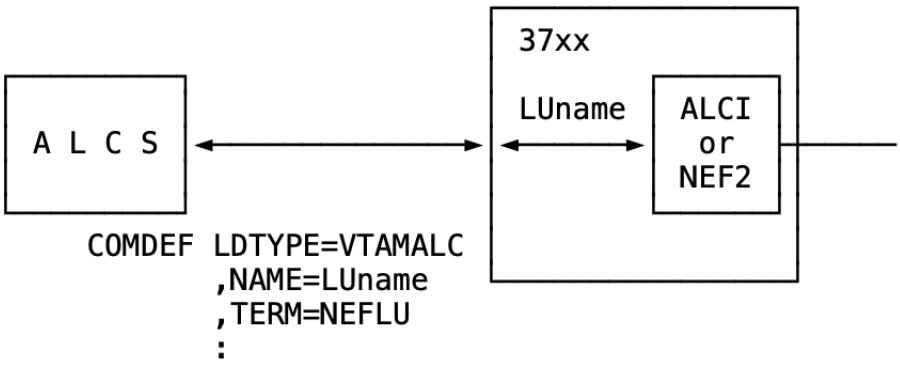


Figure 39. The COMDEF parameters to define a NEF or ALCI LU

```
[label}  COMDEF  LDTYPE=VTAMALC
                          ,NAME=lu_name
                          ,TERM=NEFLU
                          [,CRI=explicit_cri]
                          [,ISTATUS={ACTIVE|INACTIVE}]
                          [,ORD=explicit_ordinal]
                          [,UPDATE={ADD|DELETE|REPLACE}]
                          [,USERDAT=user_data]
                          [,USER1=user_data]
                          ⋮
                          [,USER16=user_data]
```

Where:

[7] COMDEF continues to support the parameters for defining an ALC terminal (or group of terminals) that connect through an IBM 3705 Communication Controller with ACF/NCP and PRPQ P09021 (sometimes called NEF1). This way of connecting ALC terminals is now obsolete, and this book no longer describes NEF1 parameters. Contact your IBM representative if you plan to use NEF1 with ALCS.

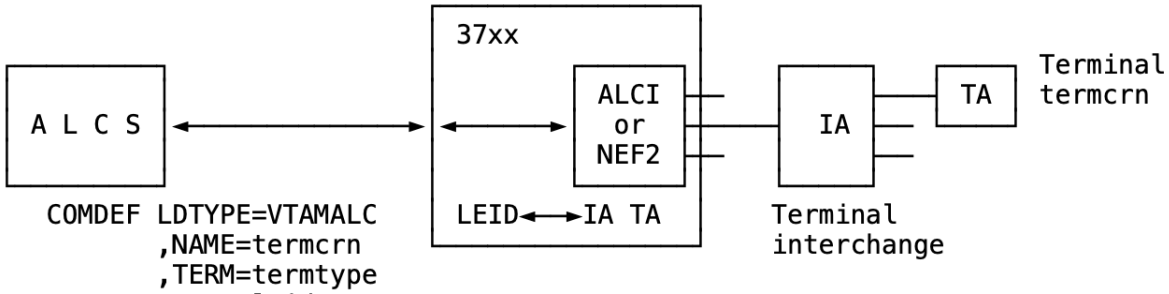
NAME=*lu_name*
The LU name of the NEF or ALCI LU. An alphanumeric string of 1 to 8 characters; the first character must be alphabetic.

TERM=NEFLU
A NEF or ALCI LU.

For other parameters, see Table 13 on page 110; and see “Setting defaults for COMDEF parameters - COMDFLT macro” on page 105.

COMDEF parameters for a NEF or ALCI ALC terminal

Use this COMDEF format to define an ALC terminal (or compatible device) that connects to ALCS through NEF[8] or ALCI and VTAM. The resource is an ALC terminal accessed and controlled by NEF or ALCI in conjunction with ALCS and VTAM. See also “COMDEF parameters for a NEF or ALCI LU” on page 151.




```
COMDEF LDTYPE=VTAMALC
       ,NAME=termcrn
       ,TERM=termtype
       ,LEID=leid
       :
       ,NEFLU=LUname (optional,
Parameters of the COMDEF macro describes the action of NEFLU parameter)
```

1. “COMDEF macro” on page 107

Parameters of the COMDEF macro

Figure 40. The COMDEF parameters to define a terminal on NEF or ALCI

[8] COMDEF continues to support the parameters for defining an ALC terminal (or group of terminals) that connect through an IBM 3705 Communication Controller with ACF/NCP and PRPQ P09021 (NEF1). This way of connecting ALC terminals is now obsolete, and this book no longer describes NEF1 parameters. Contact your IBM representative if you plan to use NEF1 with ALCS.

```
[label}  COMDEF  LDTYPE=VTAMALC
                          ,NAME=name
                          ,APPL=application_name
                          ,TERM={1977|1980|2915|4505}
                          ,LEID=leid
                          [,ASSDEV=name]
                          [,BUFSZE=size]
                          [,CRAS={ATnnn|APnnn}]
                          [,CRI=explicit_cri]
                          [,CSID=terminal_id]
                          [,ISTATUS={ACTIVE|INACTIVE}]
                          [,LOGON=({YES|(NO[,USERID=userid]})]
                          [,NEFLU=name]
                          [,ORD=explicit_ordinal]
                          [,RETRV={NO|YES}]
                          [,SCRSZE=rows]
                          [,SYSSEND={YES|NO}]
                          [,TAB={NO|YES}]
                          [,TEST={NO|YES}
                          [,TIMEOUT=({50|time1],[300|time2],[3|count1])]
                          [,UNSLITE={NO|YES}]
                          [,UPDATE={ADD|DELETE|REPLACE}]
                          [,USERDAT=user_data]
                          [,USER1=user_data]
                          ⋮
                          [,USER16=user_data]
```

Where:

NAME=*name*
Where *name* is the CRN of the terminal controlled by a NEF or ALCI LU. An alphanumeric string of 1 to 8 characters; the first character must be alphabetic. *name* must be unique within ALCS.

TERM={1977|1980|2915|4505}
Terminal device type, where:

Table 22. TERM= parameter values and system defaults for LDTYPE=VTAMALC

TERM=	Function	SCRSZE	BUFSZE
1977	ALC keyboard printer		97 bytes
1980	ALC receive-only printer		97 bytes
2915	ALC display	12 lines	
4505	ALC display	30 lines	

Use SCRSZE to override the display size and BUFSZE to override the buffer size.

TIMEOUT=({50|*time1*],[300|*time2*],[3|*countl*])
Use only for printers where TERM=1977 or TERM=1980. For further information, see "COMGEN macro" on page 98.

For other parameters, see Table 13 on page 110; and see also "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

COMDEF parameters for a VTAM 3270 terminal

Use this COMDEF format to define an IBM 3270 terminal (or compatible device) that connects to ALCS through VTAM.

```
[label}  COMDEF  LDTYPE=VTAM3270
                          ,NAME=name
                          ,APPL=application_name
                          [,ASSDEV=name]
                          [,CRAS={PRC|ROC|(ATnnn|APnnn[,|NOFALLBACK|FALLBACK])}]
                          [,CRI=explicit_cri]
                          [,CSID=terminal_id]
                          [,DBCS=NO|YES]
                          [,ISTATUS={ACTIVE|INACTIVE|SHARED}]
                          [,LOGON=({YES|(NO[,USERID=userid]})]
                          [,ORD=explicit_ordinal]
                          [,PRI={0|nnn}]
                          [,RETRV={NO|YES}]
                          [,SYSSEND={YES|NO}]
                          [,TERM={DISPLAY|3270DSP|PRINTER|3270PRT}]
                          [,TEST={NO|YES}]
                          [,UPDATE={ADD|DELETE|REPLACE}]
                          [,USERDAT=user_data]
                          [,USER1=user_data]
                          ⋮
                          [,USER16=user_data]
```

Where:

NAME=*name*
CRN of the terminal. An alphanumeric string of 1 to 8 characters; the first character must be alphabetic. *name* must be the LU name of the terminal as defined to VTAM, and must be unique within ALCS.

CRAS={PRC|ROC|(AT*nnn*|AP*nnn*[,|FALLBACK|NOFALLBACK])}
The terminal has CRAS status. Omit this parameter if the terminal does not have CRAS status. Otherwise specify one of:

PRC
Prime CRAS.

ROC
RO CRAS.

AT*nnn*
Alternate CRAS, *nnn* is decimal 1 through 255.

AP*nnn*
Alternate CRAS printer, *nnn* is decimal 1 through 255.

For alternate CRASs and alternate CRAS printers, an optional second subparameter specifies whether or not the terminal is a CRAS fallback candidate, as follows:

FALLBACK
For AT1 through AT16 and AP1 through AP16 only. The terminal is a fallback candidate. Can be abbreviated to F. This is the default for alternate CRASs AT1 through AT16.

NOFALLBACK
For AT1 through AT16 and AP1 through AP16 only. The terminal is not a fallback candidate. Can be abbreviated to NF. This is the default for alternate CRAS printers AP1 through AP16.

TERM={DISPLAY|3270DSP|PRINTER|3270PRT}
Type of terminal (display or printer) One of:

DISPLAY
3270 display

3270DSP
3270 display

PRINTER
3270 printer, LU type 3

3270PRT
3270 printer, LU type 3

The TERM= parameter is required when TEST=YES

TEST={NO|YES}
When TEST=YES, TERM specifies the type of terminal for use by the system test vehicle (STV). When TEST=N0, TERM specifies the type of terminal that is assumed until a VTAM session is established with the terminal.

For other parameters, see Table 13 on page 110; and see "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

COMDEF parameters for a connection to WAS

Use this COMDEF format to define a connection to WAS.

```
[label}  COMDEF  LDTYPE=WAS
                           ,NAME=crn
                           ,APPL=application_name
                           ,WASNAME=(daemongroupname,nodename,servername)
                           ,REGNAME=register name
                           ,SERVNAME=(requestservicename_1,requestservicename_2)
                           [,CONN=([rcvconn|10},{sendconn|0})]
                           [,CRI=explicit_cri[],ISTATUS{ACTIVE|INACTIVE}]
                           [,ISTATUS={{ACTIVE|INACTIVE}]
                           [,LOGON=(NO[,USERID=userid])]
                           [,ORD=explicit_ordinal]
                           [,PRTCOL={TYPE1|TYPE2}]
                           [,UPDATE={ADD|DELETE|REPLACE}]
                           [,USERDAT=user_data]
                           [,USER1=user_data]
                           ⋮
                           [,USER16=user_data]
```

Where:

NAME=*crn*
CRN of this resource, 1 to 8 alphanumeric characters. It must be unique within this ALCS system.

APPL=*application_name*
Name of the application where ALCS routes messages that are received on this connection when the incoming message is not destined for a terminal (see installation-wide monitor exit USRWAS3 for more details).

WASNAME=(*daemongroupname*,*nodename*,*servername*)

daemongroupname
The name of the WAS z/OS Daemon group to be joined, 1 to 8 upper-case alphanumeric characters.

nodename
The short name of the WAS z/OS node to be joined, 1 to 8 upper-case alphanumeric characters.

servername
The short name of the WAS z/OS server to be joined, 1 to 8 upper-case alphanumeric characters.

REGNAME=*registername*
The name of the set of OLA local connections for this WAS resource, 1 to 12 upper-case alphanumeric characters. This name must be unique within the GRS complex.

CONN=*({rcvconn|10},{sendconn|0})*
The number of concurrent receive/response rcvconn and send connections sendconn (OLA connections) allowed for this WAS resource. For the receive/response connections the minimum value is 1, the maximum value is 500, and the default value is 10. For the send connections the minimum value is 0, the maximum value is 99, and the default value is 0. The sum of rcvconn

and sendconn must not exceed 599. Observe that usually most traffic is "incoming" traffic, which is handled, including responses, by the receive/response connections.

SERVNAME=(*requestservicename_1*,*requestservicename_2*)

requestservicenname_1
The name of the target service specified on the InteractionSpec by the WAS application, 1 to 40 alphanumeric characters. This must be unique within the ALCS system.

requestservicenname_2
The name of the WAS service (JNDI home name for the target EJB), 1 to 40 alphanumeric characters. This must be unique in the WAS server you are targeting.

The service names can contain mixed-case alphanumeric characters and any other characters that are allowed for WAS object names (but not imbedded spaces). You can combine upper-case and lower-case characters freely in the service names. For example:

```
SERVNAME=(ALCSZ001,
ejb/com/ibm/alcs/alcsoutbound)
```

PRTCOL={TYPE1|TYPE2}
Protocol type of the WAS application, where:

TYPE1
The WAS application does not use the standard correlator format. This is the default. This means installation-wide monitor exits USRWAS3 and USRWAS5 must be used. Also the ECB-controlled exits AWA1 and AWA2 are not invoked during either activation or inactivation of the WAS Bridge.

TYPE2
The WAS application uses the standard correlator format. This means the installation-wide monitor exits USRWAS3 and USRWAS5 are not needed. Also the ECB-controlled exits AWA1 and AWA2 can be invoked during either activation or inactivation of the WAS Bridge.

The format of the standard correlator is:

Byte 1
Correlator version. The only allowed value is 01 (hexadecimal).

Byte 2-9
CRN of the subordinate device.

Byte 10
RECODV1 value of the subordinate device. See “Communication resource data DSECT - CO0RE” on page 312 for details. This is only used on ALCS-to-WAS traffic.

Byte 11
INDICATOR (Only used on ALCS to WAS traffic)

01 (hexadecimal) - "ALC" terminal
02 (hexadecimal) - "ALC" printer
03 (hexadecimal) - "3270" terminal
04 (hexadecimal) - "3270" printer

Byte 12
Reserved - Set to hexadecimal zeroes.

For other parameters, see Table 13 on page 110; and see also “Setting defaults for COMDEF parameters - COMDFLT macro” on page 105.

COMDEF parameters for a terminal connected through WAS

Use this COMDEF format to define a terminal connected through WAS.

```
[label}  COMDEF  LDTYPE=WASTERM
                           ,NAME=crn
                           ,APPL=application_name
                           ,LINK=WAS_resource_name
                           ,TERM={1977|1980|2915|4505|3270DSP|3270PRT}
                           [,ASSDEV=name]
                           [,BUFSIZE=size]
                           [,CRAS={ATnnn|APnnn}]
                           [,CRI=explicit_cri]
                           [,DBCS=NO|YES]
                           [,ISTATUS={ACTIVE|INACTIVE}]
                           [,LOGON=({YES|(NO[,USERID=userid]})]
                           [,ORD=explicit_ordinal]
                           [,RETRV=NO|YES]
                           [,SCRCOL=cols]
                           [,SCROLL={YES|NO}]
                           [,SCRSZE=rows]
                           [,SYSSEND=YES|NO]
                           [,TAB=NO|YES]
                           [,TIMEOUT=({50|time1],[300|time2],[3|count]
                           [,UNSLITE=NO|YES]
                           [,UPDATE={ADD|DELETE|REPLACE}]
                           [,USERDAT=user_data]
                           [,USER1=user_data]
                           ⋮
                           [,USER16=user_data]
```

Where:

NAME=*crn*
CRN of this resource, 1 to 8 alphanumeric characters. It must be unique within this ALCS system.

LINK=*WAS_resource_name*
The CRN of a WAS resource, through which ALCS accesses the terminal. Use COMDEF LDTYPE=WAS to define the WAS resource.

TERM={1977|1980|2915|4505|3270DSP|3270PRT}
Terminal device type, where:

Table 23. TERM= parameter values and system defaults for LDTYPE=WASTERM

TERM	Function	SCRSZE	SCRCOL	BUFSZE
1977	ALC keyboard printer			97 bytes
1980	ALC receive-only printer			97 bytes
2915	ALC display	12 lines	64 cols	
4505	ALC display	30 lines	64 cols	
3270DSP	3270-type display	24 lines	80 cols	
3270PRT	3270-type printer			1920 bytes

Use SCRSZE and SCRCOL to override the display size and BUFSZE to override the buffer size.

You can only override the number of columns with TERM=3270DSP.

Note: ALCS does not convert messages to or from 3270 datastream format for TERM=3270DSP or TERM=3270PRT.

SCROLL={YES|NO}]

Yes
Allow scrolling to different parts of those responses that are too large to fit on a single screen. This is the default, when the ALCS scrolling package is used.

No
Inhibit scrolling and send the complete unformatted response to this communication resource. The application that constructs the response may only use the DISPC ADD and SEND macros. The DISPC SEND DEST parameter is not allowed and the DISPC SEND START, NUM, SPACE, BOTTOM, LINE, CONFIRM,CLEAR parameters are ignored. Heap storage is used for the response. You must allocate type 3 storage units to allow for the additional heap storage needed. You should specify a type 3 storage unit size greater than or equal the largest response plus 50 bytes (to allow for message headers and trailers).

Printer devices (TERM=1977, TERM=1980, TERM=3270PRT) must not send printer acknowledgments to ALCS, because ALCS simulates the printer acknowledgments itself.

ALC display devices (TERM=2915 and TERM=4505) can use the field identifier (FID) sequence. The default format is a 1-byte command character followed by the FID character (hexadecimal 7A).

ALC display devices (TERM=2915 and TERM=4505) can send action key messages to ALCS. The default format is a 1-byte action key character followed by an EOP character.

3270-type display devices (TERM=3270DSP) can send function key messages to ALCS. The default format is a 2-byte function key value plus optional text characters followed by an EOP character.

DBCS={NO|YES}
Whether the resource supports the double byte character set or not. DBCS=YES is only allowed with TERM=3270DSP or TERM=3270PRT. Default is DBCS=NO.

SCRCOL=*cols*
Number of columns on the display terminal. You can only override the number of columns with TERM=3270DSP.

Table 24. SCRCOL= system defaults for LDTYPE=WASTERM

TERM	Screen size (columns)
2915	64
4505	64
3270DSP	80

SCRSZE=*rows*
Number of lines on the display terminal.

Table 25. SCRSZE= system defaults for LDTYPE=WASTERM

TERM	Screen size (lines)
2915	12
4505	30
3270DSP	24

For other parameters, see Table 13 on page 110; and see “Setting defaults for COMDEF parameters - COMDFLT macro” on page 105.

COMDEF parameters for a WTTY link

Use this COMDEF format to define a World Trade Teletypewriter link that connects to ALCS through NTO and VTAM.

```
[label}  COMDEF  LDTYPE=WTTY
                          ,TERM={SI|SO|HDX|FDX}
                          ,NAME={name|(name1,name2)}
                          ,APPL=application_name
                          [,CRI=explicit_cri]
                          [,ISTATUS={ACTIVE|INACTIVE}]
                          [,LOGON=(NO[,USERID=userid])]
                          [,ORD=explicit_ordinal]
                          [,TEST={NO|YES}]
                          [,UPDATE={ADD|DELETE|REPLACE}]
                          [,USERDAT=user_data]
                          [,USER1=user_data]
                          ⋮
                          [,USER16=user_data]
```

Where:

TERM={SI|SO|HDX|FDX}
Type of WTTY link, as follows:

SI
Simplex in. ALCS can receive data from the link but cannot send data to it.

SO
Simplex out. ALCS can send data to the link but cannot receive data from it.

HDX
Half-duplex. ALCS can send data to, and receive data from, the link but cannot do both simultaneously.

FDX
Full-duplex. ALCS can send data to, and receive data from, the link simultaneously.

NAME={*name*|(*name1*,*name2*)}
CRN of the link. Each name is an alphanumeric string of 1 to 8 characters; the first character must be alphabetic. The names must be the virtual SNA representation LU names as defined to NTO, and must be unique within ALCS.

When TERM is SI, SO, or HDX, specify *name*.

When TERM is FDX specify both *name1* and *name2*, as the virtual SNA representation LU names of the send and receive sides respectively.

See "Defining virtual SNA representations for WTTY devices" on page 36.

APPL=*application_name*
Name of the application to which ALCS routes input messages from this link. Omit for simplex out (TERM=SO). Use COMDEF LDTYPE=ALCSAPPL to define the application (see "COMDEF parameters for a local application" on page 123). The application must be owned by ALCS.

For other parameters, see Table 13 on page 110; and see "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

COMDEF parameters for AX.25 ALC terminals

Use this COMDEF format to define a group of ALC terminals (or compatible devices) attached to a high-level network that connects to ALCS through an X.25 PVC. All the terminals on one of these COMDEF macroinstructions must be the same device type.

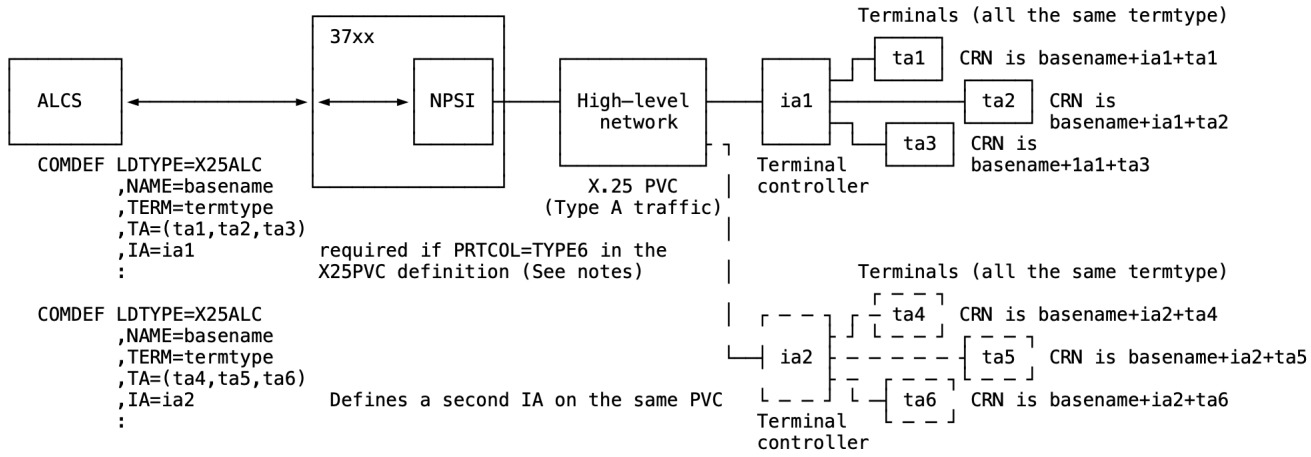


Figure 41. The COMDEF parameters to define an ALC terminal on an AX.25 PVC

Notes:

1. For a type 1 X.25 PVC:

   Notes:

   a. Use a single COMDEF LDTYPE=X25ALC to define terminals which are all the same type.

   b. Use separate COMDEF macros (with the same PVC name) to define terminals which are different types.

2. For a type 6 X.25 PVC:

   Notes:

   a. You must define at least one COMDEF macro for each interchange (IA)

   b. Use separate COMDEF macros to define terminals which are different types attached to the same interchange.

3. For a type 7 X.25 PVC:

   Notes:

   a. You must define at least one COMDEF macro for each terminal circuit identity (TCID) and interchange address (IA).

   b. Use separate COMDEF macros to define terminals which are different types attached to the same terminal circuit and interchange.

```
[label}  COMDEF  LDTYPE=X25ALC
                          ,NAME=base_name
                          ,APPL=application_name
                          ,PVC=name
                          ,TA=(ta,...)
                          ,TERM={1977|1980|2915|4505}
                          [,ASSDEV=asname]
                          [,BUFSZE=size]
                          [,CRAS={ATnnn|APnnn}]
                          [,CRI=explicit_cri]
                          [,CSID=(terminal_id,...)]
                          [,IA=ia]
                          [,ISTATUS={ACTIVE|INACTIVE}]
                          [,LOGON=({YES|(NO[,USERID=userid]})]
                          [,ORD=explicit_ordinal]
                          [,RETRV={NO|YES}]
                          [,SCRSZE={30|rows}]
                          [,SYSSEND={YES|NO}]
                          [,TAB={NO|YES}]
                          [,TCID=tcid]
                          [,TEST={NO|YES}]
                          [,TIMEOUT=({50|time1],[300|time2],[3|countl])]
                          [,UNSLITE={NO|YES}]
                          [,UPDATE={ADD|DELETE|REPLACE}]
                          [,USERDAT=user_data]
                          [,USER1=user_data]
                          ⋮
                          [,USER16=user_data]
```

Where:

NAME=*base_name*
Base CRN for the terminal. An alphanumeric string of either:

- 1 to 4 characters
- 1 to 6 characters

The first character must be alphabetic. The stage 1 generation builds the full CRN for the terminal.

For a terminal connected through an X.25 PVC that is PRTCOL=TYPE1, the terminal CRN is made up of:

- Base name (NAME); 1 to 6 characters
- Two-character terminal address (TA)

For a terminal connected through an X.25 PVC that is PRTCOL=TYPE6 or PRTCOL=TYPE7, the terminal CRN is made up of:

- Base name (NAME); 1 to 4 characters
- Two-character interchange address (IA)
- Two-character terminal address (TA)

The full CRN for each terminal must be unique within ALCS.

PVC=*name*
CRN of the X.25 PVC through which ALCS accesses the terminal. This X.25 PVC must be PRTCOL=TYPE1 or PRTCOL=TYPE6 or PRTCOL=TYPE7.

TA=(*ta*,...)
Terminal addresses (TAs) of the terminals; each is a 2-digit hexadecimal number. All the terminals must be the same device type. Other parameters on this macroinstruction apply to all of these terminals - except CRAS, which applies to the first or only terminal address.

TERM={1977|1980|2915|4505}
Terminal device type, where:

Table 26. TERM= parameter values and system defaults for LDTYPE=X25ALC			
TERM=	**Function**	**SCRSZE**	**BUFSZE**
1977	ALC keyboard printer		97 bytes
1980	ALC receive-only printer		97 bytes
2915	ALC display	12 lines	
4505	ALC display	30 lines	

Use SCRSZE to override the display size and BUFSZE to override the buffer size.

IA=*ia*
Interchange address for the terminal controller; a 2-digit hexadecimal number. Use this parameter to define the interchange address of a terminal controller on an X.25 PVC (LDTYPE=X25PVC). You must specify PRTCOL=TYPE6 or PRTCOL=TYPE7 in the COMDEF for the X.25 PVC if you want to use the IA parameter.

TAB={NO|YES}
The group of printers has (YES) or does not have (NO) the tabular skip feature. It applies to all terminals (TAs) on this COMDEF. Omit this parameter when the group does not consist of printers.

TCID=*tcid*
HLN owner's terminal circuit identity of the terminal - a 2-digit hexadecimal number. Use this parameter to define the terminal circuit identity of a terminal controller on an X.25 PVC (LDTYPE=X25PVC). You must specify PRTCOL=TYPE7 in the COMDEF for the X.25 PVC if you want to use the TCID parameter.

TIMEOUT=({50|*time1*],[300|*time2*],[3|*countl*])
Use only for printers where TERM=1977 or TERM=1980. For further information, see "COMDEF macro" on page 107.

For other parameters, see Table 13 on page 110; and see "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

COMDEF parameters for an X.25 permanent virtual circuit (PVC)

Use this COMDEF format to define an X.25 PVC.

For Type A traffic:

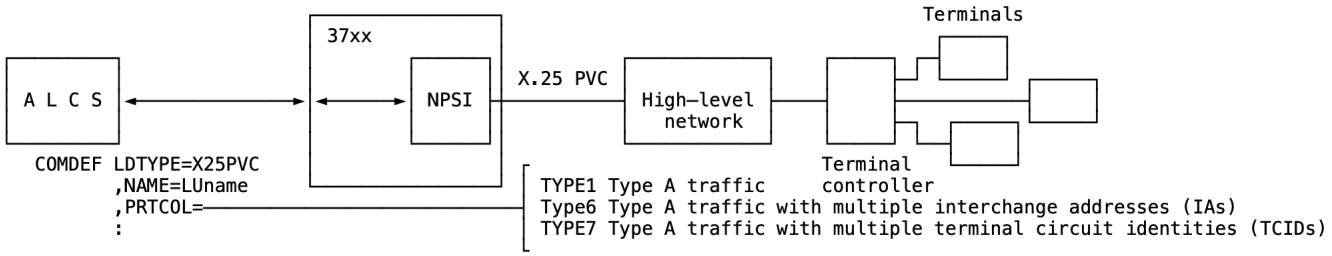


Figure 42. The COMDEF parameters to define an X.25 PVC with type A traffic

```
[label}  COMDEF  LDTYPE=X25PVC
                          ,NAME=lu_name
                          [,PRTCOL={TYPE1|TYPE6|TYPE7}]
                          [,CODE={ALC|IATA7|IATA5|NONE}]
                          [,CRI=explicit_cri]
                          [,ISTATUS={ACTIVE|INACTIVE}]
                          [,LOGON=(NO[,USERID=userid])]
                          [,ORD=explicit_ordinal]
                          [,PRI={0|nnn}]
                          [,TEST={NO|YES}]
                          [,UPDATE={ADD|DELETE|REPLACE}]
                          [,USERDAT=user_data]
                          [,USER1=default]
                          ⋮
                          [,USER16=default]
```

For other traffic:

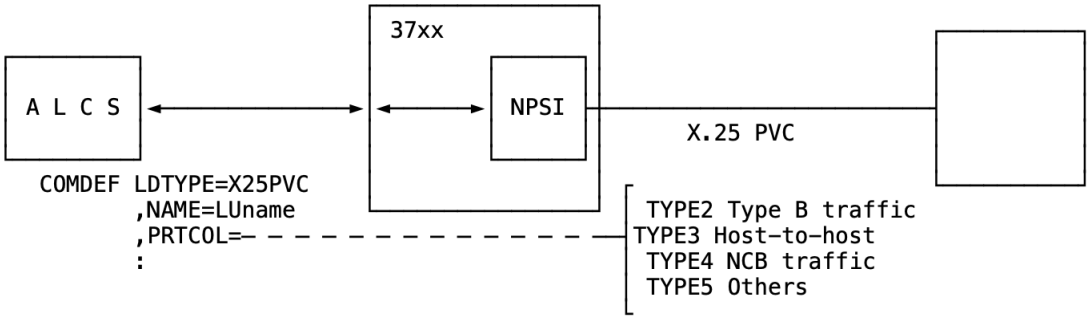


Figure 43. The COMDEF parameters to define an X.25 PVC with other traffic

```
[label}  COMDEF  LDTYPE=X25PVC
                          ,NAME=lu_name
                          ,PRTCOL={TYPE2|TYPE3|TYPE4|TYPE5}
                          ,APPL=application_name
                          [,CODE={ALC|IATA7|IATA5|NONE}]
                          [,CRI=explicit_cri]
                          [,ISTATUS={ACTIVE|INACTIVE}]
                          [,LOGON=(NO[,USERID=userid])]
                          [,ORD=explicit_ordinal]
                          [,PRI={0|nnn}]
                          [,TEST={NO|YES}]
                          [,UPDATE={ADD|DELETE|REPLACE}]
                          [,USERDAT=user_data]
                          [,USER1=user_data]
                          ⋮
                          [,USER16=user_data]
```

Where:

NAME=*lu_name*
CRN of the X.25 PVC. An alphanumeric string of 1 to 8 characters; the first character must be alphabetic. It is the LU name of the X.25 resource representing a logical channel, and must be unique within ALCS.

PRTCOL={TYPE1|TYPE2|TYPE3|TYPE4|TYPE5|TYPE6|TYPE7}
Type of X.25 PVC:

PRTCOL=TYPE1
The X.25 PVC is used for conversational (Type A) traffic. Use PRTCOL=TYPE1 if you do not want to define multiple interchange addresses. ALCS routes conversational messages to the application

specified for the input terminal. To define a terminal on this link, see "COMDEF parameters for AX.25 ALC terminals" on page 159.

PRTCOL=TYPE2
The X.25 PVC is used for conventional (Type B) traffic.

The following parameter is mandatory for this type:

APPL=*application_name*
Name of the application to which ALCS routes conventional messages. Use COMDEF LDTYPE=ALCSAPPL to define the application (see "COMDEF parameters for a local application" on page 123).

PRTCOL=TYPE3
The X.25 PVC is used for host-to-host traffic.

The following parameter is mandatory for this type:

APPL=*application_name*
Name of the application to which ALCS routes messages. Use COMDEF LDTYPE=ALCSAPPL to define the application (see "COMDEF parameters for a local application" on page 123). Do not confuse this parameter with the logical application designator/address as defined in ATA/IATA host-to-host protocol, which is a second-level designator.

PRTCOL=TYPE4
The X.25 PVC is used for network control block (NCB) traffic. Code COMDEF LDTYPE=X25PVC for each SITA regional control center (RCS) controlling ALC terminals that are connected to ALCS through an X.25 PVC.

The following parameter is mandatory for this type:

APPL=*application_name*
Name of the application to which ALCS routes NCBs.

The ALCS ECB-controlled monitor program CNCB must be defined as the input message editor for this application, if you want ALCS to forward SITA NCBs to NetView. If it is, ALCS conditionally calls the installation-wide exit program ANCB (see "Incoming SITA NCB processing exit program - ANCB" on page 376) and sends a copy of the NCB to the alternate CRAS printer AP1 (if that printer has been defined).

PRTCOL=TYPE5
The X.25 PVC is used for any X.25 communication other than those described above.

The following parameter is mandatory for this type:

APPL=*application_name*
Name of the application to which ALCS routes data. ALCS does not inspect or modify the input data. Use COMDEF LDTYPE=ALCSAPPL to define the application.

PRTCOL=TYPE6
The X.25 PVC is used for conversational (Type A) traffic. Use PRTCOL=TYPE6 to define a link which has multiple interchange addresses. ALCS routes conversational messages to the application specified for the input terminal. When you define a terminal on this link, you must use the IA=*ia* parameter (in the COMDEF macro for the terminal) to specify the interchange address. "COMDEF parameters for AX.25 ALC terminals" on page 159 shows how to use the IA parameter.

PRTCOL=TYPE7
The X.25 PVC is used for conversational (Type A) traffic. Use PRTCOL=TYPE7 to define a link that has multiple terminal circuit identities. ALCS routes conversational messages to the application specified for the input terminal. To define a terminal on this link, use the TCID=*tcid* and IA=*ia* parameters (in the COMDEF macro for the terminal) to specify the terminal circuit identity and interchange address. "COMDEF parameters for AX.25 ALC terminals" on page 159 shows how to use the IA and TCID parameters.

For other parameters, see Table 13 on page 110; and see "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

Generating a sequential file configuration table

ALCS uses the MVS sequential access method (MVS/SAM) to create and access physical sequential data sets, which are called **sequential files**. Figure 44 on page 165 gives an overview of how each type of file is used.

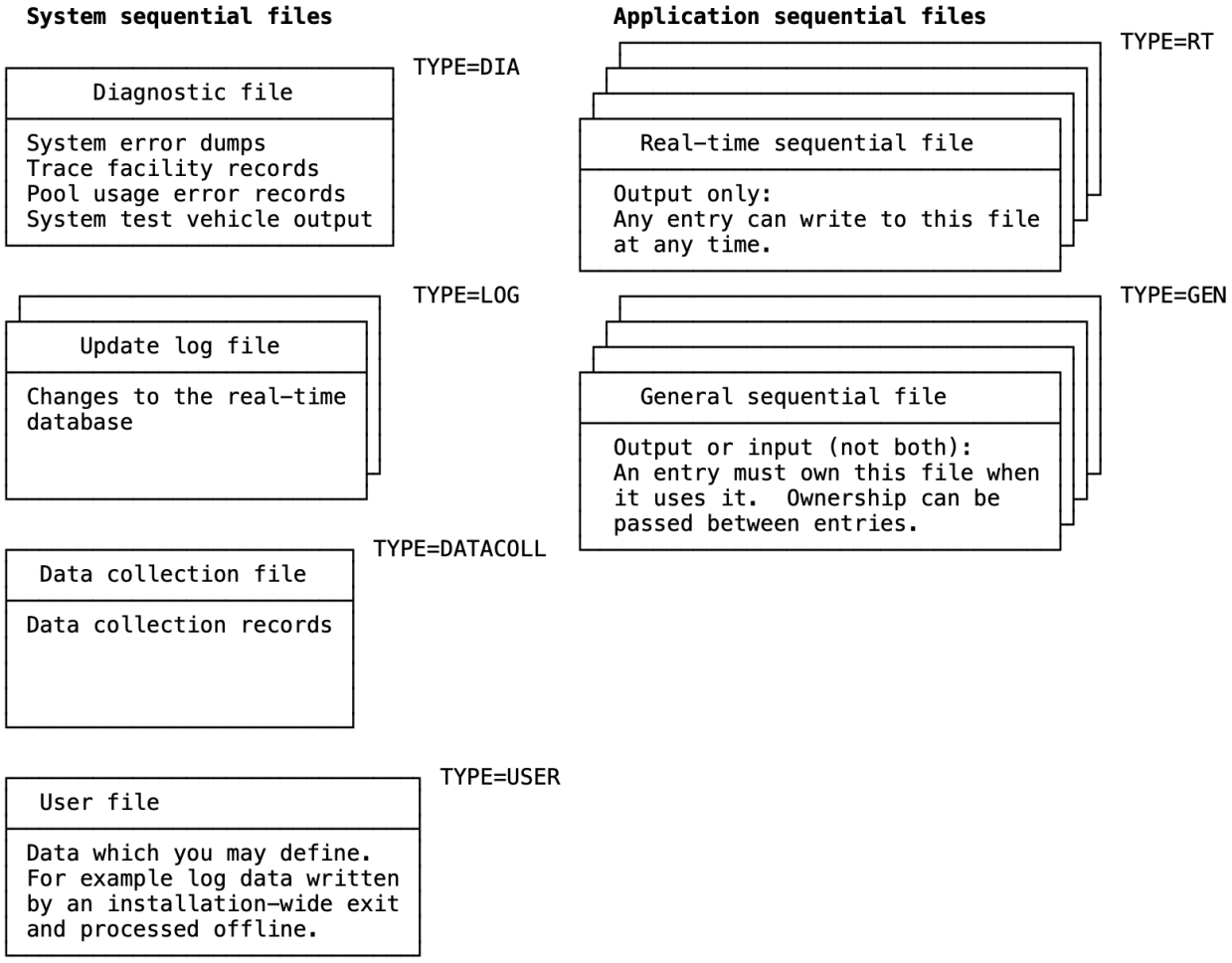


Figure 44. ALCS sequential files: Types and contents

Use the SEQGEN macro to generate the ALCS sequential file configuration table. "Specifying sequential files" on page 165 explains how to load the sequential file configuration table. "SEQGEN macro" on page 166 explains how to code the SEQGEN macro.

Specifying sequential files

To define sequential files to ALCS:

1. Run an ALCS generation to build a new sequential file configuration table. See "SEQGEN macro" on page 166 for more details.
2. Use the ZASEQ command to load the new table on to the online system. (See *ALCS Operation and Maintenance*.) The new table replaces the previous table in storage only, so at restart there is an automatic fallback to the previous table. The table change is transparent to programs that are using sequential files.

   ZASEQ only updates the existing sequential file configuration for general sequential files. Other sequential file definitions are ignored.
3. Include the name of the new table in the PARM field on the EXEC statement that starts ALCS.

SEQGEN macro

Use SEQGEN to define a sequential file. Code SEQGEN once for each sequential file that the ALCS system can use.

```
[label] SEQGEN [DUMMY,]
                          NAME=seq
                          ,TYPE={(RT[,TIME])|GEN|(LOG[,type])|DIA|DATACOLL|USER}
                          [,BLKSIZE=blksize]
                          [,BUFNO={5|bufno}]
                          [,DATACLAS=data_class_name]
                          [,DEST=node|(node,userid)]
                          [,DISP=({status]
                                     [,normal_disposition]
                                     [,conditional_disposition])]
                          [,DSNAME=dsname]
                          [,EOT={NO|YES}]
                          [,LABEL=(,[type],,[OUT|IN][,RETPD=days])]
                          [,LRECL=lrecl]
                          [,MEMBER=member]
                          [,MGMTCLAS=management_class_name]
                          [,PROTECT={NO|YES}]
                          [,RECFM={F|FB|V|VB|U}[|A|M]]
                          [,SPACE=(primary_blocks[,secondary_blocks])]
                          [,STORCLAS=storage_class_name]
                          [,SYSOUT=class]
                          [,TRTCH={COMP|NOCOMP}]
                          [,UNIT=(type[,count])]
                          [,VOLUME=(,,,[volcnt][,SER=volser])]
```

or:

```
[label] SEQGEN NAME=seq
                          ,TYPE={(RT[,TIME])|GEN}
                          ,USE=alt
```

Where:

label
Any valid assembler label.

DUMMY
Dummy data set. For input files, any attempt to read results in an immediate end-of-file. For output files, MVS discards any output. For dummy data sets, specify RECFM, BLKSIZE, and if applicable LRECL.

NAME=*seq*
The 3-character alphabetic symbolic name for the sequential file. This is the name that application programs specify in sequential file macros such as TOPNC, TWRTC, TOURC, and so on. Every sequential file must have a unique symbolic name.

TYPE={(RT[,TIME])|GEN|(LOG[,*type*])|DIA|DATACOLL|USER}
Sequential file function, see *ALCS Concepts and Facilities* for an overview of sequential files.

RT|REALTIME[,TIME|TIMESTAMP]
Application real-time sequential file.

TIME|TIMESTAMP
Append a 16-byte time-stamp field to each output record on the sequential file.

Programs designed to read TPF real-time tapes expect a 16-byte time-stamp field at the end of each logical record. To use these programs with ALCS, you must specify TYPE=(RT,TIME) when you define the sequential file in the ALCS sequential file configuration table.

TYPE=(RT,TIME) causes ALCS to add a time-stamp to each record that an ECB-controlled program writes with TOURC or TOUTC. The format of the time-stamp is:

Byte
Contents

0-7
Reserved, set to binary zeros

8-15
TOD clock value at the time the record was written.

For fixed-length (RECFM=F or RECFM=FB) records, ALCS puts the time-stamp in the last 16-bytes of the record.

For variable (RECFM=V or RECFM=VB) or undefined (RECFM=U) records, ALCS adds the time-stamp at the end of the data (that the application provides) and increases the logical record length by 16. If the new logical record length is bigger than the physical block into which the data is being written, the data is truncated to the maximum size and the time-stamp occupies the last 16 bytes of data.

GEN|GENERAL
Application general sequential file.

(LOG|LOGGING[,*type*])
ALCS database update log file, where *type* is one of:

Forward
Forward logging; this is the default. When a database record is updated, ALCS writes the new (after update) record contents to the sequential file.

Backward
Backward logging. When a database record is updated, ALCS writes the old (before update) record contents to the sequential file. ALCS can do this only for records with update mode VFA option, see "USRDTA macro" on page 190.

Merge
Combined forward and backward logging. When a database record is updated, ALCS writes both the old contents (when the record has update mode VFA option) and the new contents of the record to the sequential file.

You must include at least one database update log in every sequential file generation. You can include one of:

- A forward logging file
- A backward logging file
- A forward logging file and a backward logging file
- A merged logging file.

DIA|DIAGNOSTIC
ALCS diagnostic file.

Include one and only one ALCS diagnostic file in every sequential file generation.

DATACOLL
ALCS data collection file. If you do not define a separate data collection file, ALCS writes data collection records to the ALCS diagnostic file.

USER
ALCS user file. If you do not define a separate user file, ALCS writes user records to the ALCS diagnostic file.

BLKSIZE=*blksize*
Actual block size in bytes if fixed-length records, maximum block size in bytes if variable-length or undefined-length records. Maximum is 32 760. Minimum value is:

16
For general sequential files

16
For application real-time sequential files without time-stamps.

32
For application real-time sequential files with time-stamps

4104
For ALCS database update log

6152
For ALCS diagnostic file

For the ALCS database update log, ensure that the block size is large enough to contain the largest database record, plus 4 bytes for the block descriptor word (BDW), plus 4 bytes for the record descriptor word (RDW).

If you have defined a 32k record size (ALCS CISIZE=32768) then set the block size for the ALCS database update log to 32 760. For these records, ALCS automatically shifts the pool stamps on writing to, and reading from, the sequential file so that no data is lost.

BLKSIZE is mandatory for undefined-length records (RECFM is U, UA, or UM). BLKSIZE must be omitted if RECFM is not specified, otherwise BLKSIZE is optional. If BLKSIZE is omitted, MVS determines the optimum block size when the data set is created, based on the device type and the LRECL value specified.

BUFNO={5|*bufno*}
Number of buffers that ALCS allocates for the file. Minimum value is 1, maximum is 255, default is 5.

DATACLAS=*data_class_name*
SMS data class (up to 8 characters) used for the data set. It is allocated by the MVS Systems Programmer. This parameter can be used only when the Storage Management Subsystem is active. It only applies to general sequential files.

DEST=*node*|(*node*,*userid*)
Destination for a SYSOUT data set.

node
Identifies a node. The node is a symbolic name defined by the installation during JES initialization. It is 1 through 8 alphanumeric or national ($, #, @) characters.

userid
Identifies a TSO/E or VM userid at the node. The userid must be defined at the node. For TSO/E it is 1 through 7 alphanumeric or national ($, #, @) characters. For VM it is 1 through 8 alphanumeric or national ($, #, @) characters.

DEST is ignored if SYSOUT is not specified.

The data will be sent to the destination when the file is deallocated from ALCS. In the case of a general sequential file, this is after the file is closed. In the case of a real-time sequential time, this is after the file is switched.

The appropriate *MVS JCL Reference* manual describes the destination for a sysout data set, but note that the format of this DEST parameter is different from the format of the MVS JCL DD statement DEST= parameter.

DISP=({*status*],[*normal_disposition*],[*conditional_disposition*])
Status and disposition of the data set:

status
Data set status, one of: NEW, OLD, MOD, or SHR. Default is OLD for input files, NEW for output files. Do not specify NEW or MOD for input files. Omit or specify NEW for application real-time sequential files, the ALCS database update log, and the ALCS diagnostic file.

normal_disposition
Normal termination disposition, one of: DELETE, KEEP, CATLG, or UNCATLG.

conditional_disposition
Conditional termination disposition, one of: DELETE, KEEP, CATLG, or UNCATLG.

See the appropriate MVS *MVS JCL Reference* manual for descriptions of these parameters, including any defaults. Note that ALCS uses OLD as the default data set status for input files.

DSNAME=*dsname*
Data set name (see the appropriate MVS *MVS JCL Reference* manual).

For **general sequential files**, *dsname* can be:

- An unqualified name or a qualified name. It must not contain special characters other than periods (to indicate a qualified data set name) and hyphens. The maximum length is 44 characters.
- A generation data group member. To do this, specify *dsname* as *gdgname(generation)* where *gdgname* is the data set name of the generation data group and *generation* is the relative generation member. The maximum length for *gdgname* is 34 characters. Appendix I, "Sample definition for GDG sequential file," on page 577 gives a sample definition for a generation data group.
- A partitioned data set member. To do this, specify *dsname* as *pdsname(pdsmem)*, where *pdsname* is the data set name of the library and *pdsmem* is the member name. The maximum length for *pdsname* is 34 characters.

For the following files:

- **Application real-time sequential files**
- **ALCS database-update log**
- **ALCS diagnostic file**
- **ALCS data-collection file**
- **ALCS user file**

dsname is one or more data set name qualifiers joined by periods (see the appropriate MVS *MVS JCL Reference* manual). During execution, ALCS constructs the actual data set name; to do this, it adds an 8-character qualifier to the end of *dsname*. The maximum length of *dsname* is 35 characters.

For the **MVS log stream** use 1 to 26 characters for the name of the sequential file.

For any sequential file, *dsname* can contain one or more of the following 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 system (MVS) dates and times, not ALCS dates and times.

Note that MVS does not allow data set name qualifiers to begin with digits. Do not specify (for example):

```
DSNAME=QUAL1.QUAL2.-GDAT.-GTIME.QUAL3
```

Instead, specify (for example):

```
DSNAME=QUAL1.QUAL2.D-GDAT.T-GTIME.QUAL3
```

This parameter is ignored if **USE**= is specified. It is optional for SYSOUT and dummy files. Otherwise it is mandatory.

Note that MVS uses only the rightmost 17 characters of *dsname* in magnetic tape file labels.

You can process a specific data set within a generation data group by providing the fully-qualified data set name, including the absolute generation number and version number as the last qualifier (in the form G*nnn*V*nn*). See the appropriate *MVS Data Administration Guide*.

EOT={NO|YES}
This parameter is only valid for real-time and system sequential files.

NO
Switch to a standby real-time or system sequential file when the SPACE threshold is reached (this is the default behaviour).

YES
Switch to a standby real-time or system sequential file when an End Of Tape marker is detected overruling the SPACE parameter.

Note that EOT=YES overrules the SPACE parameter only when real tapes are used.

LABEL=(,[*type*],,[OUT|IN][,RETPD=*days*])
Label information for the data set:

type
Label type, one of: SL, SUL, AL, AUL, NSL, NL, BLP, or LTM. See the appropriate *MVS JCL Reference* manual for descriptions of these label types. The default is the IBM standard label (SL).

OUT
Output file.

IN
Input file. Do not specify IN for application real-time sequential files, the ALCS database update log, or ALCS diagnostic file.

RETPD=*days*
Retention period in days (see the appropriate *MVS JCL Reference* manual).

LRECL=*lrecl*
Actual record length in bytes if fixed-length records; maximum record length in bytes, if variable-length records. LRECL is ignored if RECFM is not specified. If RECFM is specified, specify LRECL, depending on the record format, as follows:

Fixed-length unblocked
Omit LRECL or specify *lrecl* equal to the block size.

Fixed-length blocked
For real-time sequential files with time-stamps, remember to allow for the time-stamp in the logical record length. For example, if your application programs write 80-byte records, you should specify LRECL=96 (80 bytes of user data plus 16 bytes of time-stamp).

Variable-length
Omit LRECL (*lrecl* defaults to 4 less than the block size) or specify *lrecl* as the maximum record length. The minimum for files without time-stamps is 5, the minimum for files with time-stamps is 21. The maximum is 4 less than the block size.

Undefined
Omit LRECL.

MEMBER=*member*
Name of the configuration dependent table that describes the ALCS sequential file configuration. The stage 2 link-edit step uses this name for the sequential file configuration table load module.

Specify **MEMBER**=*member* on only one SEQGEN macro in every sequential file generation. The default name is DXCCDQ*iv*, where *i* and *v* are the identifier and version specified on the ALCS macro.

MGMTCLAS=*management_class_name*
SMS management class for the data set (1 to 8 characters). This can only be used if the Storage Management Subsystem is active.

PROTECT={NO|YES}
Resource access control facility (RACF) protection for the data set or tape volume (see the appropriate *MVS JCL Reference* manual):

NO
Do not protect the data set or tape volume.

YES
Protect the data set or tape volume. During execution, ALCS ignores this parameter when RACF is not installed or is not active.

RECFM={F|FB|V|VB|U}[A|M]
Record format of the data set (see the appropriate *MVS JCL Reference* manual). Note that ALCS does not support all the record formats that the manual describes; for example, ALCS does not support **RECFM**=D or **RECFM**=VT.

The RECFM parameter is mandatory when ALCS cannot determine the record format from the data set label. That is, when one or more of the following applies:

- File is a dummy data set
- File is a SYSOUT data set
- File is unlabeled (NL or LTM) or bypass label processing (BLP)
- Data set status is NEW.

Specify either **RECFM**=V or **RECFM**=VB for the following files:

- ALCS database update log
- ALCS diagnostic file
- ALCS data collection file
- ALCS user file

BLKSIZE is mandatory for undefined-length records (RECFM is U, UA, or UM).

SPACE=(*primary_blocks*[,*secondary_blocks*])
Space required for the file and automatic file switch threshold, where:

primary_blocks
Number of data blocks in the data set primary allocation.

For data sets that are on DASD, when ALCS allocates the file, MVS rounds this primary allocation up to an integral number of cylinders. When ALCS deallocates the file, MVS releases unused space.

For application real-time sequential files, the ALCS database update log, and the ALCS diagnostic file, *primary_blocks* (unrounded) is also the automatic data set switch threshold. After writing *primary_blocks* blocks to a data set, ALCS initiates a data set switch. When the data set switch completes, ALCS writes subsequent blocks to the new data set.

For data sets that are not on DASD, and for SYSOUT data sets, *primary_blocks* does not specify space requirements; it specifies only the automatic data set switch threshold. Specify *primary_blocks* according to what is operationally convenient. For example, for the ALCS diagnostic file, ensure that the data set size is at least large enough to hold a dump. For other sequential files, set up single volume data sets so that ALCS switches data sets automatically at the end of the volume. To achieve this, specify *primary_blocks* as 80 percent of the capacity of the volume.

secondary_blocks
Number of data blocks in the data set secondary allocation.

The appropriate *MVS JCL Reference* manual describes primary and secondary space allocation, but note that the format of this SPACE parameter is different from the format of the MVS JCL DD statement SPACE= parameter.

STORCLAS=*storage_class_name*
SMS storage class of the data set (1 to 8 characters). This can only be used if the Storage Management Subsystem is active.

SYSOUT=*class*
File is a system output (SYSOUT) data set, *class* is the output class.

See the appropriate *MVS JCL Reference* manual for descriptions of system output data sets and output classes.

TRTCH={COMP|NOCOMP}
Tape unit requested. Use COMP to request an Improved Data Reading Capability (IDRC) tape unit and NOCOMP to request a non-IDRC tape unit.

UNIT=(*type*[,*count*])
Device type and optionally number of devices for the data set, where:

type
Device number, device type, or unit name for the data set.

count
Unit count for the data set.

The appropriate *MVS JCL Reference* manual describes device numbers, device types, unit names, and unit counts.

VOLUME=(,,,[*volcnt*][,SER=*volser*])
Volume information for the data set, where:

volcnt
Maximum number of volumes that an output data set requires. A decimal number from 1 through 255.

volser
Volume serial of the volume where the data set resides.

You can code only one volume serial in the SEQGEN macro. To read multivolume data sets, you must first catalog the data set by means of a job such as the following:

```
//STEP    EXEC PGM=IEFBR14
//OUT     DD   DSN=data set_name,DISP=(OLD,CATLG,CATLG),
//             UNIT=(TAPE,,DEFER),
//             VOLUME=(,,,2,SER=(volser1,volser2,...))
```

Refer to the description of the DD statement VOLUME= parameter in the appropriate *MVS JCL Reference* manual.

USE=*alt*
This sequential file is an alternative name for the *alt* sequential file. When an application program refers to the sequential file name specified in the NAME parameter of this macro, ALCS processes the request as if the application refers to *alt*. For example, applications can use different sequential file names to refer to the same file. Figure 45 on page 172 shows three different references to the same file.

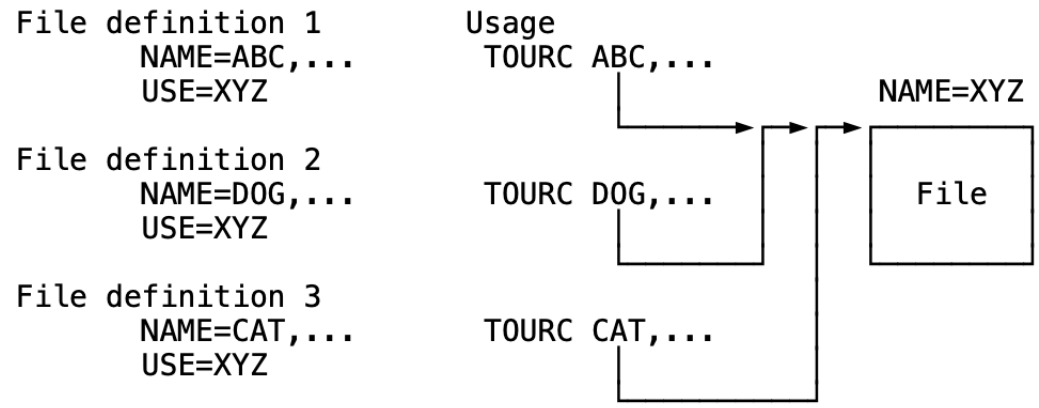


Figure 45. The USE parameter: Alternative names

Generating a DASD configuration table

When you start ALCS you need to specify a DASD configuration table as part of the system definition. The first time you start ALCS Version 2 Release 4.1 this table is automatically copied onto your configuration data set and becomes your initial DASD configuration.

As your system expands you may need to change the allocation of records on the real-time database. To do this you need to generate a new DASD configuration, and load it onto the configuration data set using the `ZDASD LOAD` command followed by `ZDASD CONFIRM` and `ZDASD COMMIT`.

ALCS Operation and Maintenance describes the ZDASD command.

To generate a DASD configuration table you need to create a DASD generation source member, and then use the ISPF panels to complete the DASD generation. Chapter 1, “Using ALCS ISPF panels,” on page 1 shows where to find more information about the ALCS ISPF panels.

A DASD generation source member is a list of calls of the following macros:

DBGEN
Specifies the size and general characteristics of the database. Include only one `DBGEN` macro in each generation.

DBHIST
Ensures that changes to the database are applied in the correct sequence and also provides an audit trail.

DBSPACE
Increases the total number of addressable records on the real-time database.

USRDTA
Specifies details of a database record type (fixed file or pool file). You can include several `USRDTA` macros in each generation, one for each record type. See “ALCS record requirements” on page 174 for more information about the records required by ALCS.

GFGEN
Specifies details of a general file. You can include several `GFGEN` macros in each generation, one for each general file.

In addition, two generation macros are available to assist in the installatio of IPARS - ALCS V2 databases. Examples are supplied on the shipment tape:

RESDTA
Specifies details of the IPARS - ALCS V2 reservations database.

MSWDTA
Specifies details of the IPARS - ALCS V2 message switching database.

The ALCS DASD generation provides summary information about your database definitions, including:

- Total record counts
- Details of record IDs and file address reference bands
- Database capacity for the:
  - initial configuration
  - updated configuration
    or
    allocatable pool configuration (if you specify `DBHIST DISPENSE_TYPE2_LONG_TERM`)
- Database data set space requirements

Specifying the initial DASD configuration

When you first use ALCS Version 2 Release 4.1 you need to generate an initial DASD configuration table. This consists of:

- A `DBGEN` macro to define the general characteristics of the database

- Macros to define the files required by your application:
  - GFGEN macros to define each general file
  - USRDTA macros to define each pool file type
  - USRDTA macros to define each fixed file type
  - USRDTA macros to define the files required by ALCS (“ALCS record requirements” on page 174 lists these files)

Any general-file, fixed-file or pool-file type which you include must specify at least one band number. “ALCS band format” on page 176 describes band-format file addressing.

Terminate your initial ALCS DASD generation with either:

- DBHIST DEFINE_NEW
- DBHIST DEFINE_MIGRATE_FROM

This signals to ALCS that any further macros are updates. “DBHIST macro” on page 195 describes these parameters.

ALCS record requirements

ALCS generation allocates #KPTRI records for system use. USRDTA cannot alter the number of #KPTRI records that ALCS allocates.

ALCS also requires the following records:

ID	Type	Size	VFA options
X'AC00'	#CPRCR	L3	FI
X'AC01'	L*n*LTPOOL	Any	FI UP
	L*n*STPOOL	Any	DF
X'AC02'	L3STPOOL	L3	DF
X'AC03'	L3LTPOOL	L3	FI UP
X'AC04'	L*n*LTPOOL	Any	FI UP
X'AC05'	L3LTPOOL	L3	FI UP
X'AC06'	L*n*STPOOL	L1, L2, L3	DF
X'AC07'	L3LTPOOL	L3	FI UP
X'AC08'	L*n*LTPOOL	Any	FI UP
X'AC09'	L*n*LTPOOL	Any	FI UP
	L*n*STPOOL	Any	DF
X'AC10'	L3STPOOL	L3	DF
X'AA00'	L3LTPOOL	L3	FI UP
X'AA01'	L*n*STPOOL	Any	DF
X'AA02'	L1LTPOOL	L1	FI UP
X'AA10'	L3LTPOOL	L3	FI UP
X'AB10'	L3LTPOOL	L3	FI UP
	L3STPOOL	L3	DF

Include USRDTA macros to specify details of the above record types. Use the following guidelines to decide the number of each record type to specify:

X'AC00'
Allocate one record for each communication resource, plus an allowance for growth.

X'AC01'
ALCS printer support and LU 6.1 support write messages to these records before sending the messages. You are recommended to use long-term pool file records. The number of records required for this purpose depends on the volume of traffic and the frequency of Recoup runs. Choose a pool with a record size big enough to contain the majority of messages in one pool file record.

Note: If the requested pool has less than 500 records available or less than 5 minutes' worth of records available at the current rate of usage, then ALCS tries to get a record from a long-term pool (it tries L1LTPOOL, then L2LTPOOL, and so on up to L8LTPOOL).

If no long-term pool file records are available, it uses an L1 short-term pool file record. The defaults of 500 records available and 5 minutes' worth of records available can be changed using the installation-wide exit APR6 (see “Set thresholds for long-term pool dispensing - APR6” on page 386).

X'AC02'
ALCS SLC support writes messages to L3STPOOL records before sending the messages. The number of L3STPOOL records required for this purpose depends on the volume of traffic across the SLC link.

X'AC03'
ALCS SLC support writes messages to these records if the application requests output Type B message queueing services. Allocate seven records for each SLC link.

X'AC04'
ALCS BATAP writes messages to these records before sending the messages. The number of records required for this purpose depends on the volume of traffic across the X.25 PVC or TCP/IP connection and the frequency of Recoup runs. Choose a long-term pool with a record size big enough to contain the majority of messages in one pool file record.

X'AC05'
ALCS PF key support uses a long-term pool file record to store the PF key settings for a given terminal. The pool file record is attached to the associated Resource Control Record. The assignment of the pool file record occurs when the first PF key setting for a terminal is defined. The number of PF key records required depends on the number of 3270 type terminals. See “User-programmable PF key exit program - AKY1” on page 372 for details. There can only be one PF key record per resource.

Note: If there are less than 8000 L3 long term records available, ALCS does not assign new PF key records, but replies with an error message.

X'AC06'
ALCS scrollable output support uses short-term pool file records of sizes L1, L2 and L3. ALCS chooses the smallest size of pool file record in which the display data fits.

X'AC07'
The ALCS 3270 screen mapping support package uses the records to contain the ALCS internal description of the 3270 screen map. You must allow two records for each screen map loaded on the system.

X'AC08'
ALCS Retrieve facility uses these records for saving input messages. One record is used for each resource using the retrieve facility. Allocate enough records to accommodate the number of users who will be using this facility. You are recommended to use long-term pool file records. The record is assigned when the first input message is received after retrieve has been made active for a particular resource. The record size determines how many messages the retrieve function can save for each end user. Retrieve imposes a fixed overhead of 34 bytes, plus 4 bytes for each message. For example, if the average length of input messages is 10 characters:

- If you allocate X'AC08' as a 1055-byte L2 record, each end user can save (1055-34)/(10+4) = 72 messages.

- If you allocate X'AC08' as a 4000-byte L3 record, each end user can save (4000-34)/(10+4) = 283 messages.

X'AC09'
ALCS BSC support writes messages to these records before sending the messages. You are recommended to use long-term pool file records. The number of records required for this purpose depends on the volume of BSC traffic and the frequency of Recoup runs. Choose a pool with a record size big enough to contain the majority of messages in one pool file record.

Note: If the requested pool has less than 500 records available or less than 5 minutes' worth of records available at the current rate of usage, then ALCS tries to get a record from a long-term pool (it tries L1LTPOOL, then L2LTPOOL, and so on up to L8LTPOOL). If no long-term pool file records are available, it uses an L1 short-term pool file record.

X'AC10'
ALCS online communication table maintenance (OCTM) uses these short-term pool records to store information about communication resource groups. These records are only used when OCTM is active on your ALCS system (`COMGEN OCTM=YES`).

X'AA00'
ALCS hierarchical file system (HFS) uses these pool records to hold HFS directory information.

X'AA01'
ALCS hierarchical file system (HFS) uses these pool records to hold input and output message data.

X'AA02'
ALCS hierarchical file system (HFS) uses these pool records to hold long-name information.

X'AA10'
ALCS hierarchical file system (HFS) uses these pool records to hold HFS state information for each end user.

X'AB10'
The ALCS e-mail facility uses these records for saving the contents of inbound and outbound e-mail messages.

- While ALCS is receiving an inbound e-mail message it saves the contents in an L3STPOOL record.
- The ALCS e-mail queue handler writes outbound messages to L3LTPOOL records before sending the messages.

For more information on what you can and cannot define in your database generation, see “Reserved symbols and values” on page 447.

ALCS band format

ALCS uses a file-address format called a **band format**. The band-format file address consists of two binary numbers (or parts):

- The **band** part
- The **band ordinal** part

The number of records within a record type can vary enormously from one record type to another. For example, one record type can contain two records and another 2 000 000 records. ALCS constructs file addresses most efficiently when there are few bands allocated for each record type.

To allow for this, a band can be 1 byte (addressing up to 16 777 216 records), 2 bytes (addressing up to 65 536 records), or 3 bytes (addressing up to 256 records). The ALCS generation can allocate different sized bands for the same record type. For example, it can allocate one 1-byte band and one 2-byte band for a record type. The two bands can address up to 16 842 752 records.

ALCS provides three band formats to suit the number of records in a record type. Figure 46 on page 177 shows the band formats and the maximum number of ordinals allowed in each format.

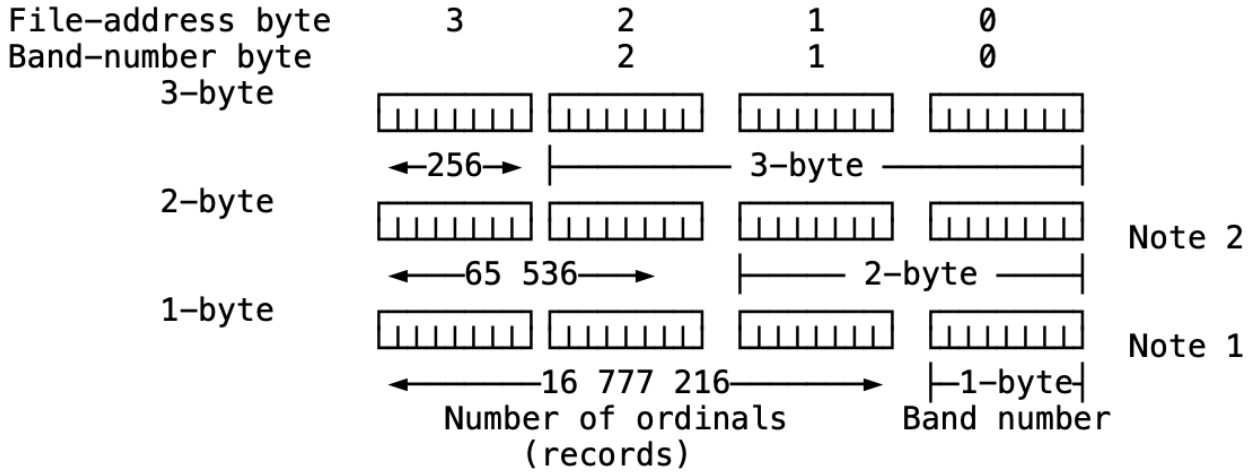


Figure 46. ALCS band-format for a 4-byte file address

Notes:

1. The value of byte 0 of a 1-byte band cannot be the same as the value of byte 0 of any 2-byte or 3-byte band. For example, if hexadecimal F0 is a 1-byte band, then hexadecimal F000 is not a valid 2-byte band.
2. The value of byte 0 and byte 1 of a 2-byte band cannot be the same as the value of byte 0 and byte 1 of any 3-byte band. For example, if hexadecimal 4020 is a 2-byte band, then hexadecimal 402030 is not a valid 3-byte band.

The ALCS system programmer:

- allocates one or more bands for each record type and specifies them in the ALCS system generation.
- never allocates the same band to more than one record type.
- must allocate more than one band, if the number of records of that type is greater than the number of ordinals in the first band.

Selecting file address bands

The ALCS generation requires parameters that specify one or more bands for every record type. You should adopt a systematic approach to the allocation of band numbers, such as that described in this section. You must define bands for:

- User-defined USRDTA macros
- IBM-required USRDTA macros
- User-defined GFGEN macros
- User-defined DBGEN macros

All valid record sizes require one or more bands in the DBGEN or DBHIST macro.

Note: Do not use X'00' as byte 0 (the first byte or only byte) of any band. Bands that start with X'00' are reserved for the following IBM usage:

0000 through 001F
Reserved for fixed-file records required by ALCS. For example, #KPTRI and #CPRCR.

0020 through 00FF
Reserved for ALCS system records. For example, system fixed file.

- **Fixed files**

  If the number of records is not more than, for example, 256K, allocate one or more 2-byte bands and construct them in accordance with Table 27 on page 178. The record type number is in the DXCURID macro (see “ALCS DXCURID user macro” on page 443).

Table 27. Example method for selecting 2-byte bands for fixed files			
Number of records	**Record type no.**	**Band byte 0**	**Band byte 1**
<64K	<256	X'01'	Record type number
<64K	256 - 511	X'21'	Record type number minus 256
<64K	512 - 767	X'31'	Record type number minus 512
<64K	768 - 1023	X'41'	Record type number minus 768
>64K	Any	X'11'	See note
This method uses a series of hexadecimal numbers, one for each 64K (or part of 64K) records allocated to the fixed record type. For example, if the number of records is 130K, you allocate three hexadecimal numbers (for example, 01, 02, and 03).			

If the number of records allocated to a fixed-file record type is more than, for example, 256K, allocate a 1-byte band in the range X'10' to X'E0'.

- **Short-term pool files**

  Note: If you wish to dynamically add short-term pool files then it is only possible with type 2 short term pool.

  Allocate a 2-byte band and construct it as follows:

  Byte 0
  X'09'

  Byte 1
  A hexadecimal value that defines the record size for the short-term pool:

  X'10' for the L1ST pool
  X'20' for the L2ST pool
  X'30' for the L3ST pool

  up to X'80' for the L8ST pool. If the number of records allocated to the short-term pool is greater than 64K, allocate additional hexadecimal numbers in the range 11 to 1F for the L1ST pool, 21 to 2F for the L2ST pool, and so on.

- **Long-term pool files**

  Allocate a 1-byte band. Use a hexadecimal value that defines the record size for the long-term pool:

  X'08' for the L1LT pool
  X'18' for the L2LT pool
  X'28' for the L3LT pool

  up to X'78' for the L8LT pool.

- **General files**

  Allocate a 2-byte band and construct it as follows:

  Byte 0
  X'F0'.

  Byte 1
  The general file number, as defined in “GFGEN macro” on page 193.

- **Allocatable pool**

  Allocate a 1-byte band for each record size that you are using. Use the DBGEN macro to specify the initial allocatable file band values but use the DBSPACE macro to specify any additional bands.

  Choose hexadecimal values that do not conflict with any existing long-term pool bands. For example,:

X'88' for the L1 allocatable pool
X'89' for the L2 allocatable pool
X'8A' for the L3 allocatable pool

- **Multiple file address format considerations**

  ALCS multiple file address format support allows you to migrate records from an existing TPF or ALCS database that uses a different file addressing method from ALCS V2.

  ALCS Concepts and Facilities gives an overview of the ALCS multiple file-address format support

  To make this possible, you must ensure that there are no ambiguous file addresses on the ALCS V2 database. For example, if you are migrating records from a system that uses the TPF file address reference format 3 (FARF3), then you must ensure that no valid FARF3 file address is also a valid band format file address. Similarly, if you are migrating records from a system that uses the ALCS/VSE record ordinal number (RON) file address format, then you must ensure that no valid RON file address is also a valid band format file address. If you do not ensure this then ALCS V2 will interpret some FARF3 (or RON) file addresses as band format file addresses and access the wrong records.

  The method just described for allocating band ordinal numbers always sets bits 29 and 30 of the band format file address (bits 5 and 6 of band byte 0) to zeros. All valid FARF3 and RON format file addresses have one or other of these bits set to 1 - both formats have bit 29 set to 1 for general file record addresses, and bit 30 set to 1 for fixed file and pool file record addresses.

  Even if you are not migrating records from an existing database that uses a different file address format, it is important to reserve one or more bits of the file address. At some time in the future, IBM may introduce a new file address format and you will need some way to distinguish between your existing ALCS Version 2 file addresses and the new format.

Updating the DASD configuration

Use the following macros to specify changes to your DASD configuration:

```
GFGEN
USRDTA
DBHIST
DBSPACE
```

Place the additional macroinstructions after any existing macros in your DASD generation source member and rerun the DASD generation process (use the DASD generation option in Chapter 1, “Using ALCS ISPF panels,” on page 1).

If you change any of the existing macros you cannot load the DASD configuration table that you generate.

The additional macros must include at least one call to `DBHIST` which should be the first in the batch (of additional macros). You can then follow the `DBHIST` macro with `GFGEN`, `USRDTA`, and `DBSPACE` macros.

Use the ZDASD command to load the new configuration. *ALCS Operation and Maintenance* describes the ZDASD command.

Fixed-file records

- **Adding a new fixed file type**

  1. For each new fixed-file record type, choose a fixed-file record type symbol. The convention is to use 6- or 7-character symbols; the first character is a hash character (#[9]). For example, to add currency conversion records allocate the fixed record type symbol `#CCTRI`. IBM reserves some fixed-file record type symbols.

     For each new fixed-file record type, choose a fixed-file record type number. IBM reserves fixed-file record type number 0.

[9] This hash (#) character is represented differently by some equipment. It is the EBCDIC hexadecimal '7B' character.

2. Update your fixed-file record type equates macrodefinition, DXCURID, to include the new fixed-file record type symbols and associated fixed file record type numbers.
3. Assemble all application programs that refer to the new fixed file-record type symbols. Ensure that the assembler accesses the updated version of the DXCURID macro. For information on updating DXCURID and on reserved fixed-file record types, see “ALCS DXCURID user macro” on page 443.
4. Decide the record identifier(s) for each new fixed-file record type. The record identifier is the 2 alphanumeric or 4 hexadecimal characters that occupy the first 2 bytes of a record. IBM reserves some record identifiers, see “Reserved symbols and values” on page 447.

   Decide any changes to the record identifiers associated with existing fixed record types.
5. Decide on the record addressing band number. “Selecting file address bands” on page 177 gives more information about selecting bands.
6. Decide on the VFA options.
7. Decide on the number of ordinals required.
8. Include a call to the USRDTA macro in your ALCS database generation.

Use the information in steps “1” on page 179 through “7” on page 180 to specify the required parameters. Refer to “USRDTA macro” on page 190 for more details of these parameters.

For example, to add a new fixed file type #IASC1 with 100 size L2 records, a band number of 4044 and a VFA option of file immediate, include a USRDTA macro like:

```
USRDTA ACTION=ADD,TYPE=#IASC1,BAND=4044,VFAOPT=FI,SIZE=L2,NUMBER=100
```

- **Increasing the number of records in a fixed file type**

  To increase the size of a fixed file type, include a USRDTA macro and specify ACTION=ADD and the NUMBER= parameter.

  For example, to increase type #IASC1 to 200 records include a USRDTA macro like:

```
USRDTA ACTION=ADD,TYPE=#IASC1,NUMBER=100
```

  Notes:

  1. If you have specified

```
DBHIST DEFINE_MIGRATE_FROM
```

     but have not yet included a

```
DBHIST DISPENSE_TYPE2_LONG_TERM
```

     macro in your generation, you can specify ACTION=SPILL instead of ACTION=ADD

     This places the additional records on a spill data set and they are still accessible if you fall back to a previous version of ALCS.

     However adding spill data can lead to an unbalanced access pattern.

     Attention Adding spill data may increase the number of required data sets. In this case, the change does not take effect until you use ZDASD VARY,ON to vary the new data set on line.
  2. If you specify ACTION=ADD (instead of SPILL), ALCS allocates space from the existing long-term pool file. The records you add are lost when you fall back to a previous version of ALCS.
  3. Once you specify ACTION=ADD as part of a DASD update, you cannot specify ACTION=SPILL in a subsequent update of the same fixed file type.
- **Decreasing the number of records in a fixed file type (delete records)**

To reduce the number of records in a fixed file type, include a USRDTA macro and specify ACTION=DELETE and the NUMBER= parameter. This marks the deleted records in preparation for a subsequent purge. Any access by an application program to a deleted record succeeds, but writes a **record usage error** to the diagnostic file.

For example,

```
      USRDTA TYPE=#IASC1,ACTION=DELETE,NUMBER=50
```

- **Purging deleted fixed-file records**

  To purge any records previously marked as deleted, include a USRDTA macro and specify ACTION=PURGE without the NUMBER= parameter.

  This purges all the marked records. Any access by an application program to a purged record results in a **retrieval error**.

  For example,

```
      USRDTA TYPE=#IASC1,ACTION=PURGE
```

- **Changing the VFA options**

  To change the VFA options, specify the new options with the VFAOPT= parameter.

  For example,

```
      USRDTA TYPE=#IASC1,ACTION=ADD,VFAOPT=(DF,HY)
```

- **Adding or deleting a record ID**

  To add or delete a record ID, include a USRDTA macro and specify ACTION=ADD or ACTION=DELETE and the ID= or IDSYM= parameter.

  For example,

```
      USRDTA TYPE=#IASC1,ACTION=ADD,ID=FX
```

- **Adding a new file address band**

  To add a new file address band, include a USRDTA macro and specify the ACTION=ADD and the BAND= parameter.

  For example,

```
      USRDTA TYPE=#IASC1,ACTION=ADD,BAND=4045
```

- **Changing the record size of a fixed file type**

  You can change the record size of a fixed file type but first you must delete and purge all the records of the previous size.

  1. Reduce the number of records in the fixed file type to zero using

```
      ACTION=DELETE
```

     as described previously.
  2. Purge the records you have marked as deleted using

```
      ACTION=PURGE
```

     as described previously.

3. Add new records of the appropriate size using

```
ACTION=ADD
```

   with new NUMBER= and SIZE= parameters.

Note: You may also change the VFA options, but you cannot change the band used for this file type. The new records inherit the previous band structure. However you can (of course) add new bands if you so wish. To add a new fixed record type symbol you must update the ALCS DXCURID user macro.

- **Converting a fixed file type to table-based addressing**

  To convert a fixed file type to table-based addressing, include a USRDTA macro and specify ACTION=BUILD. This causes ALCS to build the appropriate tables to support table-based addressing instead of algorithm-based addressing.

  For example,

```
USRDTA TYPE=#IASC1,ACTION=BUILD
```

  Alternatively, to convert all fixed file and short-term pool types to table-based addressing specify BUILD_DIRECTORIES as the parameter in the DBHIST macro call.

  For example,

```
DBHIST BUILD_DIRECTORIES,TEXT='Change all types to table-based'
```

  Note: The change takes effect only when you commit the DASD generation using ZDASD COMMIT.

Short-term pool file records

- **Adding a new short-term pool file type**

  1. Decide the record identifier(s) for each new pool file record type. Decide any changes to the record IDs for existing pool file record types. IBM reserves some record identifiers; “Reserved symbols and values” on page 447 gives details.

     Avoid using the same record ID for more than one pool file record type, or for both a fixed-file record type and a pool file record type. Instead, assign a record identifier uniquely to one pool file record type or to one fixed-file record type. This simplifies the use of the monitor service. (You can use more than one record ID in a single pool.)
  2. Decide on the number of ordinals required.
  3. Decide on the record addressing band number. “Selecting file address bands” on page 177 gives more information about selecting bands.
  4. Include a call of the USRDTA macro in your ALCS database generation.

  Use this information to specify the required parameters. Refer to “USRDTA macro” on page 190 for more details of these parameters.

  For example, to add a new L3 short-term pool with 9000 records, with record IDs of FR and FS, and a band number of 3003 include a USRDTA macro like:

```
USRDTA ACTION=ADD,TYPE=L3ST,BAND=3003,NUMBER=9000,ID=(FR,FS)
```

  Note: The change takes effect only when you commit the DASD generation (using ZDASD COMMIT) and then run Recoup.

- **Increasing the size of a short-term pool**

  To increase the size of an existing short term pool, include a USRDTA macro with ACTION=ADD and the NUMBER= parameter.

Note: The maximum size of each short-term pool is 15,724,800 records.

For example, to add 200 L3 short-term pool records include a USRDTA macro like:

```
      USRDTA TYPE=L3ST,ACTION=ADD,NUMBER=200
```

Note: The change takes effect only when you commit the DASD generation (using ZDASD COMMIT) and then run Recoup.

- **Reducing the size of a short-term pool** (delete records)

  To reduce the size of a short term pool, include a USRDTA macro and specify ACTION=DELETE and the NUMBER= parameter. This marks the deleted records in preparation for a subsequent purge and ensures they are not dispensed.

  They are still accessible until they are purged.

  For example,

  ```
      USRDTA TYPE=L3ST,ACTION=DELETE,NUMBER=50
  ```

- **Purging deleted short-term pool records**

  To purge any records previously marked as deleted, include a USRDTA macro and specify ACTION=PURGE without the NUMBER= parameter. This purges all the marked records.

  For example,

  ```
      USRDTA TYPE=L3ST,ACTION=PURGE
  ```

- **Adding or deleting a short-term pool record ID**

  To add or delete a record ID, include a USRDTA macro and specify ACTION=ADD or ACTION=DELETE and the ID= or IDSYM= parameter.

  For example,

  ```
      USRDTA TYPE=L3ST,ACTION=ADD,ID=(FA,FB)
  ```

- **Adding a new file address band**

  To add a new file address band, include a USRDTA macro and specify ACTION=ADD and the BAND= parameter.

  For example,

  ```
      USRDTA TYPE=L3ST,ACTION=ADD,BAND=3004
  ```

- **Converting a short-term pool-type to table-based addressing**

  To convert a short-term pool-type to table-based addressing, include a USRDTA macro and specify ACTION=BUILD. This causes ALCS to build the appropriate tables to support table-based addressing instead of algorithm-based addressing.

  For example,

  ```
      USRDTA TYPE=L2ST,ACTION=BUILD
  ```

  Alternatively, to convert all fixed file and short-term pool types to table-based addressing specify BUILD_DIRECTORIES as the parameter in the DBHIST macro call.

For example,

```
DBHIST BUILD_DIRECTORIES,TEXT='Change all types to table-based'
```

Note: The change takes effect only when you commit the DASD generation using ZDASD COMMIT.

Long-term pool file records

Once you specify DBHIST DISPENSE_TYPE2_LONG_TERM in your DASD generation you can increase long-term pool as follows:

- Increase the size (and if necessary, the addressability) of one or more database data sets

or

- Add one or more data sets

Note: The change takes effect only when you commit the DASD (using ZDASD COMMIT) generation and then run Recoup.

- *(INCRTDS)* **Increasing the size of a real-time data set**

  Use the following procedure to increase the size of a real-time data set:

  1. Use ZDASD VARY,OFFLINE to **vary offline** copy 2 of the data set. *ALCS Operation and Maintenance* describes the ZDASD command.
  2. Delete the copy-2 data set
  3. Reallocate the data set at the required size (use the DASD generation ISPF panel)
  4. Use ZDASD VARY,ONLINE to **vary online** the new copy-2 data set. ALCS preformats the data set and copies the data from the copy-1 data set.
  5. Use ZDASD VARY,OFFLINE to **vary offline** the copy-1 data set

     Note: At this point in the process, use the ZDASD D,DS command to check the size of the copy-1 data set. This is shown as "actual" size in the display.
  6. Delete the copy-1 data set
  7. Reallocate the data set at the required size (use the DASD generation ISPF panel)
  8. Use ZDASD VARY,ONLINE to **vary online** the new copy-1 data set. ALCS preformats the data set and copies the data from the copy-2 data set.

     Both copies are now the same increased size.
  9. Run Recoup; this flags the additional space as available.

- *(INCRTAD)* **Increasing the addressability of the real-time data sets**

  The DBGEN macro determines the initial number of segments of addressability for each database data set. This defines the number of records in multiples of 65 536 (64K) which can be addressed on each data set.

  After you increase the size of a real-time data set, it is possible that the increased size is still within the original addressability range. In this case you do not need to change the DASD configuration.

  Otherwise you need to add a DBSPACE macro to your DASD generation.

  For example, to increase the addressability of each of the existing L3 data sets by 2 segments (128K records) include a DBSPACE macro in your DASD generation like:

  ```
  DBSPACE SIZE=L3,SEGMENTS=2
  ```

- *(INCRTNU)* **Increasing the number of the real-time data sets**

  Use the following procedure to increase the number of real-time data sets:

1. Run a DASD generation including a DBSPACE macro. Use the VOLUMES= parameter to specify the number of additional volumes required.

   For example,

   ```
   DBSPACE SIZE=L3,VOLUMES=1
   ```

2. Allocate copy 1 and copy 2 for each of the additional data sets (use the DASD generation ISPF panel).
3. Use ZDASD VARY,ONLINE to **vary on line** both copies of each data set.

   If the copies are different sizes you must vary the smaller copy first, otherwise ALCS rejects the smaller copy with a size error.
4. Ensure that you have entered ZDASD COMMIT.
5. Run recoup; this flags the space in the additional volumes as available.

Note: You cannot decrease the amount of long-term pool for any type.

- **Adding or deleting a long-term pool record ID**

  To add or delete a record ID, include a USRDTA macro and specify ACTION=ADD or ACTION=DELETE and the ID= or IDSYM= parameter.

  For example,

  ```
  USRDTA TYPE=L2LT,ACTION=ADD,ID=(MX,MY)
  ```

Adding a new DASD record size

About this task

Use the following procedure to add a new DASD record size:

Procedure

1. Update the ALCS configuration. Refer to “Updating the ALCS configuration” on page 68 for more information.
2. Include a DBSPACE macro in your DASD generation to specify the number of volumes and the number of segments of addressability for each volume.

   For example to add 2 data sets for size L5 with 3 segments of addressability (196 608 records) include a DBSPACE macro like:

   ```
   DBSPACE SIZE=L5,VOLUMES=2,SEGMENTS=3,BAND=F5
   ```

3. Add USRDTA macros as required to define your new records. Refer to “Fixed-file records” on page 179 and “Short-term pool file records” on page 182 for more information about the USRDTA macro.
4. Generate and load the new DASD configuration table.
5. Allocate two data sets, copy-1 and copy-2, for each of the volumes added by the DBSPACE macro (use the DASD generation ISPF panel).
6. Use ZDASD VARY,ONLINE to vary online the new copy-1 data sets. ALCS preformats the data sets before bringing them online. Repeat the process for the copy-2 data sets.
7. Run Recoup; this flags the additional space as available.

General files

- **Adding a general file**

To add a new general file run a DASD generation include a GFGEN macro to specify the data set name and record size.

For example, to add a data set named DXC.GF3 as general file 3 with L1 record size, include a GFGEN macro like:

```
      GFGEN NUMBER=3,DSNAME='DXC.GF3',SIZE=L1,BAND=2005
```

- **Adding a new file-address band to a general file**

  To add a new file address band, include a GFGEN macro and specify the ACTION=ADD and the BAND= parameter.

  For example,

```
      GFGEN NUMBER=3,ACTION=ADD,BAND=3004
```

- **Deleting a general file**

  To delete a general file, include a USRDTA macro and specify the ACTION=DELETE parameter.

  For example, to delete the general file added above include a GFGEN macro like:

```
      GFGEN NUMBER=3,ACTION=DELETE
```

- **Changing attributes of a general file**

  To change the record size, the VFA options, or the initial status of a general file, add the new attribute(s) using the GFGEN ACTION=ADD parameter.

  For example, to change the VFA options of general file number 6, include a GFGEN macro like:

```
            GFGEN ACTION=ADD,NUMBER=6,VFAOPT=FI
```

DBGEN macro

Use DBGEN to specify general information about the ALCS real-time database. For example:

- The name of the DASD component configuration module
- Details of the data sets used for the database.

Include DBGEN once in the stage 1 input deck. The generation ignores subsequent DBGEN macros.

```
[label]  DBGEN  VOLUMES=({L1_vols],[L2_vols],...)
                [,BAND=((L1_band1[,L1_band2,...]),(L2_band1[,L2_band2,...])...)]
                [,CDNAME=(name_prefix)]
                [,CDS1={NO|YES}]
                [,CDS2={NO|YES}]
                [,DSNAME=(name_list)]
                [,DUPLEX={YES|NO}]
                [,FARMIG={NONE|USER|USERCODE|VSERON|FARF3}]
                [,MEMBER=name]
                [,PARTITION=([0|n})
                [,POOLMIG=([0|L1_non_dup},{0|L1_dup},...)]
                [,SEGMENTS=({L1_segs],[L2_segs],...)]
```

Where:

label
Any valid assembler label.

VOLUMES= ({*L1_vols*], [*L2_vols*],...)
Number of database data sets in the **initial** configuration (**excluding** spill data sets) for each record size. When you are migrating from ALCS Version 2.1.1 or ALCS/MVS/XA, the initial configuration is the *migrate-from* database.

(*L1_vols*,*L2_vols*,...)
Number of data sets for the L1, L2, ..., record sizes respectively.

Note: ALCS does not allow modification of this parameter. Use the DBSPACE macroinstruction to add more volumes.

BAND=((*L1_band1[,L1_band2,...*]),(*L2_band1[,L2_band2,...*])...)
The file-address bands in the allocatable pool file space (for each record size) in the **initial** configuration. At least one band is required for each defined record size.

Do not change these parameters in later generations, use the DBSPACE macro to add more bands.

"Specifying the initial DASD configuration" on page 173 describes the ALCS band addresses and reserved bands.

CDNAME=(*name_prefix*)
Data set name prefix for the configuration data sets. This prefix is for the data component name, not the VSAM cluster name (see "Defining and initializing the database and general file data sets manually" on page 461). If you specify one name in the DSNAME parameter, then that name is the default name for CDNAME. Otherwise the default data set name prefix is DXC*iv*, where *i* is the ALCS system identifier and *v* is the ALCS system version.

The data set name prefix(es) are used to construct the data set names for the three configuration data sets, CDS0, CDS1 and CDS2.

The data set name for the data component of CDS0 (database configuration data set) is: *prefix*.CD.C*c*.

The data set name for the data component of CDS1 (program configuration data set) is: *prefix*.CP.C*c*.

The data set name for the data component of CDS2 (communication configuration data set) is: *prefix*.CC.C*c*.

Note: ALCS does not allow dynamic modification of this information.

CDS1={NO|YES}
Whether the program configuration data set, CDS1, is used by this ALCS system.

NO
The program configuration data set, CDS1, is not used by this ALCS system.

YES
The program configuration data set, CDS1, is used by this ALCS system. Refer to CDNAME= for the construction of the data set name for CDS1.

CDS2={NO|YES}
Whether the communication configuration data set, CDS2, is used by this ALCS system.

NO
The communication configuration data set, CDS2, is not used by this ALCS system.

YES
The communication configuration data set, CDS2, is used by this ALCS system. Refer to CDNAME= for the construction of the data set name for CDS2.

DSNAME=(*name_list*)
Data set name prefix (or prefixes) for the database data sets. This prefix is for the data component name, not the VSAM cluster name (see "Defining and initializing the database and general file data sets manually" on page 461). If one name is specified, it is the prefix for all the database data sets. If a list of names is specified, the *n*th name in the list is the prefix for the size L*n* data sets. If the *n*th

item of the list is a pair of names enclosed in parentheses, the two names are used as the prefix for the size L*n* copy-1 data sets and the size L*n* copy-2 data sets respectively.

The default data set name prefix is DXC*iv*, where *i* is the ALCS system identifier and *v* is the ALCS system version. See "ALCS macro" on page 70.

Note: ALCS does not allow dynamic modification of this information.

DUPLEX={YES|NO}

YES
Database is duplicated. ALCS expects copy-1 and copy-2 data sets for each record size during data set allocation.

NO
Database is not duplicated.

ALCS issues messages (to the MVS operator) when only one copy of a data set is available. ALCS allows dynamic modification of this information.

Note: ALCS still generates two DSNs when DUPLEX=NO; you can bring the second copy online as part of the procedure to increase the size of the data set. "Long-term pool file records" on page 184 describes this procedure.

FARMIG={NONE|USER|USERCODE|VSERON|FARF3}
Format of file addresses in an ALCS/VSE or TPF database that is migrating to ALCS. If a database is migrating to ALCS, records can contain file addresses that are valid on the migrate-from (ALCS/VSE or TPF) database. These file addresses are not valid on the migrate-to (ALCS) database.

Note: ALCS does not allow dynamic modification of this information.

Specify how ALCS converts these file addresses as follows:

NONE
The database is not being migrated. ALCS does not attempt to convert invalid file addresses.

USER
The database is being migrated. If ALCS identifies an invalid file address, it calls a user-written file address migration subroutine. This subroutine determines whether the file address is valid on the migrate-from system; if it is, the subroutine determines the record class, type, and ordinal so that ALCS can access the record. "File address migration exit - USRFAR" on page 262 describes user-written file address migration subroutines.

USERCODE
The database being migrated contains a file address format that cannot coexist with the ALCS band format. For example, the file address contains the band in byte 0 and the ordinal in bytes 1-3.

ALCS will call the installation wide exit USRFAR for both encoding and decoding file addresses. "File address migration exit - USRFAR" on page 262 describes user-written file address encoding/decoding subroutines.

VSERON
An ALCS/VSE database is being migrated. If ALCS identifies an invalid file address, it determines whether the file address is an ALCS/VSE record ordinal number (RON) file address and, if it is, converts the file address to an ALCS file address. Fixed record type values in the RON must be the same as ALCS values.

FARF3
A TPF database is being migrated. If ALCS identifies an invalid file address, it determines whether the file address is a TPF File Address Reference Format 3 (FARF3) file address and, if it is, converts the file address to an ALCS file address. ALCS can handle FARF3 file addresses only if all FARF3 bands (bits 1 through 12) are equal to the fixed-file record type values[10] and if there are no 4KB pool file records. Fixed record type values in FARF3 must be the same as ALCS values.

[10] TPF calls the fixed-file record type value the FACE ID.

Note: ALCS does not convert TPF general file addresses or TPF symbolic ordinal number (SON) file addresses. If the TPF database contains these types of file addresses, or if the FARF3 addresses do not satisfy the criteria described above, specify FARMIG=USER; the user-written file address migration subroutine must include conversion routines for all the TPF file address formats, including FARF3, that exist on the migrate-from database.

MEMBER=*name*
The name of the configuration dependent table that describes the ALCS DASD configuration. The stage 2 link edit step uses this name for the DASD configuration table load module.

The default name is DXCCDD*iv*, where *i* and *v* are the identifier and version specified on the ALCS macro.

When you create an updated configuration-dependant table, you may want to use a different name for the new table. Do not change this parameter; instead specify the new name on the DBHIST macro.

PARTITION={0|*n*}
Reserves *n* percent of the total pool records for ALCS systems that use the database in read-only mode (as a **test database**).

Reserves the remaining pool records (100-*n*) for ALCS systems that use the database in read/write mode (not as a **test database**).

This partition applies to all long-term and short-term pools. If the partition is 0 percent, then the database is not partitioned.

POOLMIG=([0| *L1_non_dup*},{0|*L1_dup*},...)]
Information about the long-term pool file allocation of a TPF database that is migrating to ALCS. This information allows ALCS to import (ZDATA LOAD) and access data from the TPF database.

Omit this parameter if the database is not being migrated. Each pair of subparameters specifies the number of non-duplicated and duplicated pool file records on the migrating system:

L1_non_dup
The number of non-duplicated size L1 pool file records.

L1_dup
The number of duplicated size L1 pool file records.

L2_non_dup
The number of non-duplicated size L2 pool file records.

L2_dup
The number of duplicated size L2 pool file records.

Note: ALCS does not allow dynamic modification of this information.

SEGMENTS= ({*L1_segs*], [*L2_segs*],...)
Number of segments of addressability for each real-time database data set in the **initial** configuration. When you are migrating from ALCS Version 2 Release 1.1 or ALCS/MVS/XA, the initial configuration is the *migrate-from* database.

(*L1_segs*,*L2_segs*,...)
Number of segments for the L1, L2, ..., record sizes respectively.

For each record size, *x_segs* must be at least *n*/65 536 (rounded up) where *n* is the number of records that each existing data set contains.

ALCS calculates the appropriate number of segments if you:

- omit the entire parameter
- omit a subparameter
- specify 0 in any subparameter position.

You can allocate more segments of addressability to allow for expansion, but you can add addressability at any time using the DBSPACE macro.

Note: ALCS does not allow modification of this parameter. Use the DBSPACE macroinstruction to add more segments.

USRDTA macro

Use USRDTA to specify details of a database record type (fixed file or pool file). There is no restriction on the number of USRDTA macros in the stage 1 generation deck. See "ALCS record requirements" on page 174 for recommendations about the records that ALCS uses.

```
[label]  USRDTA TYPE=type
                          ,{NUMBER=number|PERCENT=percent}
                          ,USAGE={Z1|Z2|Z3}
                          [,BAND=(band,...)]
                          [,ACTION={ADD
                                   |BUILD
                                   |DELETE
                                   |PURGE
                                   |REPLACE
                                   |SPILL
                                   |UNDELETE
                                   }]
                          [,SIZE=Ln]
                          [,VFAOPT=({FI|DF|TI]
                                    [,NP|PR]
                                    [,NU|UP]
                                    [,HY|NH]
                                    )]
                          [,ID=(id[(qualif)],...)|,IDSYM=(id_symbol[(qualif)],...)]
```

Where:

label
Any valid assembler label.

TYPE=*type*
Database record type. For fixed-file records, this is the fixed file record type as defined in the DXCURID macro (for example, #WAARI). For pool file records this is the 4-character pool file record type:

L*n*ST
Size L*n* short-term pool

L*n*LT
Size L*n* long-term pool.

See "ALCS DXCURID user macro" on page 443 for reserved record types and "Generating a DASD configuration table" on page 173 for ALCS requirements.

Note: ALCS does not allow dynamic modification of this information.

NUMBER=*number*
Number of records of this record type.

When **ACTION**=ADD or SPILL, increase the allocation by *number*.

When **ACTION**=REPLACE, allocate *number*.

When **ACTION**=DELETE, decrease the allocation by *number*.

Notes:

1. ALCS allows dynamic modification of this information.
2. If NUMBER and PERCENT are omitted, a default value of 0 is assumed for NUMBER.
3. If NUMBER is 0 (specified or by default), only the record ID and file-address reference band information (if any) on that USRDTA macro is processed.

PERCENT=*percent*
Percentage increase in the allocation of this record type. This parameter is invalid on the first USRDTA macro for each record type, also when ACTION=REPLACE.

Note: ALCS allows dynamic modification of this information.

USAGE={Z1|Z2|Z3}
Usage zone for the record type. Usage zones can reduce average DASD access times, if highly used records are physically close to each other (that is, in the same or adjacent usage zones):

Z1
Zone 1. Put records at the beginning of the database data sets (low relative record numbers).

Z2
Zone 2. Put records in the middle of the database data sets.

Z3
Zone 3. Put records at the end of the database data sets (high relative record numbers).

Note: ALCS does not allow dynamic modification of this information. Omit USAGE when ACTION=SPILL is specified.

BAND=(*band,...*)
File address reference bands for this record type. This parameter is required for all record types. Specify one or more bands. Specify each band as 2, 4, or 6 hexadecimal digits. "Specifying the initial DASD configuration" on page 173 describes the ALCS band addresses and reserved bands.

Notes:

1. ALCS does not allow dynamic modification of the information already defined for a record type (except when deleting the record type completely).
2. ALCS allows additional band information to be added dynamically to provide additional addressability for existing record types.
3. You can increase the number of bands in a later generation by changing this parameter, However, it is preferable to add bands by adding additional USRDTA macroinstructions.

ACTION={ADD|BUILD|DELETE|PURGE|REPLACE|SPILL|UNDELETE}

ADD
Define a new record type, or increase the allocation of a record type that is already defined in this generation (L*n*STPOOL, for example).

Note: If NUMBER is 0 (specified or by default), then the ADD parameter defines new record IDs or file address reference bands.

BUILD
Stop using algorithm-based addressing for DASD access of this record type and start using table-based addressing.

DELETE
If NUMBER or PERCENT are specified and non-zero then ALCS marks the record for deletion.

If NUMBER or PERCENT are omitted or 0 then ALCS deletes the record ID or IDs.

For short-term pool, ALCS does not dispense records with ordinals in the range that is marked for deletion.

Notes:

1. *(STFFNOT)* DELETE, PURGE, and UNDELETE apply only to short-term pool or fixed file.
2. *(DELNOT)* DELETE can only be used after DBHIST DEFINE_MIGRATE_FROM or DBHIST DEFINE_NEW.

PURGE
Permanently remove the records that are marked for deletion from the database. Use PURGE only after a DELETE for the same record type. NUMBER and PERCENT should be omitted or 0.

See also, the note for the DELETE parameter.

REPLACE
Alter the definition of a record type that is already defined in this generation. Alter only the attributes that are coded. Also allocate the specified number of records.

REPLACE does not allow the record count to be changed after DBHIST DEFINE_MIGRATE_FROM or DBHIST DEFINE_NEW. It does allow the VFA options to be changed and it allows record IDs and/or address bands to be added. NUMBER should be omitted or should be 0 in this case.

SPILL
Allocate the records on a spill data set. SPILL is like ADD, except that the DASD generation puts the records in a different place on DASD.

Do not use SPILL:

- In the initial DASD generation
- In a DASD generation to define the reorganize-to data set before reorganizing the database.
- After DBHIST DISPENSE_TYPE2_LONG_TERM
- If records are added (or deleted) after:
  - DBHIST DEFINE_MIGRATE_FROM
  - DBHIST DEFINE_NEW.

Note: SPILL applies only to fixed file.

UNDELETE
Restore the records that are marked for deletion. For short-term pool, ALCS resumes dispensing records with ordinals in the range that was marked for deletion.

NUMBER and PERCENT should be omitted or 0.

See also, the notes for the DELETE parameter.

SIZE=L*n*
Record size. Specify L1 for size L1 records, L2 for size L2 records, and so on. This parameter only applies to fixed-file records.

Note: ALCS does not allow dynamic modification of this information.

VFAOPT=({FI|DF|TI[,NP|PR[,NU|UP[,HY|NH])
VFA options for the record type; this parameter only applies to fixed-file records. (VFA options for pool file records are not changeable. For short-term pool, they are DF. For long-term pool, they are FI, UP.) The options are:

FI
File immediate; that is, ALCS starts the I/O as soon as an application issues a FILE macro.

DF
Delayed file; that is, ALCS does not start the I/O until the buffer is required for another record.

TI
Time-initiated; that is, ALCS does not start the I/O until a time-initiated routine schedules I/O.

NP
Not permanently resident; that is, ALCS can re-use a VFA buffer containing the record if the record is unreferenced for long enough.

PR
Permanently resident; that is, once the record is placed in a VFA buffer, it remains there permanently.

NU
Not update mode; that is, ALCS does not ensure that the VFA buffer contains the old record contents before it services a file request.

UP
Update mode; that is, ALCS ensures that the VFA buffer contains the old record contents before it services a file request. This is required for backward logging.

HY
Use Hiperspace if available.

NH
Do not use Hiperspace.

It is possible to override some of these parameters with an installation-wide exit. See "VFA option set or option override exit - USRVFA" on page 301 for details.

ID=(*id*[(*qualif*)],...)
One or more 2-character alphanumeric or 4-character hexadecimal record identifiers (IDs) for this record type.

You can use the same ID for more than one record type. If you specify a record ID qualifier (*qualif* can be 0 through 9) then ALCS can distinguish between the same ID used for different records types. The default qualifier is 0.

For example:

```
USRDTA ACTION=ADD,TYPE=L1ST,BAND=1001,NUMBER=10000,ID=(AC11(1),AC12)
USRDTA ACTION=ADD,TYPE=L2ST,BAND=2001,NUMBER=10000,ID=(AC11(2),AC13)
```

IDSYM=(*id_symbol*[(*qualif*)],...)
One or more symbolic record IDs for this record type. A symbolic record ID is an assembler language symbol that ALCS resolves to a 2-byte record ID.

You can use the same symbolic ID for more than one record type. If you specify a record ID qualifier (*qualif* can be 0 through 9) then ALCS can distinguish between the same symbolic ID used for different records types. The default qualifier is 0.

For example:

```
USRDTA ACTION=ADD,TYPE=L3ST,BAND=3001,NUMBER=10000,IDSYM=#EDODD
USRDTA ACTION=ADD,TYPE=L3LT,BAND=3009,NUMBER=10000,IDSYM=#EDODD(1)
```

GFGEN macro

Use GFGEN to define a general file. Include one GFGEN macro for each general file.

```
[label]  GFGEN  NUMBER=num
                          [,ACTION={ADD|DELETE}]
                          [,BAND=(band,...)]
                          [,SIZE=Ln]
                          [,DSNAME=dsname]
                          [,ISTATUS={INACTIVE|ACTIVE}]
                          [,VFAOPT=({FI|DF|TI]
                                    [,NP|PR]
                                    [,NU|UP]
                                    [,HY|NH]
                                    [,NS|SH]
                                    )]
                          [,ID=(id[(qualif)],...)|,IDSYM=(id_symbol[(qualif)],...)]
```

Where:

label
Any valid assembler label.

NUMBER=*num*
General file number. A decimal number in the range 0 through 255.

ACTION={ADD|DELETE}
Add or delete a general file definition. DELETE can only be used after DBHIST DEFINE_MIGRATE_FROM or DBHIST DEFINE_NEW.

BAND=(*band,...*)
File address reference bands for records in this general file. Specify one or more bands. Specify each band as 2, 4, or 6 hexadecimal digits. "Specifying the initial DASD configuration" on page 173 describes the ALCS band addresses and reserved bands.

SIZE=L*n*
Record size for the general file. Specify L1 for size L1 records, L2 for size L2 records, and so on. Omit SIZE on the GFGEN macro that defines general file 0. General file 0 is reserved for Recoup and must have a size of L2. (See Chapter 9, "Long-term pool space recovery - Recoup," on page 480.)

DSNAME=*dsname*
The data set name for the general file. This is the data component name, not the VSAM cluster name. (See "Defining and initializing the database and general file data sets manually" on page 461.) The default name is DXC*iv*.G*nnn*, where *i* and *v* are the identifier and version as specified on the ALCS macro, and *nnn* is the decimal general file number, expanded to 3 digits with leading zeros if necessary.

If you want to use this as a general data set (that is, if your application uses GDSNC and GDSRC macros to get file addresses), the data set name must not exceed 16 characters.

ISTATUS={INACTIVE|ACTIVE}
The status of the general file when ALCS restarts:

INACTIVE
Allocate the general file to ALCS only when requested with the ALCS ZDASD command.

ACTIVE
Allocate the general file during ALCS restart.

VFAOPT=({FI|DF|TI[,NP|PR[,NU|UP[,HY|NH[,NS|SH])
VFA options for records in this general file. The options are:

FI
File immediate; that is, ALCS starts the I/O immediately an application issues a FIND or FILE macro.

DF
Delayed file; that is, ALCS does not start the I/O until the buffer is required for another record.

TI
Time-initiated; that is, ALCS does not start the I/O until a time-initiated routine schedules I/O.

NP
Not permanently resident; that is ALCS can re-use a VFA buffer containing the record if the record is unreferenced long enough.

PR
Permanently resident; that is, once the record is placed in a VFA buffer, it remains there permanently.

NU
Not update mode; that is ALCS does not ensure that the VFA buffer contains the old record contents before it services the request.

UP
Update mode; that is, ALCS ensures that the VFA buffer contains the old record contents before it services a file request. This is required for backward logging.

HY
Use Hiperspace if available.

NH
Do not use Hiperspace

NS
Not shared; that is, ALCS does not share access to the data set with other ALCSs or other offline programs. If you specify (or default) NS ALCS assumes it can buffer records in VFA. If an application program reads a record which is in a VFA buffer, ALCS does **not** read the record from the data set.

SH
Shared; that is, ALCS shares the data set with other ALCSs or offline programs. If an application program reads a record, ALCS **always** reads the record from the data set even if there is a copy of the record in a VFA buffer. SH implies file immediate (FI).

Note: ALCS does not provide serialization services to control access to shared general files. Your application must implement protocols to prevent or serialize parallel updating of shared data sets by different systems.

ID=(*id*[(*qualif*)],...)
One or more 2-character alphanumeric or 4-character hexadecimal record identifiers (IDs) for the record types in this general file.

You can use the same ID for more than one record type. If you specify a record ID qualifier (*qualif* can be 0 through 9) then ALCS can distinguish between the same ID used for different records types. The default qualifier is 0.

For example:

```
GFGEN NUMBER=3,BAND=5003,SIZE=L1,ID=(GF(0),G3)
GFGEN NUMBER=4,BAND=5004,SIZE=L1,ID=(GF(1),G4)
```

IDSYM=(*id_symbol*[(*qualif*)],...)
One or more symbolic record IDs for the record types in this general file. A symbolic record ID is an assembler language symbol that ALCS resolves to a 2-byte record ID.

You can use the same symbolic ID for more than one record type. If you specify a record ID qualifier (*qualif* can be 0 through 9) then ALCS can distinguish between the same symbolic ID used for different records types. The default qualifier is 0.

For example:

```
GFGEN NUMBER=3,BAND=5003,SIZE=L1,IDSYM=(GFTEST(1))
GFGEN NUMBER=4,BAND=5004,SIZE=L1,IDSYM=(GFTEST(2))
```

DBHIST macro

ALCS provides the DBHIST macro to:

- Generate an audit trail of database changes
- Control the migration from predecessor ALCS systems to ALCS Version 2 Release 4.1

The stage 1 generation that defines the initial configuration must contain one (and only one) DBHIST macroinstruction.

Each time you create a new DASD configuration table, you must add one or more DBHIST macroinstructions to the stage 1 generation input deck (followed by additional USRDTA, GFGEN or DBSPACE macros).

Some changes consist of only adding a DBHIST macro. For example, during migration you use the DBHIST macro (with no updates) to initiate the next migration state.

The DBHIST macroinstructions must appear in the stage 1 input deck (with associated updates) in the same order that the changes are applied. The ZDASD LOAD command compares the currently-loaded DASD configuration with the one you are trying to load. The sequence of changes and DBHIST macros must be identical in both configurations, up to but excluding any new changes in the configuration being loaded. This ensures that updates are applied in the correct sequence.

The generic form of the DBHIST macro is:

```
[label] DBHIST event,TEXT=text_string
```

You specify one event with each DBHIST macro. The TEXT= parameter allows an audit trail to be added to each event.

```
[label] DBHIST {DEFINE_MIGRATE_FROM|
                                DEFINE_NEW|
                                UPDATE|
                                DISPENSE_TYPE2_SHORT_TERM|
                                DISPENSE_TYPE1_SHORT_TERM|
                                DISPENSE_TYPE2_LONG_TERM|
                                RESTRICT_SHORT_TERM|
                                UNRESTRICT_SHORT_TERM|
                                BUILD_DIRECTORIES}
                                [,DUPLEX={YES|NO}]
                                [,MEMBER=name]
                                [,PARTITION=n]
                                ,TEXT=text_string
```

Where:

label
Any valid assembler label

DEFINE_MIGRATE_FROM
Specify this form of DBHIST when you define an existing database (migrating from ALCS/MVS/XA or ALCS Version 2 Release 1.1). Include the DBHIST at the end of the initial stage 1 database generation.

Note: Do not specify this form in any subsequent DBHIST macroinstructions (when updating the configuration). This allows fallback to the migrate-from system.

DEFINE_NEW
Specify this form of DBHIST when you define a new database (**not** migrating from a previous release of ALCS). Include the DBHIST at the end of the initial stage 1 database generation.

Note: Do not specify this form in any subsequent DBHIST macroinstructions (when updating the configuration).

UPDATE
Specify this form of DBHIST when no other form is required (or appropriate). This is the only form of DBHIST you should use when your system is fully migrated to ALCS Version 2 Release 4.1.

DISPENSE_TYPE2_SHORT_TERM
Specify this form of DBHIST when you wish to start dispensing short-term pool records using the new short-term pool mechanism introduced in ALCS V2 Release 1.3.

Before you specify this form of DBHIST, use the RESTRICT_SHORT_TERM form to ensure that the short-term pool space is migrated in an orderly way. Figure 47 on page 197 shows how you control the short-term pool dispense space between each migration state.

You can make other changes to the DASD configuration when you add this DBHIST macroinstruction.

DISPENSE_TYPE1_SHORT_TERM
Specify this in the DBHIST macroinstruction that you add to your stage 1 database generation to fall back from using the new short-term pool dispense mechanism to using a predecessor one.

DBHIST DISPENSE_TYPE1_SHORT_TERM is allowed only after a previous DBHIST DISPENSE_TYPE2_SHORT_TERM.

Before you specify this form of DBHIST, use the RESTRICT_SHORT_TERM form to ensure that the short-term pool space is migrated in an orderly way. Figure 47 on page 197 shows how you control the short-term pool dispense space between each migration state.

You can use DBHIST DISPENSE_TYPE2_SHORT_TERM to switch to the new short-term pool space.

You can make other changes to the DASD configuration when you add this DBHIST macroinstruction.

DISPENSE_TYPE2_LONG_TERM
Specify this in the DBHIST macroinstruction that you add to your stage 1 database generation when you wish to start dispensing long-term pool records with allocatable-pool file addresses.

Note that including a DBHIST DISPENSE_TYPE2_LONG_TERM macroinstruction indicates that it is no longer possible to fall back to the predecessor ALCS/MVS/XA or ALCS V2 Release 1.1 system. This is because the file addresses dispensed by the new dispense mechanism will not be "understood" by the predecessor system. ALCS V2 Release 1.3 and later releases will however continue to understand "type 1" long-term pool file addresses so that application programs can continue to access (and eventually release) file addresses that are already in use.

You can make other changes to the DASD configuration when you add this DBHIST macroinstruction.

RESTRICT_SHORT_TERM
Specify this to divide the short-term pool space into two subsets. This allows you to switch between the old (type 1) and new (type 2) short-term pool dispensing methods by using records from one subset (to dispense and release) while the other subset is allowed to become empty (as records are released but not redispensed).

Figure 47 on page 197 shows how the DBHIST DISPENSE_*xxx*_SHORT_TERM parameter affects the use of the subsets.

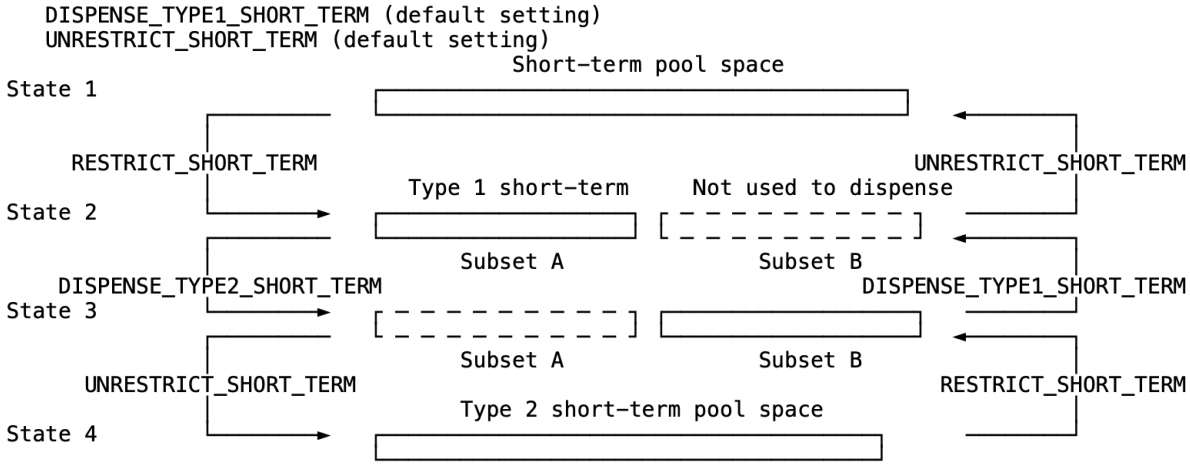


Figure 47. Restricting short-term pool space

"Migrating ALCS short-term pool" on page 217 explains where to use this parameter.

UNRESTRICT_SHORT_TERM
Specify this to allow the whole short-term pool space to be used to dispense records.

Figure 47 on page 197 shows how the DBHIST DISPENSE_*xxx*_SHORT_TERM parameter controls which method is used to dispense the short-term pool records.

"Migrating ALCS short-term pool" on page 217 explains where to use this parameter

BUILD_DIRECTORIES
Stop using algorithm-based addressing for DASD access of all record types and start using table-based addressing.

DUPLEX={YES|NO}
Use this parameter only if you want to change the duplex setting.

YES
Database is duplicated. ALCS expects copy-1 and copy-2 data sets for each record size during data set allocation. ALCS issues messages (to the MVS operator) when only one copy of a data set is available.

NO
Database is not duplicated.

Note: ALCS still generates two DSNs when DUPLEX=NO; you can bring the second copy online as part of the procedure to increase the size of the data set.

"Long-term pool file records" on page 184 describes this procedure.

MEMBER=*name*
Use this parameter only if you want to change the member name. The name of the configuration dependent table that describes the ALCS DASD configuration.

The stage 2 link edit step uses this name for the DASD configuration table load module.

You can use different names (with ZDASD LOAD) to load different generations of tables, but you must not change the name in the JCL used to start ALCS.

PARTITION=*n*
Use this parameter only if you want to change the partition value. Reserves *n* percent of the total pool records for ALCS systems that use the database in read-only mode (as a **test database**).

Reserves the remaining pool records (100-*n*) for ALCS systems that use the database in read/write mode (not as a **test database**).

This partition applies to all long-term and short-term pools. If the partition is 0 percent, then the database is not partitioned.

TEXT=*text_string*
Free-format text up to 80 characters. The standard assembler rules for quoted strings applies to the string. Use this parameter for audit purposes such as:

- Change requestor, (the name of the person or department)
- Change authority (the name of the person or department)
- Other audit comments

DBSPACE macro

The DBSPACE macro allows a user to increase the total number of addressable records on the real-time database.

DBSPACE can extend:

- The number of data sets
- The number of logically-addressable records for each data set

"Adding a new DASD record size" on page 185 describes the complete procedure to add a new record size.

Note: The number of logically-addressable records should be greater than the number of records that actually (physically) exist, for (at least) the following reasons:

- To allow for expansion.

  The user can dynamically increase the size or number (or both) of data sets up to and including the amount of logically-addressable records. To change the size:

1. Use ZDASD VARY,OFF to de-activate the existing data set.
2. Use ZDASD VARY,ON to activate the new larger edition of the same data set.

- Granularity considerations.

  Addressability is allocated in blocks of 65 536 records, which may not exactly match the number of records required. The granularity may be coarser than this because ALCS enforces the same addressability (but not the same number of records) for all data sets of a given record size.

```
[label] DBSPACE SIZE=Ln
                        [,ACTION=ADD]
                        [,BAND=(band,...)]
                        [,VOLUMES=volumes]
                        [,SEGMENTS=segments]
```

---

label
Any valid assembler label.

SIZE=L*n*
The record size to increase.

ACTION=ADD
Increase the logical addressability for the specified record size.

BAND=(*band,...*)
The allocatable-pool file-address reference band (or bands) for this record size. These bands are in addition to any bands you specify in the DBGEN macro or previous DBSPACE macro calls. "Specifying the initial DASD configuration" on page 173 describes the ALCS band addresses and reserved bands.

VOLUMES=*volumes*
The number (in decimal) of additional data sets to make addressable for the specified record size. Each data set is allocated the same amount of addressability as the existing data sets for this record size.

Note: The actual size (number of records) in the data sets can be less than the number of logically-addressable records.

SEGMENTS=*segments*
The number (in decimal) of additional segments (blocks of 65 536 records) to make addressable for the specified record size. Each data set is allocated the same amount of additional addressability.

Note: The actual size (number of records) in the data sets can be less than the amount of logically-addressable records.

ALCSGEN macro

Use ALCSGEN to specify the ALCS components generated by the generation stage 2 job stream.

Note for ISPF panels:

If you use ISPF panels to generate your system configuration, **do not** code this macroinstruction. Instead, update your ISPF system definitions with any changes.

```
[label]  ALCSGEN TYPE=(option,...)
                        [,STAGE2={YES|SMP|NO|NOJCL}]
                        [,PRINT={NOGEN|OFF|ON|GEN}]
```

Where:

label
Any valid assembler label.

TYPE=(*option*,...)
Components of the ALCS system that the stage 2 job stream generates. The following are valid parameters for TYPE:

COMS
Generate the communication configuration table. When **STAGE2**=YES or STAGE2=SMP, punch the stage 2 job stream to:

1. Assemble the communication configuration module
2. Link-edit the communication configuration module.

SEQ
Generate the sequential file configuration table. When **STAGE2**=YES or STAGE2=SMP, punch the stage 2 job stream to:

1. Assemble the sequential file configuration module
2. Link-edit the sequential file configuration module.

SYS
Generate the system configuration tables. When **STAGE2**=YES or STAGE2=SMP, punch the stage 2 job stream to:

1. Assemble the system configuration tables module
2. Link-edit the system configuration tables module.

DASD
Generate the DASD configuration table. When **STAGE2**=YES or STAGE2=SMP, punch the stage 2 job stream to:

1. Assemble the DASD configuration module
2. Link-edit the DASD configuration module.

ALL
Equivalent to specifying **COMS,SEQ,SYS,DASD**.

STAGE2={YES|SMP|NO|NOJCL}

YES
Punch a stage 2 job stream unless there are errors in the stage 1 macro coding. The generation uses catalogued procedures to execute the assembler and linkage editor programs. You can specify the names of these procedures on the **PROC**= parameter of the ALCS macro (for more information see "ALCS macro" on page 70).

SMP
Punch a stage 2 job stream unless there are errors in the stage 1 macro coding. The generation ignores the **PROC**= parameter of the ALCS macro. The generation does not use cataloged procedures to execute the assembler and linkage editor programs.

NO
Do not punch a stage 2 job stream.

NOJCL
Punch a stage 2 generation deck unless there are errors in the stage 1 macro coding. Do not punch any job control (JCL) statements.

PRINT={NOGEN|OFF|ON|GEN}
Print option for the assembler PRINT command that the stage 1 assembly job generates immediately before punching the stage 2 job. NOGEN is recommended, to avoid duplicating the output of stage 2.

Loading communication configuration load modules

You can load a communication configuration load module in either of the following two ways.

- At initial-load time during the system start

- Later, online (with the ZACOM command).

For the initial load, create a communication configuration load list that contains a list of the communication configuration load modules that are required by your ALCS system. (see "Updating the communication configuration load list" on page 205). Specify the name of the communication configuration load list in the PARM field on the EXEC statement that starts the ALCS job. At restart, ALCS loads all the communication configuration load modules in the communication configuration load list. If your ALCS system uses the optional communication configuration data set (CDS2), during the initial startup of your ALCS system the communication configuration load list will be copied to CDS2. Subsequent restarts of ALCS will load the communication configuration load modules that are referenced in the communication configuration load list on CDS2.

Online loading of communication configuration load modules

After the initial load of the communication configuration load modules on your ALCS system, you can use the ALCS communication generation to create *update* communication configuration load modules. Use the ZACOM command to load the update communication configuration load modules on your ALCS system. (See *ALCS Operation and Maintenance*).

The update communication configuration load module should reflect changes that are required in your existing ALCS communication configuration. Define the network changes in ALCS communication generation COMGEN, COMDFLT and COMDEF macroinstructions. Create an update communication configuration stage 1 generation deck, and run the generation to create the update communication configuration load module.

Before you apply the communication network changes to your ALCS system (by loading the update 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. (See the *ALCS Operation and Maintenance*).

When you use the ZACOM command to load the update communication configuration load module, it 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 update load module.

When you are confident that ALCS functions correctly with the changed configuration, take one of the following actions to ensure that the update communication configuration load module is reloaded automatically during a restart of the ALCS system.

- If your ALCS system is using the communication configuration data set (CDS2) to manage the communication configuration load list (which contains the names of the communication configuration load modules to be reloaded during the next ALCS restart), use the ZACOM CONFIRM command to request ALCS to reload the update communication configuration load module during the next system restart. For information on how you create and manage CDS2, see "Using the Communication Configuration Data Set" on page 201.
- The communication configuration data set is optional, therefore if your system does not use CDS2, you should do the following. Add the name of the update communication configuration load module to the end of the load module list in the communication configuration load list (that is referenced in the PARM field on the EXEC statement that starts the ALCS job). Do this before the next restart of your ALCS system. "Updating the communication configuration load list" on page 205 describes how you update the communication configuration load list.

Using the Communication Configuration Data Set

You can use the communication configuration data set (CDS2) to manage the communication configuration load list that specifies the communication configuration load modules to be loaded during the next ALCS system restart. CDS2 contains two communication configuration load lists, a **current** and **alternate** load list. The current load list normally contains the list of load modules to be loaded during the next ALCS restart. The alternate load list normally contains a load module list that can be used if the current load list is unsatisfactory (it provides an alternative load module list). The CDS1 data set is

comprised of two *slots*. One slot contains the current load list and the other slot contains the alternate load list. They are called **slot A** and **slot B**.

When the ZACOM command is used to load *update* communication configuration load modules, ALCS adds the name of the load module to the current communication configuration load list on CDS2. You can use the ZACOM command to confirm the load module. This ensures that ALCS will reload the load module during the next system restart.

The ZACOM command can be used for loading a new base communication configuration load list onto CDS2. From time to time, you can merge all the communication generation decks (stage 1 input for the base and update communication configuration load modules) and create a new base load module. This consolidation of the communication generation decks could result in the creation of a new base load module that defines part of your network configuration, and one or more update load modules for the remainder of the network configuration. Create a new communication configuration load list that lists the new base and update load modules, and use the ZACOM command to load and confirm the new communication configuration load list. When you have confirmed the new communication configuration load list, ALCS will use it during the next system restart.

To use the communication configuration data set (CDS2), update the DBGEN macro parameters in the ALCS data base generation. Code the DBGEN CDS2=YES parameter to request ALCS to use the communication configuration load list on CDS2 during ALCS restart. The communication configuration load list referenced by the PARM field on the EXEC statement is loaded onto CDS2 on the initial startup of your ALCS system, and is then bypassed on subsequent ALCS restarts. During this initial startup of your system, ALCS formats the new CDS2 data set, loads the communication configuration load list onto it, and sets the status of the load list and each load module in the load list as *committed*.

- **Using CDS2 for loading update communication configuration load modules**

  When update communication configuration load modules are loaded online, additional functions are performed if your ALCS system is using CDS2. The following describes the additional functions performed by ZACOM when CDS2 is used.

  – *Loading a communication configuration load module*
  When a load module has been successfully loaded by the ZACOM LOAD command, ALCS adds the name of the load module to the load list in the current communication configuration load list on CDS2. The load list entry for that load module is stamped with the date and time of the load and marked as *loaded*. On the next ALCS restart, this load module will not be reloaded.

  – *Confirming a communication configuration load module*
  When a load module that has been loaded is safe to preserve across an ALCS restart, it can be confirmed by the ZACOM CONFIRM command. The ZACOM command accesses the load list entry for that load module on CDS2, stamps the entry with the date and time of the confirm and marks it as *confirmed*. On the next ALCS restart, this load module will be reloaded. Communication configuration load modules must be confirmed in the same order in which they were loaded.

  – *Backing out a communication configuration load module*
  If a load module is to be backed out, this can be performed by using the ZACOM BACKOUT command. The ZACOM command accesses the load list entry for that load module on CDS2, stamps the entry with the date and time of the back out, and marks it as *backed out*. On the next ALCS restart, this load module will not be reloaded.

  Note: The ZACOM BACKOUT command has no affect on the communication definitions that ALCS is already using (it takes affect after the next ALCS restart).

- **Loading communication configuration load lists on CDS2**

  From time to time, you can merge all the communication generation decks (stage 1 input for the base and update communication configuration load modules) and create a new base load module (and new update load modules, if required). This consolidation of the communication configuration load modules, which includes the creation of a new communication configuration load list, is performed in two stages.

  The first stage is to run the ALCS communication generation, create new communication configuration load modules, and build a new communication configuration load list. See “Consolidating the communication configuration” on page 204 for details on how to do this.

The second stage is to load the new communication configuration load list onto CDS2, confirm it and restart the ALCS system. During the ALCS restart, the load modules referenced in the new communication configuration load list will be loaded. The ZACOM command is used for loading the new communication configuration load list onto CDS2. The following describes the ZACOM commands that you use for loading and activating the new communication configuration load list onto CDS2.

- *Loading a communication configuration load list onto CDS2*
  Use the ZACOM LOAD LIST command to load the new communication configuration load list onto CDS2. This communication configuration load list becomes the alternate load list on CDS2 and is marked as *loaded*. It will not be used by ALCS until it has been confirmed. If a restart of the ALCS system occurs, the communication configuration load modules referenced by the current load list are loaded.
- *Confirming a communication configuration load list on CDS2*
  Use the ZACOM CONFIRM LIST command to confirm the new communication configuration load list that you have loaded onto CDS2. This alternate load list is marked as *confirmed* on CDS2 and will be used by ALCS for loading the communication configuration load modules during the next system restart.
- *Backing out a communication configuration load list on CDS2*
  Use the ZACOM BACKOUT LIST command to backout the new communication configuration load list on CDS2. This alternate load list is marked as *backed out* on CDS2. You can back out a load list after it has been loaded or after it has been confirmed. If a restart of the ALCS system occurs, the communication configuration load modules referenced by the current load list are loaded.
- *Committing a communication configuration load list on CDS2*
  Use the ZACOM COMMIT LIST command to commit the new communication configuration load list on CDS2. The load list must have been previously confirmed by the ZACOM CONFIRM LIST command. You can not commit a communication configuration load list until after you have verified the contents of the load list via a restart of the ALCS system (the restart will load the communication configuration load modules referenced in the load list). The communication configuration load list is marked as *committed* on CDS2. You can not back out a load list after it has been committed. The new load list (which was the alternate list) now becomes the current list, and the load list that was previously the current list becomes the alternate list (and the contents of that load list are cleared).

- **Additional option for backing out a communication configuration load list on CDS2**

  When a new communication configuration load list has been confirmed on CDS2, you must schedule a planned restart of the ALCS system to verify the new communication configuration load list and the load modules that are included in the load list. After the restart, if you are not experiencing any communication network problems on your ALCS system, you can *commit* the new communication configuration load list. If there are communication network problems, you may decide to fallback to the previous communication configuration load list on CDS2. To do this, enter the ZACOM BACKOUT LIST command to backout the new load list, and schedule another restart of the ALCS system.

  When you have restarted the ALCS system with a new communication configuration load list, you could experience very serious problems which necessitate an immediate fallback to the previous communication configuration load list on CDS2. In normal circumstances, you could do this by entering the ZACOM BACKOUT LIST command and immediately restarting the ALCS system. However, the ALCS system could though fail to logon (or fail to accept logons) from any 3270 terminals. The system might reach IDLE system state, but not accept any ZACOM commands. In this case, you can immediately restart the ALCS system with the previous communication configuration load list on CDS2, without entering the ZACOM BACKOUT LIST command. Request ALCS to perform a *back out* of the new communication configuration load list during system restart by placing a *minus sign* in front of the communication configuration load list name in the PARM field on the EXEC statement that starts the ALCS job (for example, the load list name would be in the format **-listname**). During the restart, ALCS will not only perform the back out but also fallback to the previous communication configuration load list on CDS2 for loading the communication configuration load modules.

- **Loading a communication configuration load module prior to committing a load list**

  When a new communication configuration load list has been loaded and confirmed, a period of time may elapse before the load list is committed. For example, you may load a new communication

configuration load list and schedule the restart of the ALCS system for three days later. After the system restart, you may decide to immediately commit the load list, or you may decide to delay the commit. When a new communication configuration load list has been loaded but is between the confirmed and committed stages, you may need to load additional communication configuration load modules. You can do this. When you load additional communication configuration load modules, if CDS2 contains a new communication configuration load list that has been loaded and confirmed (but not committed), ALCS will include the name of the load module in both the current and alternate load lists on CDS2. When you confirm or back out an update communication configuration load module, ALCS will update the load module status in both the current and alternate load lists on CDS2. This will continue until you either commit or backout the new communication configuration load list. If you decide to backout the new communication configuration load list, you must first backout any additional communication configuration load modules that have been loaded since the load list was confirmed.

If this requirement to backout additional communication configuration load modules does not conform to your installation standards, you can use the installation-wide monitor exit USRCDSB to force a backout of the new communication configuration load list even if it contains additional communication configuration load modules.

Consolidating the communication configuration

From time to time, you can merge all the communication generation decks (stage 1 input for the base and update load modules) and create a new base load module. This consolidation of the communication generation decks could result in the creation of a new base load module that defines part of your network configuration, and one or more update load modules for the remainder of the network configuration. When you merge the communication generation decks, to maintain the integrity of the communication database, **each resource must keep the same CRI and the same communication resource ordinal number that it had before the merge.** To achieve this, you should do one of the following:

- Merge the contents of the generation decks so that the order of the resource definitions does not change. Figure 48 on page 205 shows how to do this.
- Update the communication generation decks to explicitly define the CRIs and the communication resource ordinals. The `ICRI=` and `IORD=` parameters are provided on the `COMDFLT` macro for defining a range of CRIs and ordinals.

  The `CRI=` and `ORD=` parameters are provided on the `COMDEF` macro for defining the CRI and ordinal number for a specific resource.

When you have created the new base communication configuration load module (and any new update load modules), build a new communication configuration load list that includes the names of the new base and update load modules. See “Updating the communication configuration load list” on page 205.

If you have created both base and update communication configuration load modules, run the ALCS communication report file generator (DXCCOMOL) to validate the communication configuration definitions in the base and update load modules. Provide the name of your new communication configuration load list as input to DXCCOMOL. See the ALCS Operation and Maintenance for details on running the ALCS communication report file generator.

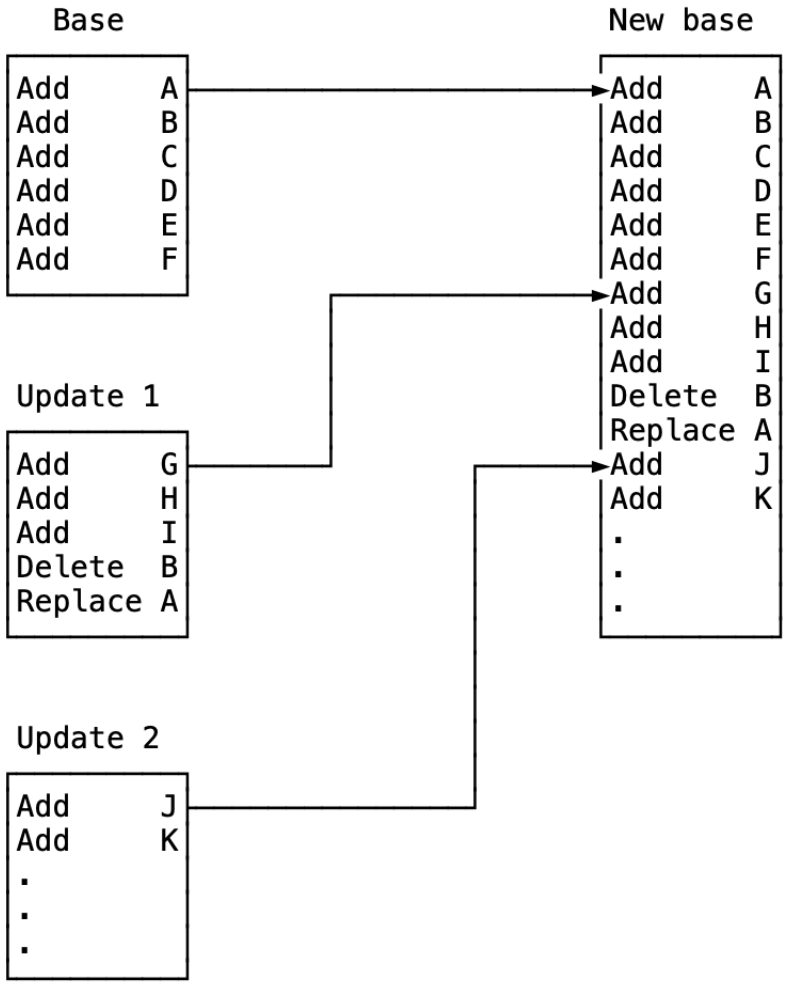


Figure 48. Merging communication generation decks

Updating the communication configuration load list

The communication configuration load list contains a list of the communication configuration load modules that are to be loaded during the next restart of the ALCS system. Figure 49 on page 206 shows an example of job control statements to build a communication configuration load list. The name of the communication configuration load list is specified on the SYSLMOD DD statement of the link-edit step.

The name of the initial (base) communication configuration load module should always be the first load module in the list. To load update communication configuration load modules, update the load list to include the name of each update communication configuration load module. Any number of update load modules can be included in the list.

```
//*--------------------------------------------------------------------*
//*        ALCS V2  COMMUNICATION CONFIGURATION LOAD LIST GENERATION  *
//*--------------------------------------------------------------------*
//*        PARAMETERS YOU MAY WISH TO CHANGE:                          *
//*                                                                    *
//*           ALCS DATASET NAMES THAT START WITH THE RECOMMENDED       *
//*           HIGHER LEVEL QUALIFIERS DXC.V2R4M1                       *
//*                                                                    *
//*--------------------------------------------------------------------*
//*
//*
//COMSLASM EXEC PGM=ASMA90,PARM='NODECK,OBJECT,LIST,XREF(SHORT),RENT'
//SYSUT1    DD DSNAME=&&SYSUT1,UNIT=VIO,SPACE=(1700,(1200,100))
//SYSLIN    DD DSNAME=&&SYSLIN,UNIT=VIO,SPACE=(CYL,(5,5,0)),
//             DCB=(BLKSIZE=400),DISP=(,PASS)
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
         TITLE ' *** LIST OF COMMUNICATION CONFIGURATION TABLES ***'
         SPACE 1
*====================================================================*
*        THIS IS A LIST OF THE ALCS COMMUNICATION CONFIGURATION LOAD  *
*        MODULES PRODUCED BY ALCS COMMUNICATION GENERATION STAGE TWO  *
*        THE NAME OF THE LOAD MODULE GENERATED FROM THIS LIST IS:     *
*        A. SPECIFIED IN THE PARM FIELD OF THE JOB THAT RUNS THE ALCS *
*           MONITOR                                                  *
*        B. OR, LOADED ONLINE USING THE ZACOM LOAD LIST COMMAND       *
*====================================================================*
         SPACE 1
COMSLIST CSECT ,
COMSLIST AMODE 31
COMSLIST RMODE ANY
         SPACE 1
         DC    CL8'BASELIST'
         DC    CL8'CRASLIST'
         DC    CL8'UPD1LIST'
         DC    CL8'UPD2LIST'
         DC    CL8'UPD3LIST'
         DC    CL8'UPD4LIST'
         SPACE 1
         END   ,
/*
//*
//*  LINK COMMUNICATION CONFIGURATION LOAD LIST
//*
//COMSLLNK EXEC PGM=HEWL,
//             PARM='XREF,LIST,LET,NCAL,RMODE=ANY,AMODE=31,AC=1'
//SYSLIN    DD DSNAME=&&SYSLIN,DISP=(OLD,DELETE)
//SYSUT1    DD DSNAME=&&SYSUT1,UNIT=VIO,SPACE=(1700,(600,100))
//SYSLMOD   DD DSNAME=DXC.V2R4M1.DXCLMD4(COMSLIST),DISP=SHR
//SYSPRINT  DD SYSOUT=*
//
```

Figure 49. Example JCL to assemble and link-edit the communication configuration load list

Updating and generating a load set

The load set defines the load modules (for system-wide programs) to be loaded with help of the ZPCTL LOAD SET command.

Figure 50 on page 207 shows an example how to build a load set.

```
//*-------------------------------------------------------------*
//* ALCS V2 LOAD SET GENERATION                                 *
//*-------------------------------------------------------------*
//* PARAMETERS YOU MAY WISH TO CHANGE:                          *
//*                                                             *
//* ALCS DATASET NAMES THAT START WITH THE RECOMMENDED          *
//* HIGHER LEVEL QUALIFIERS DXC.V2R4M1                          *
//*                                                             *
//*-------------------------------------------------------------*
//* IN THIS EXAMPLE THE LOAD SET CONTAINS OF TWO LOAD MODULES: *
//* LOADMOD1 AND LOADMOD2.                                      *
//* THE LOADSET NAME IS LOADSET1. THE ADDITIONAL LOAD SET       *
//* AUDIT INFORMATION IS 'MY_TEXT'                              *
//*-------------------------------------------------------------*
//PGLSTASM EXEC PGM=ASMA90,PARM='NODECK,OBJECT,LIST,XREF(SHORT),RENT'
//SYSLIB DD DSNAME=DXC.V2R4M1.DXCMAC1,DISP=SHR
//       DD DSNAME=DXC.V2R4M1.DXCMAC3,DISP=SHR
//       DD DSNAME=DXC.V2R4M1.DXCMAC4,DISP=SHR
//       DD DSNAME=SYS1.MACLIB,DISP=SHR
//SYSUT1 DD DSNAME=&&SYSUT1,UNIT=VIO,SPACE=(1700,(1200,100))
//SYSLIN DD DSNAME=&&SYSLIN(LOADSET1),UNIT=VIO,
// SPACE=(CYL,(5,5,2)),DCB=(BLKSIZE=400),DISP=(,PASS)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
           TITLE ' *** LOAD SET ***'
           SPACE 1
           PRINT NOGEN
LOADSET1   CSECT ,
           DXCHOST
           CP1DT ACTION=GEN,                                    -
                 NAME=LOADSET1,        LOAD SET NAME            -
                 AUDIT='MY_TEXT',      LOAD SET INFORMATION     -
                 MODLST=(MLIST,MLISTE) LIST OF LOAD MODULES
MLIST      DC    0D'0'
           DC    CL8'LOADMOD1'         1ST LOAD MODULE
           DC    CL8'LOADMOD2'         2ND LOAD MODULE
           ..
           ..
MLISTE     EQU   *
           LTORG ,
           SPACE 1
           END ,
//*
//* LINK EDIT
//*
//PGLSTLK EXEC PGM=HEWL,
// PARM='XREF,LIST,LET,NCAL,RMODE=ANY,AMODE=31,AC=1'
//SYSLIN DD DSNAME=&&SYSLIN(LOADSET1),DISP=(OLD,DELETE)
//SYSUT1 DD DSNAME=&&SYSUT1,UNIT=VIO,SPACE=(1700,(600,100))
//SYSLMOD DD DSNAME=DXC.V2R4M1.DXCLMD4(LOADSET1),DISP=SHR
//SYSPRINT DD SYSOUT=*
```

Figure 50. An example on how to build a load set.

NAME=
The load set name.

AUDIT=
A free-format text up to 80 characters. The standard assembler rules for quoted strings applies to this text.

MODLST=
The list of the load modules where 1000 is the maximum number of load modules.

Loading program and installation-wide monitor exit load modules

There are 2 types of load module that you can create and load on the ALCS system.

- Program load modules.

  These contain the application programs used on the ALCS system, plus any ECB-controlled installation-wide exit programs.
- Installation-wide monitor exit load modules.

These contain the installation-wide monitor exits.

You can load either type of module:

- At initial-load time during the system start
- Later, online (with the ZPCTL command).

For the initial load, create a program configuration table that contains a list of the program load modules and installation-wide monitor exit load modules that are required for your ALCS application. (See "Updating the program configuration table" on page 213). The program configuration table used for the initial load is called the base program configuration table. Specify the name of the program configuration table in the PARM field on the EXEC statement that starts the ALCS job. At restart, ALCS loads all the program load modules and installation-wide monitor exit load modules in the program configuration table. If your ALCS system uses the optional program configuration data set (CDS1), the program configuration table will be copied to CDS1 during the initial startup of your ALCS system. Subsequent restarts of ALCS will load the program load modules and installation-wide monitor exit load modules that are referenced in the program configuration table on CDS1.

Loading program load modules

After the initial load of the program load modules on your ALCS system, you can load additional program load modules online (for new or modified programs) using the ZPCTL command (see *ALCS Operation and Maintenance*).

Program load modules loaded online with the ZPCTL command may not be automatically reloaded during the next restart of the ALCS system. The program configuration data set (CDS1) can be used to manage the names of the application load modules that are to be loaded during the next ALCS restart. CDS1 contains a program configuration table that includes the names of the load modules to be reloaded on each ALCS restart. For information on how you create and manage CDS1, see "Using the program configuration data set" on page 210.

The program configuration data set is optional, therefore if your system does not use CDS1, ALCS manages the online loading of program load modules in the following way.

- **Loading programs for system-wide use**

  Use the ZPCTL command to load, unload, or promote programs for system-wide use.

  – *Loading programs for system-wide use*
  Load a new or modified program (or programs); that is, load a program load module. ALCS flags the module as unloadable and updates the program control table. For each program or transfer vector, it creates an entry in the program control table. It chains this entry to the earlier entry (if there is one) for the same program, and sets a multiple entry flag.

    To resolve calls to a program with multiple entries, program management compares the entry create time with the module load time. If the entry create time is before the module load time, it selects the old copy of the program. If the entry create time is after the module load time, it selects the new copy of the program. A time-initiated program management routine resets the multiple-entry flags when all entries created before the module load time have exited.

  – *Unloading programs for system-wide use*
  Unload (delete) a program (or programs); that is, unload a program load module. ALCS program management flags the program control table entries as unloaded.

    To resolve calls to a program flagged as unloaded, program management compares the entry create time with the module unload time. When the entry create time is before the module unload time, it selects the unloaded copy of the program. When the entry create time is after the module unload time, it selects the old copy of the program.

    A time-initiated program management routine deletes the program control table entries when all entries created before unloading the module have exited.

  – *Promoting programs for system-wide use*

Make a program (or programs) permanent; that is, promote a program load module. ALCS program management flags the load module so that it cannot be unloaded accidentally. You should make programs permanent when you are confident that they are functioning correctly.

To load system-wide programs during ALCS restart (that have been previously loaded online by ZPCTL), you must update the load module list in the program configuration table to include the load module name. Add the name of the load module to the end of the load module list. See "Updating the program configuration table" on page 213 for details. If your system is using the program configuration data set, ALCS will update the load list for you.

- **Loading programs for test use**

  Use the ZPCTL command to load or unload programs for test use:

  – *Loading programs for test use*
  Load a test program (or programs); that is, load a test program load module. ALCS program management flags the module as a test module. When program management loads a test module, it does the following:

  - Puts the owning terminal in test mode
  - For each program in the test module, it creates a program table entry and chains it to the main entry.
  - For each program in the test module, it flags the program table entry to indicate that a test copy exists. For a new program, it creates an entry and then flags it.

  To resolve calls to a program with test copies, program management uses the test terminal flag for the entry.

  When ALCS communication management receives input from a terminal in test mode, it sets the test terminal flag for the entry. When this flag is on, program management tries to select test copies of programs. When the originating terminal address for the entry matches that in the program table, it selects the test copy of the program. When there is no match, it selects the original copy of the program.

  – *Unloading programs for test use*
  Unload (delete) a test program (or programs); that is, unload a test program load module. ALCS program management deletes the program table entries.

  You can load and unload test versions of application programs and ECB-controlled installation-wide exit programs.

Loading installation-wide monitor exit load modules

After the initial load of the installation-wide monitor exits on your ALCS system, you can load additional installation-wide monitor exit load modules online (for new or modified monitor exits) using the ZPCTL command (see *ALCS Operation and Maintenance*).

Installation-wide monitor exit load modules that are loaded online with the ZPCTL command may not be automatically reloaded during the next restart of the ALCS system. The program configuration data set (CDS1) can be used to manage the names of the installation-wide monitor exit load modules that are to be loaded during the next ALCS restart. CDS1 contains a program configuration table that includes the names of the load modules to be reloaded on each ALCS restart. For information on how you create and manage CDS1, see "Using the program configuration data set" on page 210.

The program configuration data set is optional, therefore if your system does not use CDS1, ALCS manages the online loading of installation-wide monitor exit load modules in the following way.

- **Loading monitor exits for system-wide use**

  Use the ZPCTL command to load, unload, or promote installation-wide monitor exits.

  – *Loading installation-wide monitor exit load modules for system-wide use*
  Load a new or modified installation-wide monitor exit(s); that is, load an installation-wide monitor exit load module.

ALCS flags the module as unloadable and updates the installation-wide monitor exit control table. ALCS updates the control table for each exit loaded in the load module. ALCS chains this entry to the earlier entry (if there is one) for the same installation-wide monitor exit. Only the latest version of the exit is activated.

– *Unloading installation-wide monitor exit load modules for system-wide use*
Unload (delete) an installation-wide monitor exit (or exits); that is, unload an installation-wide monitor exit load module.

ALCS removes the reference to the exit(s) and instead calls the earlier loaded version of the exit (if any). A time-initiated program management routine removes the load module when all entries created before unloading the load module have exited.

– *Promoting installation-wide monitor exits for system-wide use*
Make a installation-wide monitor exit permanent; that is, promote an installation-wide monitor exit load module.

ALCS program management flags the load module so that you cannot unload it accidentally. You should make an installation-wide monitor exit permanent when you are confident that it is functioning correctly.

To load system-wide installation-wide monitor exits during ALCS restart (that have been previously loaded online by ZPCTL), you must update the installation-wide monitor exit load list in the program configuration table to include the load module name. Add the name of the load module to the end of the load list. See "Updating the program configuration table" on page 213 for details. If your system is using the Program Configuration Data Set, ALCS will update the load list for you.

- **Loading monitor exits for test use**

  You cannot load installation-wide monitor exits for test use.

Using the program configuration data set

You can use the program configuration data set (CDS1) to manage the list of program and installation-wide monitor exit load modules that must be loaded during the next ALCS system restart. CDS1 contains two program configuration tables, a **current** and **alternate** table. The current table normally contains the list of load modules to be loaded during the next ALCS restart. The alternate table normally contains a backup load module list. The CDS1 data set is comprised of two *slots*. One slot contains the current table and the other slot contains the alternate table. They are called **slot A** and **slot B**.

When the ZPCTL command is used to load additional load modules online, if the load module is for system-wide use, ALCS adds the name of the load module to the load list in the current program configuration table on CDS1. You can use the ZPCTL command to confirm the load module. This ensures that ALCS will reload the load module during the next system restart.

The ZPCTL command can be used for loading a new base program configuration table on CDS1. From time to time, you can merge the additional programs and installation-wide monitor exits that have been loaded online into your base program configuration table. For example, if you have used ZPCTL to load a new version of an application program, you can replace the old version in the base program configuration table with the latest version. The process of replacing old versions of programs and exits with the latest version (called consolidation) is performed offline. When this process is complete, use the ZPCTL command to load and confirm the new base program configuration table. When you have confirmed the new base program configuration table, ALCS will use it for loading the program and installation-wide monitor exit load modules during the next ALCS restart.

To use the program configuration data set (CDS1), update the DBGEN macro parameters in the ALCS data base generation. Code the DBGEN CDS1=YES parameter to request ALCS to use the program configuration table on CDS1 during ALCS restart. The program configuration table referenced by the PARM field on the EXEC statement is loaded onto CDS1 on the initial startup of your ALCS system, and is then bypassed on subsequent ALCS restarts. During this initial startup of your system, ALCS formats the new CDS1 data set, loads the program configuration table on to it, and sets the status of the load list and each load module in the load list as *committed*.

Note: Committed is synonymous with promoted.

- **Using CDS1 for loading programs and monitor exits**

  When program and installation-wide monitor exit load modules are loaded online for system-wide use, additional functions are performed if your ALCS system is using CDS1. The basic functions performed by ZPCTL are described in "Loading program load modules" on page 208 and "Loading installation-wide monitor exit load modules" on page 209. The following describes the additional functions performed by ZPCTL when CDS1 is used.

  – *Loading programs and installation-wide monitor exits*
    When a load module has been successfully loaded by the ZPCTL LOAD command, ALCS adds the name of the load module to the load list in the current program configuration table on CDS1. The load list entry for that load module is stamped with the date and time of the load and marked as *loaded*. On the next ALCS restart, this load module will not be reloaded.
  – *Confirming programs and installation-wide monitor exits*
    When a load module that has been loaded is safe to preserve across an ALCS restart, it can be confirmed by the ZPCTL CONFIRM command. The ZPCTL command accesses the load list entry for that load module on CDS1, stamps the entry with the date and time of the confirm and marks it as *confirmed*. On the next ALCS restart, this load module will be reloaded.
  – *Unloading programs and installation-wide monitor exits*
    If a load module is to be unloaded (backed out), you can do this by using either the ZPCTL UNLOAD command or the ZPCTL BACKOUT command. The ZPCTL command accesses the load list entry for that load module on CDS1, stamps the entry with the date and time of the unload and marks it as *backed out*. On the next ALCS restart, this load module will not be reloaded.
  – *Promoting programs and installation-wide monitor exits*
    When a load module that has been confirmed is ready to be made permanent, you can promote (commit) the load module by using either the ZPCTL PROMOTE command or the ZPCTL COMMIT command. Load modules that are promoted (committed) can not be unloaded (backed out). The ZPCTL command accesses the load list entry for that load module on CDS1, stamps the entry with the date and time of the promotion and marks it as *committed*.

- **Loading program configuration tables on CDS1**

  From time to time, you can consolidate the programs and installation-wide monitor exits that have been loaded online into your base program configuration table. For example, during a period of a few weeks, many new versions of application programs could be loaded online, and the original versions of these programs need replacing with the latest version in the base program configuration table load list. The program configuration table consolidation is performed in two stages.

  The first stage is to create a new base program configuration table. The load list in the current program configuration table on CDS1 is used as the basis for creating the new base program configuration table. Create load modules that contain the latest version of all programs and installation-wide monitor exits. Create a load list that contains the name of each load module, and then prepare the new base program configuration table. See "Updating the program configuration table" on page 213 for details on how to do this.

  The second stage is to load the new base program configuration table onto CDS1, confirm it and restart the ALCS system. During the ALCS restart, the load modules referenced in the new program configuration table load list will be loaded. The ZPCTL command is used for loading the new program configuration table onto CDS1. The following describes the ZPCTL commands that you use for loading and activating the new program configuration table on CDS1.

  – *Loading a program configuration table onto CDS1*
    Use the ZPCTL LOAD LIST command to load the new program configuration table onto CDS1. This program configuration table becomes the alternate table on CDS1 and is marked as *loaded*. It will not be used by ALCS until it has been confirmed. If a restart of the ALCS system occurs, the program and installation-wide monitor exit load modules referenced by the current table are reloaded.
  – *Confirming a program configuration table on CDS1*
    Use the ZPCTL CONFIRM LIST command to confirm the new program configuration table that you have loaded onto CDS1. This alternate table is marked as *confirmed* on CDS1 and will be used by

ALCS for loading the application and installation-wide monitor exit load modules during the next system restart.

- *Backing out a program configuration table on CDS1*
  Use the ZPCTL BACKOUT LIST command to backout the new program configuration table on CDS1. This alternate table is marked as *backed out* on CDS1. You can back out a table after it has been loaded or after it has been confirmed. If the alternate table has been confirmed and you have restarted your ALCS system with the alternate table, ALCS will continue to use the alternate table until the next system restart. When a restart of the ALCS system occurs, the program and installation-wide monitor exit load modules referenced by the current table are loaded.
- *Committing a program configuration table on CDS1*
  Use the ZPCTL COMMIT LIST command to commit the new program configuration table on CDS1. The table must have been previously confirmed by the ZPCTL CONFIRM LIST command. You can not commit a program configuration table until after you have verified the contents of the table via a restart of the ALCS system (the restart will load the application and installation-wide monitor exit load modules referenced in the alternate table). The program configuration table is now marked as *committed* on CDS1. You can not back out a table after it has been committed. The new program configuration table (which was the alternate table) now becomes the current table, and the table that was previously the current table becomes the alternate table (and the contents of that table are cleared).

- **Additional option for backing out a program configuration table on CDS1**

  When a new program configuration table has been confirmed on CDS1, you must schedule a planned restart of the ALCS system to verify the new program configuration table and the load modules that are included in the load list. After the restart, if the ALCS system is performing satisfactorily, you can *commit* the new program configuration table. If the system is not performing correctly, you may decide to fallback to the previous program configuration table on CDS1. To do this, enter the ZPCTL BACKOUT LIST command to backout the new table, and schedule another restart of the ALCS system.

  When you have restarted the ALCS system with a new program configuration table, you could experience serious problems that necessitate an immediate fallback to the previous program configuration table on CDS1. In normal circumstances, you could do this by entering the ZPCTL BACKOUT LIST command and immediately restart the ALCS system. However, the ALCS system might fail before it reaches IDLE system state (or before you are able to enter any ALCS commands), or the system could reach IDLE system state but may not accept any ZPCTL commands. If either of these situations occur, you can immediately restart the ALCS system with the previous program configuration table on CDS1, without entering the ZPCTL BACKOUT LIST command. Request ALCS to perform a *back out* of the new program configuration table during system restart by placing a *minus sign* in front of the program configuration table name in the PARM field on the EXEC statement that starts the ALCS job (for example, the table name would be in the format **-tablename**). During the restart, ALCS will not only perform the back out but also fallback to the previous program configuration table on CDS1 for loading the programs and installation-wide monitor exits.

- **Loading programs and exits prior to committing a program configuration table**

  When a new program configuration table has been loaded and confirmed, a period of time may elapse before the table is committed. For example, you may load a new program configuration table and schedule the restart of the ALCS system for three days later. After the system restart, you may decide to immediately commit the table, or you may decide to delay the commit. When a new program configuration table has been loaded but is between the confirmed and committed stages, you may need to load additional program or installation-wide monitor exit load modules. You can do this. When you load additional program or installation-wide monitor exit load modules, if CDS1 contains a new program configuration table that has been loaded and confirmed (but not committed), ALCS will include the name of the load module in both the current and alternate tables on CDS1. When you confirm or unload an additional program or installation-wide monitor exit load module, ALCS will update the load module status in both the current and alternate tables on CDS1. This will continue until you either commit or backout the new program configuration table. If you decide to backout the new program configuration table, you must first backout any additional program or installation-wide monitor exit load modules that have been loaded since the table was confirmed.

If this requirement to backout additional program or installation-wide monitor exit load modules does not conform to your installation standards, you can use the installation-wide monitor exit USRCDSB to force a backout of the new program configuration table, even if it contains additional program or installation-wide monitor exit load modules.

Updating the program configuration table

The program configuration table defines the following:

- Number of load modules expected in the system
- Number of programs and transfer vectors expected in the system
- Load modules (for system-wide programs and monitor exits) to load during restart.

Figure 51 on page 214 shows an example of job control statements to build a program configuration table. The name of the load module that contains the program configuration table is specified on the SYSLMOD DD statement of the link-edit step.

Update the following items in the program configuration table:

Program configuration table name
If the program configuration table is a new load module, specify a new load module name. Specify this name in the following places:

- The SYSLIN job control DD statement in step PGLSTASM
- The NAME parameter of the DXCH0ST macroinstruction and the CP0DT macroinstruction
- The SYSLIN and SYSLMOD job control DD statements in step PGLSTLK.

Number of application load modules
Specify the maximum number of application program load modules expected in the system at any time. This includes system-wide and test load modules. Specify this in the NBRMOD= parameter of the CP0DT macroinstruction.

Number of application programs
Specify the maximum number of programs and transfer vectors that are expected in the system at any time. Specify this in the NBRPGM= parameter of the CP0DT macroinstruction.

List of application load modules
The list of modules to load during ALCS restart. Update the list that starts at label MLIST to add, delete, or replace module names. This list can include two types of load module:

- System-wide modules that the operator can unload
- System-wide modules that the operator cannot unload.

For details of how to specify these load modules, see the comments preceding the load module list in Figure 51 on page 214.

List of installation-wide monitor exit load modules
The list of modules to load during ALCS restart. Update the list that starts at label XLIST to add, delete, or replace module names. This list can include two types of load module:

- System-wide modules that the operator can unload
- System-wide modules that the operator cannot unload.

For details of how to specify these load modules, see the comments preceding the load module list in Figure 51 on page 214.

```
//*--------------------------------------------------------------------*
//*        ALCS V2  PROGRAM LIST GENERATION                            *
//*--------------------------------------------------------------------*
//*        PARAMETERS YOU MAY WISH TO CHANGE:                          *
//*                                                                    *
//*           ALCS DATASET NAMES THAT START WITH THE RECOMMENDED       *
//*           HIGHER LEVEL QUALIFIERS DXC.V2R4M1                       *
//*                                                                    *
//*--------------------------------------------------------------------*
//*
//*
//PGLSTASM EXEC PGM=ASMA90,PARM='NODECK,OBJECT,LIST,XREF(SHORT),RENT'
//SYSLIB    DD  DSNAME=DXC.V2R4M1.DXCMAC1,DISP=SHR
//          DD  DSNAME=DXC.V2R4M1.DXCMAC3,DISP=SHR
//          DD  DSNAME=SYS1.MACLIB,DISP=SHR
//SYSUT1    DD  DSNAME=&&SYSUT1,UNIT=VIO,SPACE=(1700,(1200,100))
//SYSLIN    DD  DSNAME=&&SYSLIN(DXCPGLW0),UNIT=VIO,
//              SPACE=(CYL,(5,5,2)),DCB=(BLKSIZE=400),DISP=(,PASS)
//SYSPRINT  DD  SYSOUT=*
//SYSIN     DD  *
         TITLE ' *** PROGRAM CONFIGURATION TABLES ***'
         SPACE 1
         PRINT NOGEN
DXCPGLW0 CSECT ,
         DXCHOST NAME=DXCPGLW0
         SPACE 1
******** PROGRAM TABLES HEADER
         SPACE 1
         CP0DT ACTION=GEN,                                          -
               NAME=DXCPGLW0,                                       -
               MODLST=(MLIST,MLISTE),   START/END MODULE LIST       -
               MEXLST=(XLIST,XLISTE),   START/END MON IWE LIST      -
               NBRMOD=35,               NUMBER OF LOAD MODULES      -
               NBRPGM=1200              NUMBER OF PROGRAMS
         EJECT ,
```

Figure 51. (Part 1 of 2). Example JCL to assemble and link-edit the program configuration table

```
*====================================================================*
*        LOAD MODULE LIST                                            *
*====================================================================*
*        THIS IS LIST OF LOAD MODULES WHICH ARE LOADED DURING        *
*        ALCS INITIALIZATION IN THE SEQUENCE THAT THEY APPEAR HERE.  *
*        IF A PROGRAM APPEARS IN MORE THAN ONE MODULE, THEN THE LAST *
*        LOADED COPY BECOMES EFFECTIVE.                              *
*        THE FORMAT OF EACH LOAD LIST ENTRY IS -                     *
*               BYTES 0-7   = NAME OF THE LOAD MODULE                *
*               BYTE  8     = FLAGS                                  *
*                           = AL1(0) -> NOT UNLOADABLE BY OPERATOR   *
*                           = AL1(CP0MDUN) -> UNLOADABLE BY OPERATOR *
*               BYTES 9-11  = AL3(0)                                 *
*                                                                    *
*        DO NOT SPECIFY THE ALCS ECB CONTROL MONITOR PROGRAM LOAD    *
*        MODULES IN THIS LIST -- THEY ARE LOADED AUTOMATICALLY FIRST *
*--------------------------------------------------------------------*
         SPACE 1
MLIST    DC    0D'0'
         DC    CL8'DXCAI001',AL1(CP0MDUN),AL3(0)  IPARS RESERVATIONS
         DC    CL8'DXCAI002',AL1(CP0MDUN),AL3(0)  IPARS RESERVATIONS
         DC    CL8'DXCAI003',AL1(CP0MDUN),AL3(0)  IPARS RESERVATIONS
         DC    CL8'DXCAI004',AL1(CP0MDUN),AL3(0)  IPARS RESERVATIONS
         DC    CL8'DXCAI005',AL1(CP0MDUN),AL3(0)  IPARS RESERVATIONS
         DC    CL8'DXCAI006',AL1(CP0MDUN),AL3(0)  IPARS RESERVATIONS
         DC    CL8'DXCAI007',AL1(CP0MDUN),AL3(0)  IPARS RESERVATIONS
         DC    CL8'DXCAI008',AL1(CP0MDUN),AL3(0)  IPARS RESERVATIONS
         DC    CL8'DXCAI009',AL1(CP0MDUN),AL3(0)  IPARS RESERVATIONS
         DC    CL8'DXCAI010',AL1(CP0MDUN),AL3(0)  IPARS RESERVATIONS
         DC    CL8'DXCAI011',AL1(CP0MDUN),AL3(0)  IPARS RESERVATIONS
         SPACE 1
         DC    CL8'DXCAX001',AL1(CP0MDUN),AL3(0)  IPARS MSG SWITCHING
         DC    CL8'DXCAX002',AL1(CP0MDUN),AL3(0)  IPARS MSG SWITCHING
         DC    CL8'DXCAX003',AL1(CP0MDUN),AL3(0)  IPARS MSG SWITCHING
         SPACE 1
         DC    CL8'DXCAAGRL',AL1(CP0MDUN),AL3(0)  IPARS GLOBAL AREA DEF
         DC    CL8'DXCAARCP',AL1(CP0MDUN),AL3(0)  IPARS RECOUP DESCR
         DC    CL8'DXCAXECB',AL1(CP0MDUN),AL3(0)  ALCS USER EXITS
         DC    CL8'DXCAAUI0',AL1(CP0MDUN),AL3(0)  IPARS UIXX PROGRAMS
MLISTE   EQU   *
         SPACE 1
XLIST    DC    0D'0'
         DC    CL8'DXCAXMON',AL1(CP0MDUN),AL3(0)  ALCS MON EXITS
XLISTE   EQU   *
         SPACE 1
         LTORG ,
         SPACE 1
         END   ,
/*
//*
//*  LINK PROGRAM LIST TABLE
//*
//PGLSTLK  EXEC PGM=HEWL,
//             PARM='XREF,LIST,LET,NCAL,RMODE=ANY,AMODE=31,AC=1'
//SYSLIN   DD   DSNAME=&&SYSLIN(DXCPGLW0),DISP=(OLD,DELETE)
//SYSUT1   DD   DSNAME=&&SYSUT1,UNIT=VIO,SPACE=(1700,(600,100))
//SYSLMOD  DD   DSNAME=DXC.V2R4M1.DXCLMD4(DXCPGLW0),DISP=OLD
//SYSPRINT DD   SYSOUT=*
//
```

Figure 52. (Part 2 of 2). Example JCL to assemble and link-edit the program configuration table

Online communication table maintenance (OCTM)

OCTM enables you to define communications network changes the ALCS communication table via an online process, eliminating the requirement to perform an offline communications generation.

Please refer to the *OCTM User Guide* on the ALCS Web site for information about the ALCS OCTM facility.

Chapter 5. Migrating ALCS pool

This chapter describes how to migrate ALCS short-term and long-term pool.

ALCS Concepts and Facilities describes allocatable pool.

Type-1 short-term pool and type-1 long-term pool

By default, ALCS uses type 1 short-term pool support and type 1 long-term pool support. These are the original pool dispense methods and the only ones available in the earliest versions of ALCS.

You can continue to use type 1 short-term pool dispense for as long as you wish, however there are benefits in migrating to type 2 short-term pool.

You can continue to use type 1 long-term pool dispense for as long as you wish, however there are benefits in migrating to type 2 long-term pool.

The migrations of short-term pool and long-term pool are independent of each other. You can use type 1 short-term pool and type 2 long-term pool. You can use type 2 short-term pool and type 1 long-term pool.

Type 2 short-term pool dispense benefits

Type 2 short-term pool dispense has the following significant benefits when compared with type 1:

- It provides significantly better error detection and reporting.
- It does not round-up the number of records that you request to an integral multiple of 8256.
- It provides more stable and controllable redispense delays. There are two of these delays:
  - After an entry releases a short-term-pool record, there is a delay before ALCS redispenses the record to another entry. During this period, ALCS can detect and report any access to the record (this is an error, applications are not allowed to access a record after releasing it).
  - If an application dispenses a record but fails to release it, then there is a delay before ALCS redispenses the record to another entry. When this delay expires, ALCS reports the error before it redispenses the record.

You will probably want to take advantage of these benefits as soon as possible.

Type 2 long-term pool dispense benefits

Type 2 long-term pool dispense has the following significant benefits:

- It allows you to increase the number of records available for use as long-term pool dynamically - that is, without an ALCS outage.
- It allows you to reuse any fixed-file or short-term pool records that you delete. The deleted records become available for use as long-term pool.
- All LT pool directories are held in main storage so Recoup directory build only requires one scan of the Recoup general file.

You will probably want to take advantage of these benefits as soon as possible.

Allocatable pool file addresses

All real-time database records in ALCS have an allocatable pool file address.

Each fixed-file record has a fixed-file file address and an allocatable pool file address. Usually application programs only use the fixed-file file address.

Similarly, each short-term pool record has a short-term pool file address and an allocatable pool file address. Usually application programs only use the short-term pool file address.

Some, but not all, long-term pool records have both long-term pool and allocatable pool file addresses. All records dispensed by type 1 long-term pool dispense have both a long-term pool file address and an allocatable pool file address. But some records dispensed by type 2 long-term pool dispense only have an allocatable pool file address.

Test database

ALCS always saves the allocatable pool file address of the record in the copy that it writes to the test data set. This means that you cannot use, for example, the fixed-file file address of a record as the VSAM key to locate the record on the test data set.

Recoup tagging

The first time that you run Recoup on an ALCS system, it includes a process called "tagging". This tagging process reads and writes every fixed-file and short-term pool record on your database. This means your first Recoup run will take longer than normal.

Subsequent Recoup runs do not repeat the full tagging process, but only tag short-term pool records newly added after the previous Recoup.

ALCS does not tag newly-added fixed-file records until they are first filed by an application program.

Logging

ALCS always saves the allocatable pool file address of the record in the copy that it writes to the log dataset.

Database reorganization

ALCS always saves the allocatable pool file address of the record in the copy that it writes to a sequential file for the `ZDATA DUMP` command.

With former versions of ALCS it was possible to reorganize the layout of the database, in order to increase the size of the database, by offloading and reloading all the data. ALCS now uses online facilities to increase the size of the database. `ZDATA LOAD` can be used to load data to a database of the same physical layout as the one used by `ZDATA DUMP`, but it cannot be used if the physical layout is different.

See “Long-term pool file records” on page 184 on how to increase the size of long-term pool.

Migrating ALCS short-term pool

ALCS provides a special facility for migrating from type 1 to type 2 short-term pool dispense. This facility is designed to minimize the chance that you lose or corrupt any data that you may already have in short-term pool. It also allows for controlled fallback to type 1 short-term pool dispense if required.

Using DBHIST parameters to control migration

`DBHIST` parameters allow you to control the migration from predecessor ALCS systems and control the migration of pool dispense.

Use the `DBHIST` *xxxx*`_SHORT_TERM` form to control the migration to type 2 short-term pool dispense, and the fallback to type 1 short-term pool dispense.

Migration states for short-term pool file

ALCS allows fallback at certain points (states) during migration. You can switch between these states as often as you require during migration.

Figure 53 on page 218 shows these migration states, and how each DBHIST option affects the way ALCS dispenses and releases the short-term pool space.

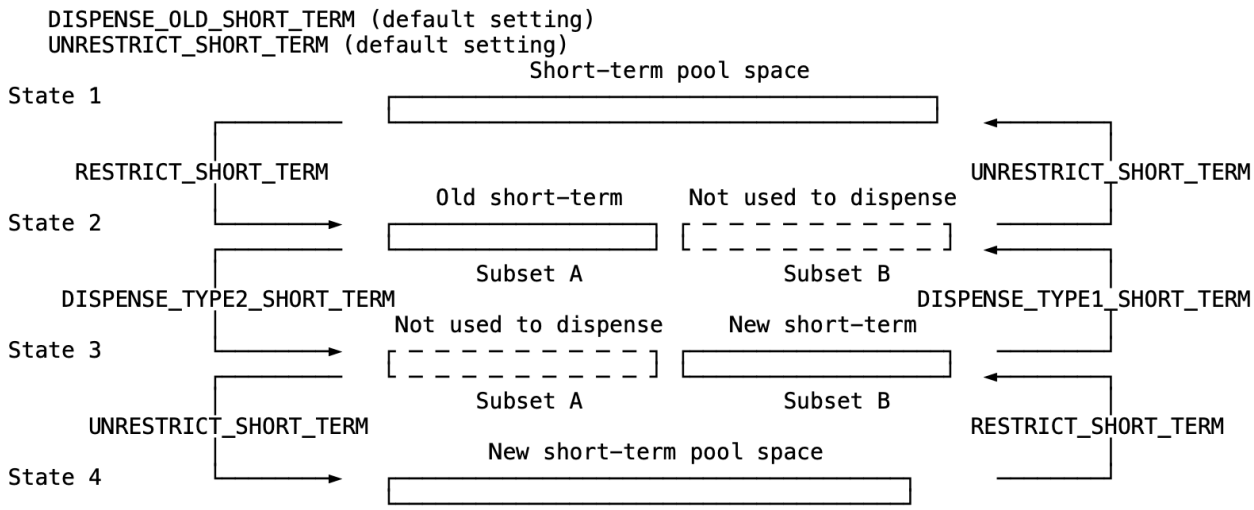


Figure 53. ALCS migration states: Short-term pool space

See "Short-term pool file records" on page 182 on how to increase and decrease the size of short-term pool.

State 1

The default values of DISPENSE_TYPE1_SHORT_TERM and UNRESTRICT_SHORT_TERM place the system in state 1. In this state, ALCS uses the type 1 method (one directory bit for each record) for short-term pool dispensing. ALCS dispenses from the whole range of short-term pool records which are defined when the database is first built.

Migrate to state 2

Specify RESTRICT_SHORT_TERM to migrate to state 2. DISPENSE_TYPE1_SHORT_TERM is the default method for dispensing. In state 2, ALCS uses the type 1 method to dispense records from subset A of the short-term pool space (see Figure 53 on page 218).

Fallback to state 1

Specify UNRESTRICT_SHORT_TERM to return to state 1. ALCS uses the type 1 method to dispense records from the whole short-term pool space.

Migrate to state 3

Specify DISPENSE_TYPE2_SHORT_TERM to migrate to state 3. ALCS uses the type 2 method (one directory byte for each record) to dispense records from subset B of the short-term pool space and eventually purges the records from subset A.

Fallback to state 2

Specify DISPENSE_TYPE1_SHORT_TERM (or ZDASD BACKOUT command) to return to state 2. ALCS continues to use the type 2 method to **release** short-term pool space (but not to dispense space). The new short-term pool (subset B) is purged within the following 24 hours (as successive short-term pool records are released).

Migrate to state 4

Specify UNRESTRICT_SHORT_TERM to migrate to state 4. ALCS uses the type 2 method to dispense records from the whole of the short-term pool space.

Fallback to state 3

Specify RESTRICT_SHORT_TERM to return to state 3. ALCS uses the type 2 method to release short-term pool, but does not dispense any short-term pool from the **old** space (subset A). The old short-term pool is purged within the following 24 hours (as successive short-term pool records are released).

Elapsed time for migrating

During migration from state 1 to state 4 you must keep the system in state 2 for some hours and then in state 3 for some hours. This time delay is necessary during the transition from one state to another to allow applications to release records before they are redispensed.

For example:

From state 1 to state 2, wait for subset B to be empty
From state 2 to state 3, wait for subset A to be empty
From state 3 to state 4, wait for subset A to be empty

Diagnostic file - short-term pool usage errors

Type 2 short-term pool dispense writes diagnostic information to the ALCS diagnostic file. This additional diagnostic information (compared with type 1 short-term pool dispense) can fill the diagnostic file more quickly and cause more frequent sequential file switches.

Migrating ALCS long-term pool

Note: You cannot fall back to type 1 long-term pool dispense when you migrate to type 2.

Using DBHIST parameters to control migration

DBHIST parameters allow you to control the migration from predecessor ALCS systems and control the migration of pool dispense.

Use the DBHIST DISPENSE_TYPE2_LONG_TERM form to control the migration to type 2 long-term pool dispense.

Application considerations

When type 1 long-term pool dispense is active, ALCS always dispenses long-term pool file addresses. When type 2 long-term pool dispense is active, it always dispenses allocatable pool file addresses. Following a long-term pool dispense, the application always uses the dispensed file address (long-term or allocatable) to address the record.

The existence of allocatable pool file addresses does not normally impact application programs, except for the following special cases:

- "PNR locator" and similar mechanisms

  Some applications generate an external token, based on a file address, that can be used to retrieve a particular record. An example is the passenger name record (PNR) locator that airline seat reservation applications such as IPARS use.

  IPARS converts the file address of a PNR into a six-character string called the PNR locator that is (for example) printed on the passenger's ticket. The program in IPARS that comes with ALCS that does this is PRC1. The application programs that convert between the file address and the PNR locator can

contain dependencies on the file address format. They may need changes to allow for two different types of file address (long-term pool and allocatable pool).

Product-sensitive Programming Interface

ALCS includes a product-sensitive interface (an extension to the FACE program function) that this type of application can use. FACE is described in *ALCS Application Programming Reference - Assembler*.

End of Product-sensitive Programming Interface

- Programs that systematically read long-term pools

  Some applications include functions that systematically read all the records in a long-term pool - including records that are not in use. Former ALCS versions do not include intended programming interfaces for this function. But some existing applications implement the function by using the internal interface macro PFACC to calculate consecutively all the valid file addresses in a long-term pool.

  When type 2 long-term pool dispense is activated, ALCS dispenses allocatable pool file addresses. Not all of these have corresponding long-term pool file addresses. Therefore, existing programs that use PFACC in this way do not work reliably.

  ALCS includes a general-use programming interface to read all data base records. The interface is an ECB-controlled program called CAP1. If you have applications that systematically read long-term pools, you must change them to use this new interface, so you can migrate to type 2 long-term pool. See “Scan data base program - CAP1” on page 421.
- Other programs that read not-in-use pool records

  ALCS normally regards any access to pool records that are not in use as an error and reports the error on the ALCS diagnostic file.

  Some application programs have a requirement to read a pool file record when they do not "know" if the record is in use or not. ALCS includes product-sensitive interfaces that this type of application can use. They are:

  Product-sensitive Programming Interface

  FNDPC - Find record, including pool control information. Do not report pool usage errors. BLKIC - Extract block size information, including offset of pool control information within record. RS0RS - DSECT macro for pool control information.

  These macros are described in *ALCS Application Programming Reference - Assembler*.

  End of Product-sensitive Programming Interface
- Programs that process ALCS database update log files ALCS always saves the allocatable pool file address of the record in the copy that it writes to the log file or files. If you have programs that read ALCS log files, you may need to change them to allow for this.

Migrating ALCS long-term pool procedure

About this task

Use the following procedure to migrate your long-term pool to type 2:

Procedure

1. Run a DASD generation that includes a DBHIST macro with DISPENSE_TYPE2_LONG_TERM. You can include other updates in this DASD generation.
2. Load, confirm, and commit the new DASD tables using the ZDASD command.
3. Run Recoup.

Results

Recoup automatically creates new type 2 directories and ALCS begins to dispense allocatable pool file addresses. At this stage you cannot fall back to type 1 long-term pool addressing.

Chapter 6. Online Communication Table Maintenance (OCTM)

Introduction

ALCS includes an online communication table maintenance facility which can help to manage changes in your communication network configuration.

OCTM is comprised of a number of separate elements, including an ALCS product-specific programming interface (COMTC macro), an ALCS operator interface (ZOCTM command), an OCTM database that contains communication resource definitions, and a communication end user system (CEUS). The CEUS is not part of the ALCS product, although an example CEUS front-end utility, based on 3270 screen maps, is available from the ALCS Web Site. You can use this, or you can develop your own CEUS utility.

The purpose of the CEUS is to provide a user-friendly way of specifying changes to the ALCS communication network and applying those changes directly in the ALCS communication table. The end-user submits communication change requests to your CEUS which checks if the change requests are valid and authorized. If they are, the CEUS presents them to ALCS using the COMTC programming interface. The end-user then requests your CEUS to load those change requests into the ALCS communication table, confirm and then commit them. The CEUS can also display changes to the ALCS online communication table and can manage change requests either singly or in batches. The COMTC programming interface enables the CEUS to perform all these functions.

We advise you to review the following information about the OCTM facility in case you want to exploit this new feature:

- ZOCTM command in *ALCS Operation and Maintenance*.
- "OCTM policing exit program - AOCM" on page 376.
- COMGEN generation macro in "COMGEN macro" on page 98.
- COMTC programming interface in Appendix L, "COMTC Communication Table Update Macro," on page 594.
- Guidance on the design of the CEUS front-end utility in Appendix K, "Communications End User System (CEUS)," on page 580.

Summary of OCTM Components

This section of the user guide provides a summary of the major components of OCTM.

COMTC Macro Interface

About this task

The COMTC Communications Table Update monitor-request macro provides the primary interface between the CEUS and OCTM. There are twelve different options available on the COMTC macro, providing the following primary functions:

- Query status of communication resource or group(s)
- Allocate and unallocate a communications group
- Submit communications change requests
- Activate communications change requests

When the CEUS submits *change requests* for the ALCS communication table via the COMTC macro, those change requests may be adding new communication resources, changing current resources or deleting

obsolete resources. The CEUS will issue a separate COMTC macro for the submission of each individual *change request* to OCTM. The CEUS can submit just a single change request (with a single COMTC macro) or a group of change requests (with multiple COMTC macros). When a group of *change requests* is being submitted, the CEUS can allocate an OCTM ***communications group*** that will be used to manage and identify that group. When the CEUS is ready to activate the group of change requests in the online communication table, the complete group can be activated together. The COMTC macro interface therefore provides functionality that allows the CEUS to ***allocate*** a communications group, use that communications group name when submitting each individual change request, check the status of that group, activate the complete group in the online communication, and finally ***unallocate*** the communications group.

When an ALCS system has been migrated to the OCTM facility, all permanent changes to the ALCS communication table (those to be retained over an ALCS restart) for terminal and X.25 PVC resources (types 1/6/7) must be submitted to ALCS via the COMTC macro. These changes can no longer be submitted to ALCS via the offline communications generation process. The ALCS COMCC monitor-request macro (COMCC - Alter communication resource information) can still be used for temporary changes to the ALCS communication table for all resource types.

The following is a list of the twelve options provided on the COMTC macro:

Procedure

1. COMTC QUERY - Query status of communication resource or group
2. COMTC GROUPS - Obtain status of communications groups
3. COMTC ALLOCATE - Allocate a new communications group
4. COMTC ADD - Add a new communication resource
5. COMTC REPLACE - Change a communication resource
6. COMTC DELETE - Delete a communication resource
7. COMTC CANCEL - Cancel a communication change request
8. COMTC LOAD - Load communication change requests
9. COMTC BACKUP - Back out communication change requests
10. COMTC CONFIRM - Confirm communication change requests
11. COMTC COMMIT - Commit communication change requests
12. COMTC UNALLOCATE - Unallocate a communications group

Results

In addition to the COMTC macro there are three data macros (DSECT's):

- The Communication Resource Definition DSECT (CT1TM)
- The Communications Resource and Group Information DSECT (CT2TM)
- The Communications Groups Information DSECT (CT3TM)

See "Overview of COMTC Macro Interface" on page 226 for more information on each of the twelve COMTC macro options.

OCTM Database

The OCTM database contains details of every terminal resource (except Prime and RO Cras), plus X.25 PVC resources, types 1/6/7. During restart of the ALCS system, the online communication table is built from data provided by the offline communications generation plus data from the OCTM database.

The OCTM database is comprised of two system fixed file areas. The first system fixed file contains the ***base communications area*** and the second system fixed file contains the ***update communications area***. The communication resources that are defined in the ***base communications area*** are loaded into the online communication table during ALCS restart (these are the confirmed/committed resources). The communication resources that are defined in the ***update communications area*** are the recent in-progress communication table changes (which are not loaded during ALCS restart). Records in the OCTM database

are 4K in size (L3) with each physical record holding a number of logical records. Each logical record contains details of a single communication resource. Those details include both the communications system data and the user data.

The first ten physical records in the ***base communications area*** contain the OCTM control record and the OCTM communications group name records. The control record contains information that is vital for the successful operation of the OCTM facility. The communications group name records control the management of the OCTM communications groups.

ALCS dynamically obtains available space from the L3 allocatable pool when it builds the OCTM database and when it expands it. ALCS users must ensure that their L3 allocatable pool is large enough to accommodate the needs of the OCTM database. See “Initialize the OCTM database” on page 229 for information on the L3 record space requirements of OCTM.

OCTM Communication Generation Parameters

Three parameters are provided on the ALCS communication generation COMGEN macro for OCTM. The first of these three is the **OCTM**= parameter. This parameter should be coded as OCTM=YES if the ALCS system is using the OCTM facility. The other two COMGEN parameters are **CRIRANGE**= and **ORDRANGE**=.

When the migration to OCTM is complete, the management of the communication resources in the ALCS communication table will be shared between the OCTM facility and the offline communications generation. The allocation of CRI addresses and resource ordinal numbers to new communication resources must therefore be carefully managed so that duplicate usage of them does not occur. The **CRIRANGE**= and **ORDRANGE**= parameters provide a mechanism for managing the allocation of new CRIs and ordinals. These COMGEN parameters enable explicit ranges of CRI addresses and resource ordinal numbers to be defined for the exclusive use of the offline communications generation. OCTM will not use any CRI addresses or ordinal numbers that are defined in these COMGEN parameters. We therefore recommend that the **CRIRANGE**= and **ORDRANGE**= parameters are included in the base communications generation deck so that duplicate usage of CRI addresses and resource ordinals can be avoided.

Details of these three COMGEN parameters is provided in ALCS Installation and Customization.

OCTM Policing

An OCTM policing function monitors the contents of the base and update communications areas on the OCTM database. OCTM policing is activated once each hour and its primary function is to monitor the status of communication resources and groups, and to identify periods of inactivity since the last COMTC macro was issued by the CEUS. If a period of inactivity is detected, OCTM policing sends a warning message to the RO Cras, indicating that further action is required. OCTM also activates an ECB-controlled installation-wide exit that can be used to perform additional functions. For example, the policing exit program can suppress transmission of the warning message and can issue a COMTC macro to activate the next COMTC function for the communication resource or group.

When the OCTM policing function is checking for periods of inactivity (for a communications resource or group) it does this for four COMTC macros:

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

For example, if a COMTC CONFIRM has been issued for a communications group, but after 2 days the group has still not been committed (via COMTC COMMIT), the policing exit can decide to suppress the warning message and issue a COMTC COMMIT for that communications group. See “Using the OCTM policing exit” on page 591 for more information about OCTM policing.

ZOCTM Operator Commands

The OCTM facility provides operator commands for the ALCS operations personnel. The **ZOCTM - Control online communication table maintenance** command enables operations personnel to control the usage of OCTM, perform OCTM database backup/restore functions and to display OCTM status information. The ZOCTM command is also used to activate the OCTM database build function during the migration to OCTM. The commands are restricted to Prime Cras and Alternate Cras (AT1 to AT16). Details of each ZOCTM command can be found in ALCS Operations and Maintenance.

ALCS 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 include in its report details of the communication resources that are managed by OCTM. (DXCCOMOL only includes communication resources that are confirmed or committed; it does not include any unconfirmed changes.) The DXCCOMOL communications report provides a valuable summary of all the communication resources defined in the ALCS communication table.

When the OCTM facility 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 ZOCTM BACKUP command is used to create this sequential file (it copies the contents of every record on the OCTM database to the sequential file).

DXCCOMOL produces an output file that contains details of every communication resource. That output file is in a format that allows the IEBPTPCH utility to print selective details of each resource. If required, a utility can be used to sort the contents of the output file prior to printing it with IEBPTPCH (for example, sort the resources into resource ordinal number sequence). Information about running DXCCOMOL can be found in *ALCS Operation and Maintenance*.

OCTM Offline Support Program

About this task

The communication resource definitions on the OCTM database are normally modified by the CEUS (via the COMTC macro), but they can also be modified by the OCTM Offline Support program, DXCCTMOL. This offline program can be used to modify the OCTM database when changes are needed in a large number of communication resources. This offline program can also 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 be used to create a dataset that contains an ALCS generation COMDEF macro for every communication resource on the OCTM database. The primary input to this offline program is the OCTM sequential file which is created by the OCTM database backup function (see *ALCS Operation and Maintenance*). The following is a summary of the functionality provided by DXCCTMOL:

Procedure

1. Modify and Delete communication resources on the OCTM database

   Use this program to modify multiple communication definitions on the OCTM database. For example, change the "test" status of communications resources, change the "initial" status, change and reformat the user data, etc. You can also use this program to delete a large number of communication resources that require removal from the OCTM database. Use this offline program in conjunction with the installation-wide exit program DXCUTMOL to apply these changes to the OCTM database. The DXCUTMOL exit program must identify the communication resources to be deleted (and mark them for deletion) and also identify the communication resources to be modified (and apply the required modifications). The exit program is activated by DXCCTMOL for every communications resource on the input OCTM sequential file. DXCCTMOL creates an output OCTM sequential file that contains all the communication resources that have not been deleted, and for those that have been modified, the updated communications definitions.
2. Validate and implement new CRIRANGE and ORDRANGE parameters

Use this program to validate and implement modified ranges of CRI addresses and resource ordinal numbers in the communications generation **CRIRANGE**= and **ORDRANGE**= parameters (these parameters specify ranges of CRI addresses and ordinal numbers that are for the exclusive use of the offline communications generation). This offline program identifies the communication resources on the OCTM database that are using any of the CRI addresses or resource ordinals that are within these new ranges and outputs a report listing the names of those communication resources. This offline program can then be used to update the OCTM database with the modified ranges of CRI addresses and ordinal numbers and to delete resources that have either a CRI address or ordinal number that is within the modified ranges. DXCCTMOL creates an output OCTM sequential file that contains the new ranges of CRIs and ordinals (in the OCTM control record) and also contains the communications resources that have not been deleted.

3. Create communications generation COMDEF macros

   Use this program to create a sequential dataset that contains a COMDEF macro for every communication resource in the base communications area of the OCTM database. This enables communication generation decks to be created for the communication resources managed by OCTM.

Results

For the functions in the above list which create on output OCTM sequential file, the OCTM database restore function can be used (see *ALCS Operation and Maintenance*) to process that sequential file and rebuild the OCTM database on the ALCS system.

Overview of COMTC Macro Interface

About this task

The COMTC macro allows the CEUS to obtain information about communication resources, submit change requests for those resources and to permanently update the online communication table. The following describes the primary CEUS functions that use the COMTC macro.

Procedure

1. Query status of communication resource or group(s)

   Query the status and obtain information about a communication resource or a communications group. This is provided by the COMTC QUERY macro.
   Query the status of all currently allocated communications groups. This is provided by the COMTC GROUPS macro.

2. Allocate and unallocate a communications group

   Allocate a communications group name for a batch of communication change requests and unallocate the group name when it is no longer required. This is provided by the COMTC ALLOCATE and COMTC UNALLOCATE macros. A communications group is required when multiple change requests are being submitted to ALCS via the COMTC macro, and those change requests must be grouped together under the name of a communications group.

3. Submit communications change requests

   Submit different types of change request for the communication resources. This is provided by the COMTC ADD, COMTC REPLACE and COMTC DELETE macros. This enables new communication resources to be added, current resources to be changed, and obsolete resources to be deleted. The COMTC CANCEL macro can be used to cancel a previous COMTC ADD, REPLACE or DELETE.

4. Activate communications change requests

   Activate a single change request or a group of change requests in the online communication table. This is provided by the COMTC LOAD, COMTC CONFIRM and COMTC COMMIT macros. When all the required change requests have been submitted to ALCS, they are activated in the online communication table via a three stage process. Those three stages are called load, confirm and commit. After a load

or confirm, the change requests can be backed out of the online communication table by a COMTC BACKUP macro.

Results

There are twelve different COMTC actions that can be requested by the CEUS. The following provides a brief description of each of those twelve COMTC actions.

- QUERY

  Query status and obtain information about any communication resource. Also, query status and obtain information about a communications group, including the list of communication resources that belong to the group. See "COMTC QUERY - Query status of communication resource or group" on page 594 for details of the COMTC QUERY macro. The information is provided by COMTC QUERY in the Communications Resource and Group Information DSECT (CT2TM). See "CT2TM - Communications Resource and Group Information DSECT" on page 653 for details of this DSECT.

- GROUPS

  Query status and obtain information about every currently allocated communications group. When you need to allocate a new communications group, you may wish to determine if the communications group name is already in use. The COMTC GROUPS macro allows you to obtain a list of the currently active groups and the status of each group. See "COMTC GROUPS - Obtain status of communications groups" on page 597 for details of the COMTC GROUPS macro. Basic information about each communications group, together with status information, is provided by COMTC GROUPS in the Communications Groups Information DSECT (CT3TM). See "CT3TM - Communications Groups Information DSECT" on page 657 for details of this DSECT.

- ALLOCATE

  Allocate a new communications group. When you have a group of change requests to submit, use COMTC ALLOCATE to allocate a group name for them. It is easier to manage a group of changes when a group name has been allocated for them. When change requests have been submitted (via COMTC) for the communications group, a single COMTC LOAD macro will load all the change requests into the online communication table. See "COMTC ALLOCATE - Allocate a new communications group" on page 599 for details of the COMTC ALLOCATE macro.

- ADD

  Add a new communication resource. When a new communications resource is being added, full details of the resource must be provided, including its CRN, the system data and user data. Details of the resource are provided on COMTC ADD macro parameters and via the Communication Resource Definition DSECT, CT1TM. See "COMTC ADD - Add a new communication resource" on page 601 and "CT1TM - Communication Resource Definition DSECT" on page 631. If a group of new resources are being added, the name of the communications group must also be provided. Details of the new resource provided by COMTC ADD are written to the OCTM database.

- REPLACE

  Change a communication resource. The current system and user data for a communication resource can be changed by COMTC REPLACE. The CRN or CRI address is used to identify the resource being changed. Details of the changes are provided in a storage block formatted by the Communication Resource Definition DSECT, CT1TM. See "COMTC REPLACE - Change a communication resource" on page 607 and "CT1TM - Communication Resource Definition DSECT" on page 631. If a group of resources are being changed, the name of the communications group must also be provided. The information provided by COMTC REPLACE is written to the OCTM database.

- DELETE

  Delete a communication resource. The communication resource being deleted must currently exist on the OCTM database. The CRN or CRI address is used to identify the resource being deleted. If a group of resources are being deleted, the name of the communications group must also be provided. The communication resource being deleted is flagged as deleted in the OCTM database. See "COMTC DELETE - Delete a communication resource" on page 611 for details of the COMTC DELETE macro.

- CANCEL

  Cancel a communication change request. Communication change requests are submitted by COMTC ADD, COMTC REPLACE and COMTC DELETE macros. After submitting the change requests, if one is incorrect (for example, the wrong CRN is used in a COMTC DELETE), the COMTC CANCEL can be used to cancel the change request. If the change request being cancelled belongs to a communications group, the name of the communications group must be provided. Only those change requests that have not yet been loaded via COMTC LOAD can be cancelled. The cancel deletes the change request from the OCTM database. See "COMTC CANCEL - Cancel a communication change request" on page 613 for details of the COMTC CANCEL macro.

- LOAD

  Load communications change requests in the online communication table. Change requests submitted via COMTC ADD, COMTC REPLACE and COMTC DELETE macros are loaded into the online communication table. For example, if six new communication resources had been added in a group, those six resources are now added to the online communication table. If two communication resources had been deleted, those resources are now removed from the online communication table. A single communication change request can be loaded or a group of change requests can be loaded. See "COMTC LOAD - Load communication change requests" on page 616 for details of the COMTC LOAD macro.

- BACKOUT

  Back out communications change requests from the online communication table. A COMTC BACKUP can be used after a COMTC LOAD or a COMTC CONFIRM. If a COMTC LOAD was used to load a group of change requests, the COMTC BACKUP will back out (remove) that group of change requests. For example, if a group of change requests included some COMTC DELETE requests, the COMTC BACKUP reinstates the deleted resources in the online communication table. The change requests that have been backed out still reside on the OCTM database. They can be subsequently removed from the OCTM database (if they are no longer required) by a COMTC CANCEL. See "COMTC BACKUP - Back out communication change requests" on page 619 for details of the COMTC BACKUP macro.

- CONFIRM

  Confirm communications change requests. Communication change requests should be confirmed after they have been successfully loaded. When communication change requests have been confirmed, they are retained in the online communication table over an ALCS restart. The logical records are flagged as confirmed in the OCTM database. A single communication change request can be confirmed or a group of change requests can be confirmed. See "COMTC CONFIRM - Confirm communication change requests" on page 622 for details of the COMTC CONFIRM macro.

- COMMIT

  Commit communications change requests. Communication change requests should be committed after they have been successfully confirmed. When communication change requests have been committed, they reside permanently in the OCTM database and the online communication table. When the change requests belong to a communications group, the communications resources that were added, updated or deleted by those change requests are no longer locked to the communications group. See "COMTC COMMIT - Commit communication change requests" on page 625 for details of the COMTC COMMIT macro.

- UNALLOCATE

  Unallocate a communications group. When all change requests for a communications group have been committed, the communications group name should be deleted. The COMTC UNALLOCATE deletes the group name from the OCTM database. See "COMTC UNALLOCATE - Unallocate a communications group" on page 627 for details of the COMTC UNALLOCATE macro.

Migrating to OCTM

Specific tasks must be performed for the migration of the terminal and X.25 PVC resources (types 1/6/7) from the offline communication generation process to the OCTM facility. The offline communication generation process will continue to be used for the management of non-terminal communication

resources. None of the facilities provided in ALCS for managing the communications resources have been made obsolete by OCTM. For example, if the extended CDS support (CDS2) is being used for managing the communication load list, usage of this can continue.

The primary tasks that must be performed for the OCTM migration are:

- Update the ALCS communications generation
- Initialize the OCTM database
- Build the OCTM database
- Activate OCTM

Before commencing the migration process, add the OCTM pool record ID of AC10 to the L3ST pool in your DASD configuration, if it is not already defined.

Update the ALCS communications generation

The ALCS communications generation COMGEN macro must be updated. Code the OCTM=YES parameter on the COMGEN macro to specify that the ALCS system will be using the OCTM facility. Apply this update to the base communication generation deck and restart the ALCS system with this updated communications generation.

Perform a consolidation of your communications generation COMDFLT and COMDEF macros. Many ALCS users have large terminal networks and therefore maintain their base COMDFLT and COMDEF macros in multiple generation decks. Those ALCS users may also have many small generation decks which have been loaded online via the ZACOM LOAD command. The COMDFLT and COMDEF macros in those small generation decks must be consolidated into the base generation decks before the OCTM database is initialized. Build a new communication configuration load list that contains a list of the new consolidated generation decks. If the communications CDS (CDS2) is being used, load this new load list via the ZACOM LOAD,LIST command (then confirm and commit this load list). Also, update the name of the communication configuration load list in the PARM field on the EXEC statement that starts the ALCS job. Do this even if you are using CDS2. When the OCTM database is built (see "Build the OCTM database" on page 230), ALCS uses the communication configuration load list defined in the EXEC statement PARM field, it does not use the load list on CDS2.

There are two further COMGEN generation parameters that should be coded *after* the OCTM migration is complete. These are the **CRIRANGE**= and **ORDRANGE**= parameters. These parameters ensure that CRI addresses and ordinal numbers used by the communications generation process do not conflict with those used by the OCTM facility. Conflicts could occur when new communication resources are being added to ALCS via the offline communications generation and via OCTM at the same time. Refer to ALCS Installation and Customization for a description of these COMGEN parameters.
If these parameters are coded *before* the OCTM migration, they will be used by the OCTM database build function (see "Build the OCTM database" on page 230) to exclude all communication resources whose CRI address or ordinal number is within the ranges defined by these parameters.

Soon after the OCTM migration is complete, the offline communication generation deck(s) should be condensed by removing all generation macros (COMDFLT and COMDEF) that define resources which are now being managed by the OCTM facility. During this condense process, the COMGEN **CRIRANGE**= and **ORDRANGE**= parameters should be coded to specify the ranges of CRIs and ordinal's that will be used by the non-terminal resources in the communication generation deck. See "Condensing the Communication Generation Decks" on page 231 for further information on this.

Initialize the OCTM database

The initialization of the OCTM database is performed automatically by ALCS. When the ALCS system is restarted with a base communications generation load module that includes an updated **OCTM**= parameter (updated from OCTM=NO to OCTM=YES on the COMGEN macro) it checks for the existence of an OCTM database. If none exists, it performs the OCTM database initialization function. During this ALCS restart, the programs that comprise the CEUS application could be loaded. Before ALCS initializes the OCTM database, it will build the online communication table using the current ALCS communications load

list (on the EXEC statement PARM parameter or on the communications CDS2). The following describes the OCTM database initialization process.

- `Allocate the OCTM system fixed files`

  The OCTM database is comprised of two system fixed file areas, each containing L3 sized records (4000-byte records). The first system fixed file contains the ***base communications area*** and the second contains the ***update communications area***. ALCS allocates the initial records required for these two system fixed files. These initial records are taken from the L3 allocatable pool, and ALCS verifies that there are at least 500 available records remaining in the L3 allocatable pool before it performs the allocation process (if there are less than 500, the OCTM database initialization function will fail). For this initial allocation, only eleven records are obtained for the ***base communications area*** and one record obtained for the ***update communications area***. Further records are obtained when the OCTM database build function is performed (see below).

- `Initialize eleven records in base communications area`

  The first record in the base communications area is initialized as the OCTM control record. The following nine records are initialized as communications group name records. The eleventh record is initialized as the first base communication record.

In the ***base communications area***, apart from the first ten records, every record is a base communication record. ALCS fits a maximum of 15 logical records into each physical record (each logical record is a minimum of 264 bytes). If the communications user data area is used, the size of the logical record is increased to accommodate this. This may reduce the number of logical records that fit within a physical record, for example, if the user data area is 40 bytes long, ALCS will fit 13 logical records within a physical record.

In the ***update communications area***, ALCS fits 6 logical records into each physical record (each logical record is a minimum of 648 bytes).

Each communication resource in the ***base communications area*** is held in a condensed C0OIN macro format. Each logical record not only contains details of a communication resource, but also critical information that is required by the OCTM facility. For example, when a `COMTC` macro is issued for the resource, information about that `COMTC` (the `COMTC` action, the time/date when the `COMTC` was issued, etc.) is stored in the logical record in the ***base communications area***. If a `COMTC` macro is updating a current communication resource, the logical record in the ***base communications area*** contains a pointer to a logical record in the ***update communications area*** (which contains details of the update).

When the OCTM database build occurs (see below) additional records will be required for the ***base communications area***. These will be dynamically obtained from the L3 allocatable pool in batches of 50 records. When usage of the OCTM facility commences and additional records are required for the ***update communications area***, these will also be dynamically obtained from the L3 allocatable pool. ALCS will obtain additional records, as required, in batches of 50 physical records (each batch providing 300 logical records). Each time ALCS requires an additional batch of 50 records from the L3 allocatable pool, it verifies that there are at least 500 available records remaining in the L3 allocatable pool.

Build the OCTM database

After the OCTM database initialization is complete, the ALCS system can be brought up to NORM system state. The operator command `ZOCTM BUILD` can now be used to activate the OCTM database build function. The OCTM database is built from the definitions of the terminal and X.25 PVC resources (types 1/6/7) in the communications generation load module(s). OCTM accesses the communications load list, reads each communications generation load module, and populates the ***base communications area*** in the OCTM database with all the required definitions (except Prime and RO Cras). The `ZOCTM STATUS` command can be used to verify the status of OCTM both before and after the OCTM database build.

Each ALCS user must ensure that they have sufficient L3 allocatable pool records to accommodate the requirements of the OCTM database before activating the OCTM database build. When the OCTM database has been built, a restart of the ALCS system is required to enable the OCTM facility to become fully operational (as described below).

Before restarting ALCS, a verification of the OCTM database could be performed to ensure that it is fully ready for operational use. To do this, run the OCTM database backup function (ZOCTM BACKUP) to create an OCTM sequential file. Run the ALCS Communications Report program (DXCCOMOL) to obtain a report of all the communication resources, including those on the OCTM database. Review the communications report and verify that it includes all the communication resources.

Activate OCTM

To fully activate OCTM, two further actions are required. Firstly, another restart of ALCS is required to build all the core resident tables that are required for operational usage of OCTM. Secondly, when the ALCS system has reached either IDLE or NORM system state, the ZOCTM START command must be entered to fully activate OCTM, therefore allowing the system administrators to start using the CEUS (enabling them to utilize the functionality provided by the OCTM facility).

During system restart, ALCS builds the online communication table from the communications generation load module(s) and the OCTM database. It also builds many core resident tables which are required by the operational OCTM facility. Ensure that the same communications generation load modules (same communications load list) are used by the ALCS system when the OCTM database initialization function is performed and when the OCTM activation is performed. When ALCS builds the online communication table during restart, it first reads the communications generation load module(s) and then reads all the logical records in the ***base communications area*** of the OCTM database.

During this ALCS restart, when ALCS builds the online communication table, it will find identical resources in the communications generation load module(s) and the OCTM database. These resources are the terminals (except Prime and RO Cras) and the X.25 PVC's (types 1/6/7). When ALCS restart finds these resources in the generation load module(s), it will treat them as *dummy* resources and not load them. It will though check in the generation load module(s) to see if these *dummy* resources have explicitly defined CRI addresses and ordinal numbers, and if they do not, will allocate CRIs and ordinal's for them in the same way as it did previously. This is required so that the CRIs and ordinal's that are allocated to the non-OCTM communication resources remain the same as they were prior to the OCTM migration.

Initial Usage of OCTM

After the OCTM migration is complete, the CEUS can be used to start adding and updating communication resources. Select some specific resources that require updating so that you can verify that OCTM is functioning correctly on your ALCS production system. Within a week after the OCTM migration has completed, begin the process of condensing the communication generation decks (see “Condensing the Communication Generation Decks” on page 231). There are important reasons for commencing this condensing process as soon as possible. Until the condensing function has been completed, communication resources must `not be deleted` from either the OCTM database or the communications generation. Also, non-terminal resources in the communication generation must not be added or modified.

Condensing the Communication Generation Decks

The terminal and X.25 PVC resources that are defined in the offline communications generation but which are now managed by OCTM should be removed from the communication generation base and update decks. This is not required immediately after OCTM has become fully operational, but it should be done within one week after that. Most ALCS systems have a large terminal network connected to them, therefore they require multiple communication generation decks to accommodate all the `COMDFLT`/`COMDEF` macros for the terminal resources. When the communication resources managed by OCTM are removed from the generation decks, the remaining communication resources could be combined together into a single generation deck (these are the X.25 PVC types 2 to 5, the APPC connections, the ALCI LU's, etc.).

Although the OCTM facility provides significant benefits in the management of the communication resources, it does though split the communication resources between the communications generation and the OCTM database. This split required the introduction of additional parameters on the communications generation `COMGEN` macro for managing the `CRI` addresses and the resource ordinal

numbers. These are the **CRIRANGE**= and **ORDRANGE**= parameters. Immediately after the OCTM migration is complete, there will be no conflict in the usage of CRIs and ordinals because the OCTM database had been built from the communication generation load module(s). When the CEUS starts adding new terminal resources it can request OCTM to give specific CRI addresses and ordinal numbers to the new resources. In parallel with this, new COMDEF's could be added to the communication generation (for example, a TCP/IP server resource) and use the same CRI addresses and ordinals as those used by OCTM. The COMGEN **CRIRANGE**= and **ORDRANGE**= parameters therefore ensure that this double usage of CRIs or ordinals does not occur. The OCTM facility will not allow any new terminal or X.25 PVC resources that are being added to the OCTM database to use the CRIs or ordinals that are in these ranges.

When the condensing of the communication generation occurs, special attention must be given to the CRI addresses and resource ordinal numbers of those resources still in the generation deck. The current CRIs and ordinal numbers can be retained, or new CRIs and ordinals can be allocated.

- Retaining the current CRIs and ordinals

  In the communications generation you can either explicitly define CRIs and ordinals or you can allow ALCS to allocate them. If you have allowed ALCS to allocate them, you must now change your COMDFLT/COMDEF macroinstructions and explicitly define the CRIs and ordinals. You can run the OCTM communications report program to find out which CRIs and ordinals are allocated to each resource.

  You can now optionally use the new parameters on the COMGEN macro (**CRIRANGE**= and **ORDRANGE**=) to reserve these CRIs and ordinals for the exclusive use of the communications generation and to also reserve additional CRIs and ordinals for future communication resources in the communication generation. These new COMGEN parameters allow 10 ranges of CRIs and ordinal numbers to be reserved. You could, for example, use up to 8 of these ranges to reserve current CRIs and ordinals and up to 2 of these ranges to reserve new groups of CRIs and ordinals (for future resources). Ensure that all the CRIs and ordinals used in the communication generation are defined in one of these ranges, otherwise you will receive warning messages when you run the communications generation.

  If you find it difficult to accommodate the current CRIs and ordinals within just 8 ranges, then we would recommend implementation of the alternative option (allocate new CRIs and ordinals) as described below.

- Allocate new CRIs and ordinals to current resources

  Identify a range of CRIs and ordinals that are not used by any communication resources (in the communication generation or the OCTM database). Define those ranges in the COMGEN **CRIRANGE**= and **ORDRANGE**= parameters. Allocate new CRIs and ordinals within those ranges to the communication resources in the condensed communication generation. Do this by coding the ICRI= and IORD= parameters on the COMDFLT macro (alternatively, code the CRI= and ORD= parameters on the COMDEF macro).

When you have chosen the appropriate solution for managing the CRIs and ordinals, implement the required changes in the communication generation deck and complete the condensing of the COMDFLT/COMDEF's. Run the communications generation stages 1 and 2, create the load module(s) and prepare a new communications load list. Run the OCTM communications report program to validate the contents of the load module(s).

Chapter 7. Customizing ALCS

This chapter describes the installation-wide monitor exits that are supplied with ALCS and how to use them to customize ALCS in your installation.

Note: This chapter contains Product-Sensitive Programming and Associated Guidance Information.

When you use the ALCS installation-wide monitor exits and the ALCS callable services you must ensure that all addresses and labels are valid. ALCS assumes that your customization code is written and fully tested before it is used.

Migrating customization from previous versions and releases

If you are migrating from previous versions and releases of ALCS, and you have customized your existing system, you may want to apply the same (or similar) customization to ALCS Version 2 Release 4.1. Your customization might include:

- User modifications to ALCS itself
- Installation-wide exit routines (user exits in ALCS/MVS/XA).

You may also have specialized programs that use undocumented or product-sensitive programming interfaces.

The ALCS installation-wide monitor exits

ALCS Version 2 Release 4.1 implements installation-wide monitor exits in a similar way to ALCS Version 2 Release 1.3, Release 2.1, and Release 3.1 but in a different way to ALCS/MVS/XA and ALCS Version 2 Release 1.1.

Entry points in DXCUSR and ALCS Version 2 Release 1.1

User exits in ALCS Version 2 Release 1.1 (and predecessor ALCS systems) are implemented through entry points into the installation-wide monitor exit routine DXCUSR. Figure 54 on page 233 shows a schematic representation of the ALCS Version 2 Release 1.1 installation-wide monitor exits.

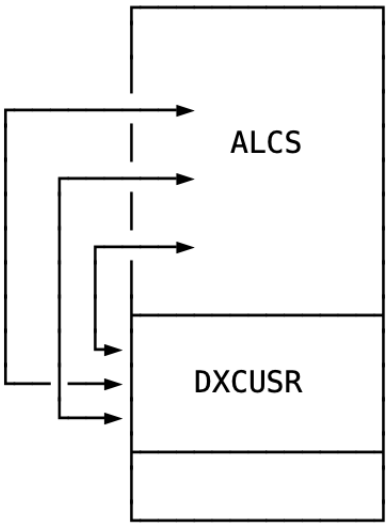


Figure 54. Installation-wide monitor exits in ALCS Version 2 Release 1.1

This method restricts the total size of the exit code to 4KB. It also means that to change an exit requires the following sequence:

© Copyright IBM Corp. 2003, 2019

- Upgrade the code
- Re-link the ALCS monitor
- Stop ALCS
- Start ALCS

Installation-wide monitor exits in ALCS Version 2 Release 4.1

In ALCS V2 Release 4.1, the installation-wide monitor exits are programs which can be loaded dynamically using the ALCS program loading mechanism (ZPCTL command with the U option). Different versions of the same exit can be activated at different times.

Figure 55 on page 234 shows a schematic representation of the ALCS installation-wide monitor exits.

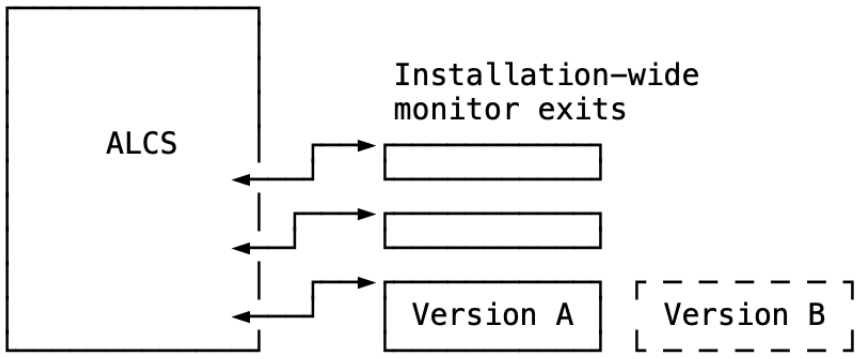


Figure 55. Installation-wide monitor exits in ALCS Version 2 Release 4.1

The load module for the installation-wide monitor exits:

- Can contain one or multiple installation-wide exit programs
- Must contain only installation-wide exits
- Must be in an authorized library

The source code can be maintained by the user's own library-maintenance system, or by using SMP/E. To load new load modules (for new or modified installation-wide exits) either update the program configuration table or use the ZPCTL LOAD,USER command. *ALCS Operation and Maintenance* describes the ZPCTL command.

“Updating the program configuration table” on page 213 describes how to update this table.

Invoking installation-wide monitor exits

Each ALCS installation-wide monitor exit is invoked by an MVS CALL macro. The figure below gives an overview of the calling process and register usage.

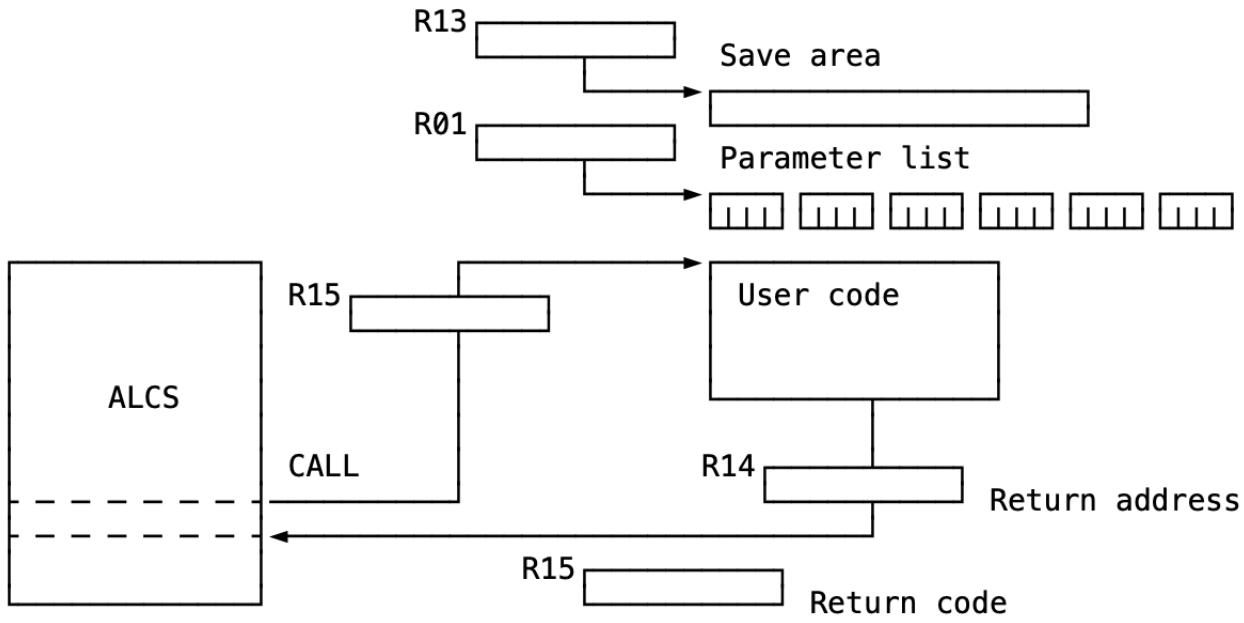


Figure 56. Installation-wide monitor exit linkage conventions

Note: Unless specifically mentioned, ALCS enters all installation-wide monitor exits in tables-storage PSW key.

Register usage on entry

ALCS sets the following entry conditions before calling any installation-wide monitor exit program:

R01
The address of a parameter list to pass to the user code

R11
Reserved

R12
The address of the information table. "ALCS Installation-wide monitor exit information table - IW0IT" on page 328 describes this table.

R13
Standard register save area address

R14
Return address

R15
The address of the entry point in the exit program.

The contents of general registers R00, and R02 through R10 are undefined.

Register usage by the user code

The user code can use the following general registers in the exit program:

- R00 through R10
- R14
- R15

The user code must save all registers (except the return-code register, R15) during the execution of the exit program.

Do **not** use general registers R11 or R12. Register R13 must point at the save area. Sample installation-wide monitor exit programs which are included with the ALCS product, contain code to save and restore the registers.

Register usage on return

The user code must restore all registers (except the return-code register, R15).

R15
The return code set by the user code.

Note: Unless specifically mentioned, all installation-wide monitor exits must return in tables-storage PSW key.

The sample installation-wide monitor exits

Each exit consists of a sample shell which contains:

- A header, with the name of the exit and the name of the entry point to the user code
- Standard register save code
- Standard register restore code
- A trailer which delimits the user code.

Figure 57 on page 236 gives an overview of the header and trailer in the sample exit programs.

```
          ┌  DXCUHDR          ┐ Header
          │ NAME=exit_name    │ Name of exit program
          │ ENTRY=label       │ Start of user code
          │                   │
          └                   ┘
    label xxxxxxxx              User code
          standard register save code
          xxxxxx
          xxxxxxxx
          ┌  DXCUEND          ┐ Trailer (must be the last instruction
          └                   ┘          before the END instruction)
```

Figure 57. Installation-wide monitor exits: The header and trailer

Installation-wide monitor exit header macro - DXCUHDR

Use DXCUHDR as the first instruction in each installation-wide monitor exit program.

Format

```
DXCUHDR NAME=name,ENTRY=label
        [,PRINT={NOGEN|GEN}]
        [,VERSION={00|vv}]
```

NAME=*name*
The name of the installation-wide monitor exit. “Implementing installation-wide monitor exits” on page 237 lists these names.

ENTRY=*label*
The name of the label where the executable code (or table) starts within the installation-wide monitor exit.

PRINT={NOGEN|GEN}
Print (GEN) or do not print (NOGEN or omitted) the macro expansion.

VERSION={00|*vv*}
The version of the installation-wide monitor exit. Must be a 2-byte decimal number. This is for user purposes only, ALCS does not use this field.

Description

The DXCUHDR macro is used to identify the installation-wide monitor exit to ALCS. It **must** be the first instruction in the installation-wide monitor exit. You should use the empty installation-wide monitor exit shell supplied with the product, which has the DXCUHDR macro already coded, wherever possible.

Installation-wide monitor exit trailer macro - DXCUEND

Use DXCUEND as the last instruction in each installation-wide monitor exit program.

Format

```
DXCUEND
```

Description

The DXCUEND macro is used to identify the end of the installation-wide monitor exit to ALCS. It must be the last instruction in the installation-wide monitor exit before the END statement. You should use the empty installation-wide monitor exit shell as supplied with the product, which has the DXCUEND macro already coded, wherever possible.

Implementing installation-wide monitor exits

This section describes installation-wide monitor exits in ALCS that allow the user to modify processing within ALCS. These exits operate as an extension to ALCS. By using these exits the user can tailor the system without modifying ALCS code.

Attention

Errors in, or misuse of, an exit can compromise the operation and integrity of ALCS, the ALCS database, MVS, and the processor. IBM cannot accept APARs for loss of integrity caused in this way.

You are strongly recommended to adopt testing procedures that guard against such loss of integrity; and also to use RACF (or a similar external security manager) to protect your installation against the effects of unauthorized usage.

APPC/MVS (LU 6.2) exit - USRAPPC

ALCS calls this exit when it processes an APPC/MVS request from an application program or from the ALCS monitor. Use this exit to:

- Verify that the originator has sufficient authority to use APPC/MVS, as the security environment for APPC/MVS is the security environment for the ALCS job or started task
- Indicate that the requesting entry should be terminated
- Identify an input or output message. This causes the count of input or output APPC messages to be incremented. ZSTAT MSG or ZSTAT ALL displays these counts. If data collection is currently collecting statistics about input and output messages, this also causes the statistics for input or output APPC messages to be incremented.

A sample installation-wide monitor exit is provided with the product in the installation-wide exit library, which shows how to identify input and output APPC messages. This is called DXCUAPPC. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRAPPC with the following conditions:

param_1
The address of the ECB (or zeros for monitor calls). Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of a 2-byte field to indicate the type of call:

0
Application program CPI-C call.

4
Application program APPC call.

8
Monitor call.

param_3

Restriction - Do not alter this address or field:

The address of request type area (see below).

For a CPI-C call (**param_2** is 0), **param_3** contains the address of a 1-byte request-type area. The value in this byte indicates the type of CPI-C call.

Value	CPI-C call	
1	CMINIT	Initialize_Conversation.
2	CMALLC	Allocate.
3	CMCFM	Confirm.
4	CMCFMD	Confirmed.
5	CMDEAL	Deallocate.
6	CMFLUS	Flush.
7	CMEMN	Extract_Mode_Name.
8	CMEPLN	Extract_Partner_LU_Name.
9	CMESL	Extract_Sync_Level.
10	CMACCP	Accept_Conversation.
11	CMECT	Extract_Conversation_Type.
12	CMPTR	Prepare_To_Receive.
13	CMRCV	Receive.
14	CMRTS	Request_To_Send.
15	CMSEND	Send_Data.
16	CMSERR	Send_Error.
17	CMSCT	Set_Conversation.
18	CMSDT	Set_Deallocate_Type.
19	CMSED	Set_Error_Direction.
20	CMSF	Set_Fill.
21	CMSLD	Set_Log_Data.

Value	CPI-C call	
22	CMSMN	Set_Mode_Name.
23	CMSPLN	Set_Partner_Lu_Name.
24	CMSPTR	Set_Prepare_To_Receive_Type.
25	CMSRT	Set_Receive_Type.
26	CMSRC	Set_Return_Control.
27	CMSST	Set_Send_Type.
28	CMSSL	Set_Sync_Level.
29	CMSTPN	Set_TP_Name.
30	CMTRTS	Set_Test_Request_To_Send_Received.
31	CMECS	Extract_Conversation_State.

For an APPC or monitor call (**param_2** is 4 or **param_2** is 8) param_3 contains the address of a 1-byte request-type area. The value in this byte indicates the type of APPC or monitor call.

Value	APPC call	
1	ATBALLC	Allocate.
2	ATBCFM	Confirm.
3	ATBCFMD	Confirmed.
4	ATBDEAL	Deallocate.
5	ATBFLUS	Flush.
6	ATBGETA	Get_Attributes.
7	ATBGETC	Get_Conversation.
8	ATBGETP	Get_TP_Properties.
9	ATBGETT	Get_Type.
10	ATBPTR	Prepare_To_Receive.
11	ATBRCVI	Receive_Immediate.
12	ATBRCVW	Receive_And_Wait.
13	ATBRTS	Request_To_Send.
14	ATBSEND	Send_Data.
15	ATBSERR	Send_Error.
23	ATBALC2	Allocate
24	ATBGTA2	Get_attributes
FD	ATAWAIT	Wait.

param_4

Restriction - Do not alter this address or field:

The address of the APPC/MVS data area. Refer to the DXCAPPCA description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

“ALCS ECB descriptor - DXCECBD” on page 321 explains how to access the ECB descriptor. The following ECB descriptor fields are particularly relevant to this installation-wide exit. They contain:

CE3TPIDS
B'00000001' TP_ID in use.

B'00000010' inbound conversation.

CE3TPID
TP_ID.

CE3TOKL
Length of user security token.

CE3TOK
User security token.

CE3PLUNM
Partner LU name.

CE3TPPRF
TP profile (ALCS application name).

All registers (except R15) must be the same as at entry. Do not use the CPDMP macroinstruction in this exit.

ALCS tests the following return conditions from USRAPPC:

R15=0
Continue processing.

R15=4
Return to caller. The originating terminal is not authorized for this request.

R15=8
Terminate the ECB.

R15=12
Increment the input message count and continue processing.

R15=16
Increment the output message count and continue processing.

Note that this exit can be called more than once for the same request.

CDS load list loading exit - USRCDSA

ALCS calls this exit during the processing of the ZPCTL LOAD,LIST and ZACOM LOAD,LIST commands. The ZPCTL command loads an **alternate** program configuration table (load list) onto configuration data set 1 and the ZACOM command loads an **alternate** communication configuration load list onto configuration data set 2. This exit is activated for each load module that is in the **loaded** or **confirmed** status in the **current** load list on the CDS. Use this exit to copy the name of each load module that is in the **loaded** or **confirmed** status (in the **current** load list) to the **alternate** load list.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUCDSA. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRCDSA with the following conditions:

param_1
The address of a 4-byte field containing the name of the CDS upon which the alternate load list is being loaded:

CDS1
program load list

CDS2
communications load list

param_2
The address of an 8-byte field containing the load module name.

param_3
The address of a 1-byte field containing the status of the load module:

Value	Status of load module
1	module is loaded
2	module is confirmed

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRCDSA:

R15=0
Do not copy load module. (This is also the default action that ALCS takes if the exit is not loaded).

R15=4
Copy load module.

CDS load list backout exit - USRCDSB

ALCS calls this exit during the processing of the `ZPCTL BACKOUT,LIST` and `ZACOM BACKOUT,LIST` commands. The `ZPCTL` command backs out an **alternate** program configuration table (load list) from configuration data set 1 and the `ZACOM` command backs out an **alternate** communication configuration load list from configuration data set 2. This exit is activated once during the processing of these commands. Use this exit to allow the backout of the **alternate** load list to be performed even if additional load modules have been loaded or confirmed since the **alternate** load list was loaded.

After an **alternate** load list has been loaded and confirmed (on CDS1 or CDS2) it must be verified during the next restart of the ALCS system. Once the verification has been completed, it can be committed or backed out. When an **alternate** load list has been loaded and is between the confirmed and committed stages, additional load modules can be loaded (`ZPCTL` or `ZACOM`) and confirmed. The names of these additional load modules are included in both the **current** and the **alternate** load lists. If the verification of the **alternate** load list fails and it has to be backed out, ALCS will require a backout of each additional load module (`ZPCTL` or `ZACOM`) before it will accept a backout request for the **alternate** load list. The USRCDSB monitor exit will allow a backout of the **alternate** load list to proceed even if additional load modules have been loaded or confirmed since the **alternate** load list was loaded.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called `DXCUCDSB`. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRCDSB with the following conditions:

param_1
The address of a 4-byte field containing the name of the CDS from which the alternate load list is being backed out:

CDS1	program load list
CDS2	communications load list

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRCDSB:

R15=0
Do not allow backout of the **alternate** load list if additional load modules are loaded or confirmed on the **alternate** load list. (This is also the default action that ALCS takes if the exit is not loaded).

R15=4
Do allow backout of the **alternate** load list even if additional load modules are loaded or confirmed on the **alternate** load list.

IBM 3270 display screen formatting exit - USRCOMA

This exit allows the user to modify or discard an IBM 3270 display format output message. ALCS calls this exit whenever it builds a format message:

- When an ECB-controlled program issues a COMCC monitor-request macro with the FORMAT parameter to format an IBM 3270 display.
- When an IBM 3270 display terminal establishes a session with ALCS, ALCS sends it a format message (known as a welcome screen) which includes a user-defined logon response and the IBM copyright notice. The user-defined logon response is sometimes known as the "Good Morning" message.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUCOMA. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRCOMA with the following conditions:

param_1
The address of the ECB if USRCOMA is called during COMCC monitor-request macro processing, otherwise zero. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of the ECB.

param_2

Restriction - Do not alter this address or field:

You can alter the **user area** of this communication-table item.

The address of the communication table entry for the destination resource. Refer to the C0ORE description in "Intended interfaces for installation-wide monitor exits" on page 308 details of the layout of this area.

param_3
The address of an IOCB used as the format message block.

param_4
The address of the current location in the format message block. This is the location immediately following any 3270 data stream characters that ALCS has already placed in the format message block. You can change the data in the message block.

When USRCOMA is called for the user-defined logon response message, the format message block already contains a 7-byte 3270 data stream write command sequence followed by the ALCS user-defined logon response message text to display at the top of the screen.

When USRCOMA is called for COMCC FORMAT=TEXTT, the format message block already contains a 7-byte 3270 data stream write command sequence followed by the user text to display at the top of the screen.

When USRCOMA is called for any other COMCC, the format message block already contains a 2-byte 3270 data stream write command sequence.

Refer to the *IBM 3270 Information Display System: Data Stream Programmer's Reference* manual for more information about the 3270 data stream.

param_5
The address of a 2-byte field which contains the number of display lines already included in the format message. You can store a new value in this field. For example, you can set this field equal to the total number of lines on the screen in order to suppress any user text for the bottom of the screen.

param_6
The address of a 2-byte field which contains the number of display lines to be formatted, or zero. This is from the COMCC ROWS= parameter. You can store a new value in this field.

param_7
The address of a 2-byte field which contains the number of display columns to be formatted, or zero. This is from the COMCC COLS= parameter. You can store a new value in this field.

param_8
The address of a 1-byte field of display format options to use. A bit mask indicates the options as follows: You can store a new value in this field.

Bit
Meaning

1
ON (1) to clear the screen

2
ON (1) to set the unsolicited message indicator on

3
ON (1) to set the unsolicited message indicator off

4
ON (1) to display text at the top of screen

5
ON (1) to display text at the bottom of screen

6
ON (1) to unlock the keyboard

param_9
The address of the user text to display at the bottom of the screen, or zero. This is from the COMCC TEXTB= parameter. The first 2 bytes of the data are the count field (the length of text excluding the 2-byte count field). The storage immediately following this count contains the text to be displayed. You can change the value in **param_9** to point to your own user text.

Note: If you store a new value in this parameter and it was previously zero, you must also set bit 5 ON in the field where **param_8** points.

param_10
The address of the user text to display at the top of the screen, or zero. This is from the COMCC TEXTT= parameter. The first 2 bytes of the data are the count field (the length of text excluding the 2-byte count field). The storage immediately following this count contains the text to be displayed.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRCOMA:

R15=0
Continue processing.

R15<>0
Abandon processing and discard the format message data.

Communication VTAM RECEIVE exit - USRCOM0

ALCS calls this exit for the following incoming data-flow-control requests from VTAM, immediately after it completes the receive processing:

LUSTAT
RTR
SIGNAL

ALCS normally processes LUSTAT and RTR requests from 3270 printer terminals. You can use this exit to process LUSTAT or RTR requests from other devices if required.

ALCS does not normally process SIGNAL requests. You can use this exit to process SIGNAL requests if required.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUCOM0. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRCOM0 with the following conditions:

param_1
The address of the ECB for LUSTAT or RTR, or zero for SIGNAL.

ALCS normally discards this ECB.

Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this field:

The address of the VTAM RPL. Refer to the VTAM macro IFGRPL for details of the layout of this area.

param_3

Restriction - Do not alter this field:

You can alter the **user area** of this communication-table item.

The address of communication table entry for the originating resource, one of:

VTAM 3270 display terminal
VTAM 3270 printer
ALCI logical unit (LU)
X.25 permanent virtual circuit (PVC)
WTTY link

Refer to the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_4
The address of a 2-byte field containing the type of data-flow-control request:

0
LUSTAT

4
RTR

8
SIGNAL

All registers (except R15) must be the same as at entry. ALCS test the following return conditions from USRCOM0:

R15=0
Continue processing.

R15=4
The data is completely processed by the exit. Bypass any further processing and discard the ECB (if there is one).

R15=8
The data is completely processed by the exit. Bypass any further processing but do not discard the ECB (if there is one). Your exit must use either the UECBQUE callable service to queue the ECB on a work list or the UECBREL callable service to release it.

Communication VTAM RECEIVE exit - USRCOM1

ALCS calls this exit for all incoming messages from VTAM immediately after it completes the receive processing. No ALCS processing or reformatting is done on the message. You can use this exit to validate and (or) change incoming messages.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUCOM1. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRCOM1 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2
The address of the receive buffer. Refer to the CM5CM description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_3

Restriction - Do not alter this address or field:

You can alter the **user area** of this communication-table item.

The address of the resource entry in the communication table. Refer to the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRCOM1:

R15=0
Continue processing.

R15=4
Bypass any further processing and queue the ECB on the input list. The data is completely processed by the exit. The post-interrupt address must be set in field CE3PIA in the ECB descriptor. "ALCS ECB descriptor - DXCECBD" on page 321 explains how to access the ECB descriptor. Refer to the DXCECBD description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

Before calling USRCOM1, ALCS stores a default post-interrupt address in field CE3PIA. It is the address of routine HSOPZRO in monitor CSECT DXCOPZ.

R15=8
Discard the input data.

Communication input message exit - USRCOM2

ALCS calls this exit for every input message, just before it passes control to an application or to the ALCS command processor. Use this exit to:

- Add to the normal input message processing
- Discard the message.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUCOM2. IBM advises that you use this as the basis for your installation-wide exit.

You can use this exit to detect abnormally high input message rates from a terminal, and reduce the number of input messages which are passed to an application. The communication resource data includes fields RETDMTC, RETDMTD, and RETDMTT which USRCOM2 can refer to or modify for this purpose. (See the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for a description of the communication resource data.)

ALCS enters USRCOM2 with the following conditions:

EBROUT
Contains the CRI of the originating terminal.

D0
Either the input message is in a block attached to level D0 (when the message is in the standard format), or data level D0 contains the address of a heap storage area containing the input message (when the message is in the extended format).

When the message is in the extended format then you must use the 4-byte field CM1CCX instead of the 2-byte field CM1CCT if you inspect the message block.

Example:

```
          CLI   CE1CT0+1,1            STORAGE BLOCK ON D0
          BE    UCOM2EXT              BRANCH IF NOT
          SPACE 1
UCOM2STD  DC    0H'0'                 ** STANDARD FORMAT **
          L     R06,CE1CR0            LOAD MESSAGE ADDRESS
          B     UCOM2GO               GO TO PROCESS MESSAGE
UCOM2EXT  DC    0H'0'                 ** EXTENDED FORMAT **
          L     R06,CE1FA0            LOAD MESSAGE ADDRESS
          B     UCOM2GO               GO TO PROCESS MESSAGE
```

param_1
The address of the ECB. Refer to the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

You can alter the **user area** of this communication-table item.

The address of the communication table entry for the resource. Refer to the C00RE description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_3

Restriction - Do not alter this address or field:

The address of a 4-byte field which contains the name of the input edit program for the application or the ALCS command processor.

param_4

Restriction - Do not alter this address or field:

You can alter the **user area** of this communication-table item.

The address of the communication table entry for the input edit program. Refer to the C00RE description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_4 is zero if:

- The input message is to be passed to the ALCS command processor
- An error prevents it from being passed to an application. Errors include:
  - The originating terminal is not routed to an application
  - The application name is not defined to ALCS
  - The application is not currently active
  - The application is restricted by the current system state.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRCOM2:

R15=0
Continue processing.

R15=4
Discard the input message and the ECB.

Communication VTAM SEND exit - USRCOM3

ALCS calls this exit for all outgoing messages from VTAM immediately before passing the message to VTAM. No further ALCS processing or reformatting is done on the message. You can use this exit to validate and (or) change outgoing messages.

ALCS may delay the message transmission if the LU is busy.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUCOM3. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRCOM3 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of the VTAM routing parameter list.

param_3

Restriction - Do not alter this address or field:

You can alter the **user area** of this communication-table item.

The address of the communication table entry for the resource. Refer to the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRCOM3:

R15=0
Continue processing as normal, send the data.

R15=4
Bypass sending the data.

Communication output message exit - USRCOM4

ALCS calls this exit for every output message, at the start of R0UTC or SEND-type monitor-request macro processing. Use this exit to:

- Add to the normal output message processing
- Discard the message.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUCOM4. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRCOM4 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of the R0UTC or SEND-type monitor-request macro parameters.

param_3

Restriction - Do not alter this address or field:

You can alter the **user area** of this communication-table item.

The address of the communication table entry for the destination resource. Refer to the C0ORE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_4

The address of a 2-byte field containing the level index where the output message block or heap storage area is attached to the ECB. Use the same conventions as the GETCC monitor request macro. Either the output message is in a block attached to this storage level (when the message is in the standard format), or this data level contains the address of a heap storage area containing the output message (when the message is in the extended format).

When the message is in the extended format then you must use the 4-byte field CM1CCX instead of the two-byte field CM1CCT, if you inspect the message block. This extended format is used when large messages are sent to an ALC terminal that connects to ALCS through TCP/IP or to a TCP/IP connection.

Example:

```
          L     R02,PARM4               LOAD PARAMETER
          LH    R02,0(,R02)             LOAD LEVEL INDEX
          AR    R02,R09                 POINT TO LEVEL
          CLI   CE1CT0+1-EB0EB(R02),1   STORAGE BLOCK ON Dn
          BE    UCOM4EXT                BRANCH IF NOT
          SPACE 1
UCOM4STD  DC    0H'0'                   ** STANDARD FORMAT **
          L     R06,CE1CR0-EB0EB(,R02)  LOAD MESSAGE ADDRESS
          B     UCOM4GO                 GO TO PROCESS MESSAGE
UCOM4EXT  DC    0H'0'                   ** EXTENDED FORMAT **
          L     R06,CE1FA0-EB0EB(,R02)  LOAD MESSAGE ADDRESS
          B     UCOM4GO                 GO TO PROCESS MESSAGE
```

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRCOM4:

R15=0

Continue processing.

R15=4

The output message has been completely processed by this exit.

R15=8

Retry the operation. ALCS returns control to the ECB-controlled program that issued the ROUTC or SEND-type monitor-request macro.

Communication COMCC macro exit - USRCOM5

The COMCC monitor-request macro service routine uses this exit when a specified field name is not found in the ALCS tables. There is no executable code in this exit, it is a table of data.

An empty installation-wide monitor exit shell, DXCUCOM5, is provided with the product in the installation-wide exit library. IBM advises that you use this as the basis for your installation-wide exit.

The USRCOM5 table contains the names and characteristics of user-defined fields in the communication tables. Define fields here so that COMCC can update fields in the user part of a communication table entry.

Use the following format for each field:

```
         DC    CL8'field_name'
         DC    AL2(DSECTlabel-DSECTstart)
         DC    BL1'authority'
```

```
         DC    BL1'DSECTtype'
         DC    AL1(length)
         DC    BL1'bit_mask'
         DC    AL2(0)                RESERVED
```

Where:

field_name
The name specified on the FIELD parameter of the COMCC monitor-request macro.

DSECTlabel
The name of the field in the C0ORE DSECT, or in your own DSECT that describes the user part of a communication table entry.

DSECTstart
The name of DSECT containing *DSECTlabel*.

authority
Authority required by the originator of the ECB to make the update. An 8-bit field with a bit set ON to indicate the origin of the ECB as follows:

Bit
Entry authorization

0
System

1
Prime CRAS

2
Alternate CRAS

4
Unrestricted

5
The same CRI as the communication table entry

Bits 3, 6 and 7 are reserved and must be B'0'.

DSECTtype
Indicates whether *DSECTstart* is the start of the system part or user part of the communication table entry. An 8-bit field with a bit set ON to indicate the type of DSECT as follows:

Bit
Type of DSECT

7
DSECTstart is the start of the user part of the communication table entry.

Bits 0 through 6 are reserved and must be B'0'.

length
The length of the field, one of:

Zero, for 1 bit in a field
The number of bytes in the field.

bit_mask
Isolates 1 bit within the byte that is identified by *DSECTlabel*.

Example 1

The bit FRED is defined within the CO0RE DSECT:

```
FRED     DC    0BL(B'00010000')'0'
```

entry coding:

```
         DC    CL8'FRED'             FIELD NAME
         DC    AL2(FRED-CO0RE)       FIELD DISPLACEMENT
         DC    BL1'authority'
         DC    BL1'0'                FIELD IS IN CO0RE
         DC    AL1(0)                LENGTH 0 (BIT FIELD)
         DC    AL1(L'FRED)           BIT MASK
         DC    AL2(0)                RESERVED
```

Example 2

The CMUSR DSECT describes the user data in a communication table entry, and field CMLINK is defined within the CMUSR DSECT:

```
CMLINK DC CL7' '
```

entry coding:

```
         DC CL8'CMLINK'              FIELD NAME
         DC AL2(CMLINK-CMUSR)        FIELD DISPLACEMENT
         DC BL1'authority'
         DC BL1'1'                   FIELD IN USER AREA
         DC AL1(7)                   LENGTH 7
         DC AL1(0)
         DC AL2(0)                   RESERVED
```

Mark the end of the table with 8 bytes of X'FF' as follows:

```
         DC    X'FFFFFFFFFFFFFFFF'            END OF TABLE MARKER
```

Note: You can also define fields in the system part of a communication table entry to override the COMCC restrictions used by ALCS, but use this option with care as it may prevent correct operation of ALCS communications.

Communication logon exit - USRCOM6

This exit allows the user to accept or reject a VTAM logon request. Use this exit, for example, to verify that the LU has sufficient authority to use ALCS.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUCOM6. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRCOM6, in key zero and supervisor state, with the following conditions:

R01

Restriction - Do not alter this address or field:

The address of the VTAM logon-exit parameter list.

This exit must return in key zero and supervisor state. Do not use the CPDMP macroinstruction in this exit.

All registers (except R15) must be the same as at entry.

Refer to the *VTAM Programming* manual for more information on using exit routines.

ALCS tests the following return conditions from USRCOM6:

R15=0
Continue processing.

R15=4
Data has been processed by this exit, return to VTAM.

R15=8
Reject the logon request.

Communication SCIP exit - USRCOM7

This exit allows the user to accept or reject a VTAM logon request for an LU 6.1 link. Use this exit, for example, to verify that the LU has sufficient authority to use ALCS.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUCOM7. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRCOM7, in key zero and supervisor state, with the following conditions:

R01

Restriction - Do not alter this address or field:

The address of the VTAM SCIP exit parameter list.

This exit must return in key zero and supervisor state. Do not use the CPDMP macroinstruction in this exit.

All registers (except R15) must be the same as at entry.

Refer to the *VTAM Programming*, SC31-6409 manual for more information on using exit routines.

ALCS tests the following return conditions from USRCOM7:

R15=0
Continue processing.

R15=4
Data has been processed by this exit, return to VTAM.

R15=8
Reject the logon request.

ALC acknowledgement exit - USRCOM8

This exit allows the user to take any additional actions required when ALCS receives a message from an ALC printer or a printer defined with TERM=3270PRT connected through MQ (LDTYPE=MQTERM).

ALCS calls this exit when it receives a message from the printer before it checks whether the message is an acknowledgement.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUCOM8. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRCOM8 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2
The address of a message block. Refer to the CM1CM description in *ALCS Application Programming Reference - Assembler* for details of the layout of this area.

param_3

Restriction - Do not alter this address or field:

You can alter the **user area** of this communication-table item.

The address of the communication table entry for the originating resource. Refer to the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRCOM8:

R15=0
Continue processing.

R15=4
Bypass acknowledgement checks (accept as ACK).

R15=8
Discard message and release ECB.

DASD load mismatch exit - USRDAS1

ALCS calls this exit when an attempt is made to load a DASD generation table, (with the ZDASD LOAD command) which does not match the one currently loaded. A new table will normally match the old table except for any new items which must be at the end. You can use this exit to decide whether to abort the load of to continue loading.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUDAS1. IBM advises that you use this shell as the basis for your installation-wide exit.

ALCS enters with the following conditions:

R01
The address of the item in the DASD generation table that is already loaded (current table).

R02
The address of the equivalent item in the DASD generation table being loaded (new table).

On return to ALCS all registers except R15 must be the same as on entry.

ALCS tests the following return condition from USRDAS1:

R15=0
The item being loaded is acceptable, despite the mismatch. Accept the new item and continue with the load.

R15=4
There is a genuine mismatch. Abort the load.

This exit can assist in resolving problems that may occur when a DASD generation table is loaded. You should not use this exit unless a problem occurs. Use the exit to resolve a specific problem that has occurred with the new DASD generation table and after successful completion of the load, remove (ZPCTL UNLOAD) this exit.

ALCS activates this exit while it is validating the new DASD generation table. The DASD generation table is comprised of 16-byte DFDTB data items, the format of which is defined in the DXCDBU Data Base Update Table macro. Each item contains information for a specific file type. ALCS activates this exit for each item that has a mismatch, therefore the exit may be activated multiple times during the DASD generation table load.

If the load fails because the VFA options are not identical (in the current and new tables), use this exit to allow the load to complete if the only difference between the current and new tables is the VFA options. The installation-wide monitor exit shell provided with the ALCS product contains sample code that will allow a ZDASD LOAD to complete when there is a mismatch in the VFA options.

Data collection exit - USRDCR1

ALCS calls this exit when the online data collection facility collects data about an event. Use this exit to modify the data being collected, or to append your own data.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUDCR1. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRDCR1 with the following conditions:

param_1
The address of a 1-byte data collection event type.

Value	Data collection event
1	ECB exit
2	Input message (3270, NetView, ALCI, BSC, WTTY)
3	Output message (3270, NetView, ALCI, BSC, WTTY)
4	Program enter
5	Real DASD I/O
6-10	(reserved)
11	CPU loop wait
12	(reserved)
13	Input message - SLC
14	Output message - SLC
15	VFA Find/File
16	Input message - X.25
17	Output message - X.25
18	Input message - APPC
19	Output message - APPC
20	Input message - TCP/IP
21	Output message - TCP/IP
22	Input message - LU6.1
23	Output message - LU6.1
24	Output message - 3270 format
25	Application call - APPC input message
26	Application call - APPC output message
27	Application call - TCP/IP input message
28	Application call - TCP/IP output message
29	Application call - MQ input message
30	Application call - MQ output message
31	Input message - MQ
32	Output message - MQ
33	Input message - WAS
34	Output message - WAS
35	Application call - WAS input message
36	Application call - WAS output message

Note: If **param_1** is 20 (Input message TCP/IP) or 21 (Output message TCP/IP) and you use TCP/IP large messages, then you must use the 4-byte field CM1CCX instead of the 2-byte field CM1CCT if you

inspect the message block (refer to CM1CM in *ALCS Application Programming Reference - Assembler* for a description of the message format). The message block for TCP/IP messages can be either in the extended or in the standard format. When in standard format it is guaranteed that the two characters preceding CM1CCT are binary zeros.

param_2
The address of a 4-byte field containing the address of the data collection logging record mapped by DCLOG.

param_3
The address of a 4-byte field containing the length of the data collection logging record mapped by DCLOG.

param_4
The address of a 4-byte field containing the address of the data collection user area (or zero if there is none).

param_5
Output parameter to be set by the exit. On entry, this is zero. On return, this is the address of a 4-byte field containing the length of the data collection user area that is utilised by this exit.

param_6 to param_9
Parameters specific to the data collection event.

param_1	**param_6 to param_9**	
1	param_6	Address of 4-byte field containing address of ECB
2	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry
3	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry
4	param_6	Address of 4-byte field containing address of ECB
	param_7	Address of 4-byte field containing address of program name table entry for the called program
5	param_6	Address of 4-byte field containing address of IOCB
11	param_6	Address of 4-byte field containing address of 2 consecutive doublewords containing the TOD before wait and the TOD after wait
13	param_6	Address of 4-byte field containing data length
	param_7	Address of 4-byte field containing address of data
	param_8	Address of 4-byte field containing address of link keypoint

param_1	param_6 to param_9	
	param_9	Address of 4-byte field containing address of channel keypoint
14	param_6	Address of 4-byte field containing data length
	param_7	Address of 4-byte field containing address of data
	param_8	Address of 4-byte field containing address of link keypoint
	param_9	Address of 4-byte field containing address of channel keypoint
15	param_6	Address of 4-byte field containing address of VFA buffer header
16	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry
17	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry
18	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry
19	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry
20	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry
21	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry
22	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry

param_1	param_6 to param_9	
23	param_6	Address of 4-byte field containing data length
	param_7	Address of 4-byte field containing address of data
	param_8	Address of 4-byte field containing address of comms table entry
24	param_6	Address of 4-byte field containing data length
	param_7	Address of 4-byte field containing address of data
	param_8	Address of 4-byte field containing address of comms table entry
25	param_6	Address of 4-byte field containing address of ECB
26	param_6	Address of 4-byte field containing address of ECB
27	param_6	Address of 4-byte field containing address of ECB
28	param_6	Address of 4-byte field containing address of ECB
29	param_6	Address of 4-byte field containing address of ECB
30	param_6	Address of 4-byte field containing address of ECB
31	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry
32	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry
33	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry
34	param_6	Address of 4-byte field containing address of CM1CM format message block
	param_7	Address of 4-byte field containing address of comms table entry

param_1	param_6 to param_9	
35	param_6	Address of 4-byte field containing address of ECB
36	param_6	Address of 4-byte field containing address of ECB

All registers must be the same as at entry.

Data collection exit - USRDCR2

Application programs use monitor-request macro DCLAC to update the ECB user data collection area. The length of this area is specified by parameter DCLECBU on the system generation macro SCTGEN. A user data collection area of this size is allocated in each entry immediately following the system data collection area.

ALCS calls this exit when an ECB exits and the online data collection facility is collecting statistics about ECBs. Use this exit to return the actual length of the ECB user data collection area to be written, from zero up to the maximum specified by the SCTGEN DCLECBU parameter.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUDCR2. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRDCR2 in table storage PSW key with the following conditions:

param_1
The address of a 1-byte field containing the data collection options.

Bit	Option
0	Input/Ouput messages
1	SLC traffic
2	ECB
3	Program use
4	CPU utilization
5	Input/Output operations
6	ALCS task waits

param_2
The address of the ECB. Refer to the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_3
The address of the start of the ECB user data collection area.

param_4
Address of a 2-byte field containing the length of the ECB user data collection area.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRDCR2:

R15=0
Use installation-defined data length (as at entry).

R15¬=0
Use this value for user data length.

Data collection exit - USRDCR3

ALCS calls this exit when data collection is active, before writing each record to the data collection sequential file. Use this exit to write the record to a different location, or to prevent the record from being written.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUDCR3. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRDCR3 in table storage PSW key with the following conditions:

param_1
The address of a 1-byte field containing the data collection options.

Bit	Option
0	Input/Ouput messages
1	SLC traffic
2	ECB
3	Program use
4	CPU utilization
5	Input/Output operations
6	ALCS task waits

param_2
Address of a 4-byte field containing the address of the data collection logging record mapped by DCLOG.

param_3
Address of a 4-byte field containing the length of the data collection logging record mapped by DCLOG.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRDCR3:

R15=0
Write the record to the data collection sequential file.

R15¬=0
Do not write the record. Set this return condition if this exit writes the record to a different location, or if you want to prevent the record from being written.

Date and time formatting - USRDFMT

This exit allows you to resolve time and date formats which the ALCS TIMEC macro is not designed to cope with. This exit can be used, for example, in areas where there are more than twelve months in the local calendar. If this exit is not loaded ALCS attempts to resolve the date and time request using its own date and time token characters (see *ALCS Application Programming Reference - Assembler* for a description of the TIMEC macro).

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUDFMT. IBM advises that you use this as the basis for your installation-wide exit.

ALCS will pass control to this exit if a TIMEC macro was issued with the parameter:
FORMAT=INSTALLATION, ALCS uses the installation default date and/or time format provided at system generation time.
FORMAT=CUSTOM, the user of the TIMEC macro provides the format to use (CUSTOM) ALCS enters USRDFMT in entry or tables storage key with the following conditions:

param_1

Date to be converted. This is a fullword in packed decimal format as follows: DDMMYYYY, where:

DD
Day in month

MM
Month in year

YYYY
Year

param_2

Time to be converted. This is a fullword in packed decimal format as follows: hhmmss, where:

hh
Hours (24 hour clock)

mm
Minutes

ss
Seconds

param_3

The address of the target area. The target area comprises a 2-byte length field immediately followed by a format string. The length is the maximum length of the string that your routine can return. Your routine must store the return string immediately following this length field and replace the length field contents with the length of the string. Your routine must not store past the target area.

For example, if your exit routine is called with **param_3** pointing to:

```
          DC   AL2(17),C'DD LLLLLLLLL YYYY'
```

Then your routine might modify this to:

```
          DC   AL2(15),C'03 January 1997'
```

param_4

Restriction - Do not alter this address or field:

Address of month names table.

The month names table is a full-word containing the number of month names, immediately followed by one full-word for each month name - each full-word contains the address of a month name field. The month name field is a 2-byte binary length, immediately followed by the month name in characters. These are the names you specified in the MONTHNAME parameter of the SCTGEN generation macro.

For example, if you default MONTHNAME, then **param_4** contains the address of:

```
          DC   F'12'
          DC   A(M1,M2,M3,...)
⋮
M1        DC   AL2(7),C'January'
M2        DC   AL2(8),C'February'
⋮
```

param_5

Restriction - Do not alter this address or field:

Address of abbreviated month names table.

The abbreviated month names table is a full-word containing the number of month names, immediately followed by one full-word for each month name - each full-word contains the address of an abbreviated month name field. The abbreviated month name field is a 2-byte binary length, immediately followed by the abbreviated month name in characters. These are the names you specified in the MONTHABBR parameter of the SCTGEN generation macro.

For example, if you default MONTHNAME and MONTHABBR, then **param_5** contains the address of:

```
          DC   F'12'
          DC   A(A1,A2,A3,...)
⋮
A1        DC   AL2(3),C'Jan'
A2        DC   AL2(3),C'Feb'
⋮
```

All registers (except R15) must be the same as at entry. Do not use callable services in this exit. Do not change the storage protection key in this exit.

ALCS tests the following return codes from USRDFMT:

R15=0
Formatting completed without error.

R15<>0
No date and time formatting performed. Use ALCS formatting.

Dump extension exit - USRDMP

Use this exit to dump storage areas that ALCS does not normally include in system error dumps (for example, user tables).

Note: The system error routine does not call this exit if the system error occurs in another installation-wide monitor exit.

Do not use the CPDMP macro in this exit.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUDMP. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRDMP with the following conditions:

param_1
The address of the ECB (or zero). Refer to the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

Interrupt work area address. Refer to the IR0WA description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_3

Restriction - Do not alter this address or field:

The address of a 2-byte field which contains a branch vector:

0 on the first entry to this routine
4 on the second
8 on the third
and so on.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRDMP:

R15=0
No user area to dump.

R15=4
Dump the user area.

The user must set the following return parameters when R15=4:

param_4
The address of user area to dump.

param_5
The address of a 4-byte field which contains the length of the area to dump.

param_6
The address of a 4-byte field with the following contents:

Byte 0
Reserved (must be 0).

Byte 1
Reserved (must be 0).

Byte 2
ALCS diagnostic file processor option (see below).

Byte 3
User area number (see below).

This exit is called again when R15 is not 0. This allows several user areas to be dumped. To control the dumping sequence, **param_3** is set to:

0
on the first entry

4
on the second

8
on the third

and so on.

When **param_6** byte 2 is 0, the ALCS diagnostic file processor prints the area with the heading *user area number n*, where *n* is the value in **param_6** byte 3, printed in decimal.

When **param_6** byte 2 is 1, the ALCS diagnostic file processor expects two user areas with the same option and area number. It prints the first area as the heading for the second.

Attention If this exit returns with an address that is not valid (for example a fetch protected area) then ALCS can abend.

Change entry originator authorization exit - USREID

ALCS calls this exit when an application program issues a `GTFCC CONV,MODIFY` monitor-request macro to change the originating identity for an entry. ALCS normally allows the following changes:

- For an entry from a communication link (for example, LU 6.1 links) the new originator must be defined as an OSYS terminal on the other system that the originating link connects.
- For an entry from Prime CRAS, or an entry created by the ALCS system, the new originator must be defined as an OSYS terminal on any other system.

Use this exit to:

- Allow an application program to change the entry originator to any terminal (local or remote).
- Prevent a change that ALCS would normally allow.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called `DXCUEID`. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USREID with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2
The address of the communication table entry for the current entry originator. Refer to the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area. For an entry created by the ALCS system, this parameter is zero.

param_3
The address of the communication table entry for the new entry originator. Refer to the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_4
The address of a 4-byte field containing a hexadecimal number which indicates the default action that ALCS will take. One of:

0
ALCS will change the entry originator.

4
ALCS will not change the entry originator.

This is also the default action that ALCS takes if the exit is not loaded (the same values as the R15 return condition).

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USREID:

R15=0
Change the entry originator.

R15=4
Do not change the entry originator.

Third party conversational trace exit - USRENBK

ALCS calls this exit during conversational trace ENTER/BACK processing.

This exit is intended for use with third party software. For more information contact the ALCS Support Group (email alcs@uk.ibm.com).

EXITC macro service extension exit - USREXT

Use this exit to add extra processing, or checks (or both) when any ECB exits (EXITC monitor request macro, SERRC with exit, and so on).

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUEXT. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USREXT with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

All registers must be the same as at entry.

File address migration exit - USRFAR

Use this exit to support file address formats that are not supported by ALCS. When a database is migrated to ALCS, the file address format used on that database may be different to the ALCS band format file

address. Because file addresses are normally imbedded within records on the database, support must be provided through the USRFAR exit for the file address format.

The file address format may be able to coexist with the ALCS band format (this is the normal situation). It is also possible that the file address format cannot coexist (for example, the file address contains the band in byte 0 and the ordinal in bytes 1-3). There are therefore two methods provided in USRFAR for the support of the file address format.

1. When an application issues a monitor-request macro which includes a file address, if that file address is not in the ALCS band format, USRFAR can provide the ALCS file address processing routines with the record class, record type and record ordinal number for that record. This method is used when the file address format can coexist with the ALCS band format.
2. When an application requires the file address of a record (for example, the file address of a fixed file record) ALCS passes the record class, record type and record ordinal to USRFAR and the exit returns a file address to ALCS (which forwards the file address to the application).

   When an application issues a monitor-request macro which includes a file address, ALCS passes the file address to USRFAR and the exit returns to ALCS with the record class, record type and record ordinal number for that record. This method is used when the file address format cannot coexist with the ALCS band format.

Use only one of these methods. Identify the method that you will use by coding the appropriate parameter on the ALCS database generation macro DBGEN. Code FARMIG=USER when the file address format can coexist with the ALCS band format, and code FARMIG=USERCODE when the file address format cannot coexist with the ALCS band format.

Attention:

Please contact IBM for advice if you plan to implement this installation-wide exit.

The following describes the USRFAR entry and exit conditions for these two methods.

File address migration method where formats coexist

This method must provide the ALCS file address processing routines with the record class, record type and record ordinal number for the record. This method is used when FARMIG=USER has been coded on the ALCS DBGEN generation macro.

Within USRFAR, code DXCRID without parameters. DXCRID generates EQU instructions for the application fixed file record types. (DXCRID gets these record types from the user-written macro DXCURID.)

ALCS enters USRFAR in entry or tables storage PSW key with the following conditions:

R00
Contains the file address.

You can use the following registers without restoring them:

R00 through R07 and
R15

Note: USRFAR must not:

- Use parameters to pass input information.
- Use the ALCS callable services.
- Use the save area.
- Change the storage key.

The user code must set the following return conditions from USRFAR:

R15	Record class	R01	R00
0	Fixed file	Record type number (for example, the value of #WAARI)	Ordinal number

R15	Record class	R01	R00
4	Pool file	Record type number (0=L1ST, 1=L1LT, 2=L2ST and so on)	Ordinal number
8	General file	General file number	Ordinal number
12	Not valid	0	0

An empty installation-wide exit shell is provided with the product in the installation-wide exit library for this method. It is called DXCUFAR. IBM advises that you use this as the basis for your installation-wide exit when this method is used.

File address migration method where formats do not coexist

Use this method when it is not practical to convert all the imbedded file addresses in your migrated database to the ALCS band format. This method is used when FARMIG=USERCODE has been coded on the ALCS DBGEN generation macro. It is used for both encoding and decoding of file addresses.

When called for encoding file addresses, ALCS passes the record number, record type and record ordinal number of a record. When called for decoding file addresses, ALCS passes the file address of the record, and this exit must provide the record class, record type and record ordinal number.

The empty installation-wide monitor shell called DXCUFAR should not be used for implementation of encoding/decoding file addresses. Request the required shell from IBM.

Note: USRFAR must not:

- Use parameters to pass input information.
- Use the ALCS callable services.
- Use the save area.
- Change the storage key.

You can use the following registers without restoring them:

R00 through R04,
R06 through R07, and
R15

ALCS enters USRFAR in entry or tables storage PSW key with the following conditions:

For decoding file addresses:

R02
Contains the value F'8'.

You must test R02 contents to determine if USRFAR is called for encoding or decoding a file address.

R00
Contains the file address.

The user code must set the following return conditions from USRFAR:

R15
Record class

R01
Record type

R00
Record ordinal

Note: The precise contents of record class and record type may vary from one ALCS release to the next, or as a result of ALCS maintenance updates.

For encoding file addresses:

R02

Byte 1
Record class

Bytes 2-3
Record type

R01
Contains the record ordinal

Note: The precise contents of record class and record type may vary from one ALCS release to the next, or as a result of ALCS maintenance updates.

The user code must return the encoded file address in General Purpose Register 0 (R00).

Long-term pool dispense exit - USRGFS

Use this exit to override the normal criteria for dispensing a long-term pool file record.

The normal dispensing criteria is based upon the data in the long-term pool control fields in the record.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUGFS. IBM advises that you use this as the basis for your installation-wide exit.

You can use this exit, for example, to:

- Prevent re-use of a record that has been released, even if Recoup has run without finding any reference to it

  For example, if you want to preserve some record contents for a fixed time, regardless of whether the record has been released
- Allow immediate re-use of lost addresses
- Allow re-use of a record that has not been released, even if Recoup has found references to it.

Attention

ALCS provides a high level of long-term pool integrity, but at the same time it avoids retaining data that is no longer required. Incautious use of this installation-wide exit can cause loss of data from long-term pool or unnecessary wastage of long-term pool file records.

ALCS enters USRGFS with the following conditions:

param_1

Restriction - Do not alter this address or field:

The address of a 4-byte field containing the file address.

param_2

Restriction - Do not alter this address or field:

The address of the long-term pool control fields in the record. Refer to the RS0RS description in “Intended interfaces for installation-wide monitor exits” on page 308 "Interfaces for installation-wide monitor exits" for details of the layout of this area. This is the 31-bit address. You can not use this address if you specified SCTGEN AMODE64=VFA in your system configuration; use **param_6** instead.

param_3

Restriction - Do not alter this address or field:

The address of the long-term pool record. This is the 31-bit address of the VFA buffer. You can not use this address if you specified SCTGEN AMODE64=VFA in your system configuration; use **param_7** instead.

param_4

Restriction - Do not alter this address or field:

The address of a 2-byte field containing the status of the long-term pool file record. One of:

0
Record is valid and available.

4
Record is lost address.

8
Record was released after the last Recoup run.

This is also the default action that ALCS takes if the exit is not loaded (the same values as the R15 return condition).

param_5
Address of 2-byte field containing type of call.

The type of call will be one of the following binary values:

0
Normal dispense.

4
Dispense by the emergency pool recovery function.

param_6

Restriction - Do not alter this address or field:

The address of an 8-byte field containing the address of the long-term pool control fields in the record. Refer to the RS0RS description in “Intended interfaces for installation-wide monitor exits” on page 308 "Interfaces for installation-wide monitor exits" for details of the layout of this area.

You can use this parameter whether your VFA buffers are above the bar or not.

- When your VFA buffers are above the bar (`SCTGEN AMODE64=VFA`), the 8-byte field contains a 64-bit address.
- When your VFA buffers are below the bar, the 8-byte field contains a 31-bit address preceded by 4 bytes of binary zeros.

param_7

Restriction - Do not alter this address or field:

This is the address of an 8-byte field containing the address of the long-term pool record. You can use this parameter whether your VFA buffers are above the bar or not.

- When your VFA buffers are above the bar (`SCTGEN AMODE64=VFA`), the 8-byte field contains a 64-bit address.
- When your VFA buffers are below the bar, the 8-byte field contains a 31-bit address preceded by 4 bytes of binary zeros.

Note: When you use a 64-bit address for accessing the pool record or pool control fields, you must use 64-bit binary operations and 64-bit addressing mode (AMODE). See *z/Architecture Principles of Operation* for more information about 64-bit programming techniques.

After invoking a callable service in your exit, you must re-load any 64-bit address before using it again. For example:

```
.....
DXCSERV URTN1,PARM=(UDATA1,UDATA2)
L     R01,PARMSAVE  LOAD INPUT PARAMETER LIST
L     R03,PARMx     LOAD ADDRESS OF PARAMETER FIELD
LG    R02,0(,R03)   LOAD 64-BIT ADDRESS
SAM64 ,             AMODE 64
.....
.....
SAM31 ,             AMODE 31
```

ALCS tests the following return conditions from USRGFS:

R15=0
Dispense the record.

R15=4
Do not dispense the record (lost address).

R15=8
Do not dispense the record (other reason).

Global record ID changed exit - USRGIDC

This exit is called when the ALCS global area keypointing routine detects a change in the record ID of a global record that is being keypointed. Use this exit to control the action that is taken when this condition occurs.

ALCS calls this exit if it detects an incorrect ID at the time of keypointing a global record (for which keypointing is defined). You can decide whether ALCS terminates or issues a dump.

If this exit is not loaded, ALCS terminates when it detects this error condition.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUGIDC. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRGIDC with the following conditions:

param_1
The address of the copy of the record as it is currently loaded in the global area.

param_2

Restriction - Do not alter this address or field:

The address of the 2-byte record ID expected by ALCS.

param_3

Restriction - Do not alter this address or field:

The address of the 4-byte file address.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRGIDC:

R15=0
Bypass the keypointing request and issue a dump.

R15=4
Terminate ALCS.

Third party conversational trace exit - USRGTIS

ALCS calls this exit during conversational trace instruction stepping.

This exit is intended for use with third party software. For more information contact the ALCS Support Group (email alcs@uk.ibm.com).

Validate global record keypointing exit - USRGUPD

This exit is called when the ALCS global keypointing routine receives a request to keypoint a record in the global area. Use this exit to validate the contents of the global record before ALCS writes it to the database.

When you specify CHECK=YES in the G01G0 macro for a global record, ALCS calls this exit before it keypoints the record to the database.

You can approve the keypoint request or decide whether ALCS terminates or issues a dump.

If this exit is not loaded, keypointing requests are not validated.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUGUPD. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRGUPD with the following conditions:

param_1

Restriction - Do not alter this address or field:

The address of the old copy of the record as it was previously keypointed.

param_2

Restriction - Do not alter this address or field:

The address of the new copy of the record as it is currently loaded in the global area.

param_3
The address of a 1-byte field which contains a byte-count for the number of bytes that were header-stripped from the record when it was loaded in the global area.

param_4

Restriction - Do not alter this address or field:

The address of a 2-byte field which contains a byte-count that defines the length of this record in the global area.

param_5

Restriction - Do not alter this address or field:

The address of the 4-byte file address.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRGUPD:

R15=0
Allow the keypointing request.

R15=4
Bypass the keypointing request and issue a dump.

R15=8
Terminate ALCS.

HTTP Client receive response header exit – USRHTTP1

This exit allows the user to read the HTTP headers and trailers that are sent by the remote HTTP Server in response to an HTTP request sent by ALCS. It is called from DXCHTTP multiple times once for each header or trailer. This exit must be called if the user wants to read the HTTP headers and trailers. This exit must be called if the user wants to read the HTTP response.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUHTP1. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRHTTP1 with the following conditions:

param_1
The address of the ECB. See the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_2
The address of the status line structure. The format of the structure is mapped by HWTH_STATUS_LINE_TYPE in the HWTHIASM macro.

param_3
The address of 4 bytes of exit flags.

param_4
The address of the header name address.

param_5
The address of a fullword containing the header name length.

param_6
The address of the header value address.

param_7
The address of a fullword containing the header value length.

param_8
The address of the header user data address or zeros.

param_9
The address of a fullword containing the header user data length or zeros.

`param_8` and `param_9` are zeros unless an HWTHSET call with a HWTH_OPT_RESPONSEHDR_USERDATA option is specified.

param_10
The address of a 768-byte work area that the exit can use.

Do not use the following services or macros:

- DXCSAVE PUSH/POP macros
- CPDMP macros
- Callable services (DXCSERV) except for UMSGT2

All registers (except R15) must be the same as at entry.

HTTP (not ALCS) tests the following return conditions on return from USRHTTP1:

R15=0
HTTP continues to process response headers, response trailers, and the response body HWTH_RESPONSE_EXIT_RC_OK

R15=1
HTTP does not process any more response headers, response trailers, or the response body. HTTP discards any remaining data from the remote HTTP Server HWTH_RESPONSE_EXIT_RC_ABORT.

HTTP Client receive response body exit – USRHTTP2

This exit allows the user to read the HTTP response that is sent by the remote HTTP Server in response to an HTTP request sent by ALCS. It is called from DXCHTTP once for the response body. This exit must be called if the user wants to read the HTTP response.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUHTP2. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRHTTP2 with the following conditions:

param_1
The address of the ECB. See the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_2
The address of the status line structure. The format of the structure is mapped by HWTH_STATUS_LINE_TYPE in the HWTHIASM macro.

param_3
The address of 4 bytes of exit flags.

param_4
The address of the response body data address or zeros.

param_5
The address of a fullword containing the length of the response body data or zeros.

param_4 and param_5 are zeros if there is no response data from the HTTP Server.

param_6
The address of the response body user data address or zeros.

param_7
The address of a fullword containing the length of the response body user data or zeros.

param_6 and param_7 are zeros unless an HWTHSET call with a HWTH_OPT_RESPONSEBODY_USERDATA option is specified.

All registers must be the same as at entry.

Neither ALCS nor HTTP checks R15 on return from USRHTTP2.

ICSF parameter exit - USRICF1

This exit allows the user to change the encryption/decryption key used in the CRYPC macro or the crypc() api. It is called from DXCICSF. This allows a simple key management policy to be established, where the keys used to encrypt data can be changed on a regular basis - a requirement for some organizations.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUICF1. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRICF1 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2
The address of the processing type — one byte field containing:

```
x'01' Encrypt
x'02' Decrypt
```

param_3
The address of the key type — one byte field containing:

```
x'01' DES Key
x'02' AES key
```

param_4
The address of the key label name length — a one-byte field containing the length of the key label name, which can have a value between x'01' and x'40'

param_5
The address of the key label name — a field of 1 through 64 bytes containing the name of the key as defined in the CKDS.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRICF1:

R15=0
The key label parameters were not changed, processing continues normally.

R15=4
At least one of the key label parameters was changed. They will be re-checked in DXCICSF and processing will continue normally.

Initialization exit - USRINIT

This exit is called when the ALCS job or started task starts. This is similar to the TPF IPL. Use this exit (for example) to initialize user tables, or to open MVS data sets.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUINIT. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRINIT with the following conditions:

param_1

Restriction - Do not alter this address or field:

The address of an 8-byte field which holds the user parameter (as specified on the EXEC statement of the ALCS JOB) padded to the right with blanks.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRINIT:

R15=0
The user initialization is successful.

R15=4
The user initialization is not successful. This return causes ALCS to terminate abnormally.

Logging option override exit - USRLOG

Use this exit to override the normal logging options. For example, logging options normally apply to all the records of one type (all #WAARI records are logged or all #WAARI records are not logged); this exit point can set different options for one record (a particular #WAARI record can be logged even if other #WAARI records are not logged).

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCULOG. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRLOG with the following conditions:

param_1

Restriction - Do not alter this address or field:

The VFA buffer address.

In some instances, this parameter will be zero (see note at the end of this topic). This is the 31-bit address of the VFA buffer. You can not use this address if you specified SCTGEN AMODE64=VFA in your system configuration; use **param_6** instead.

param_2

Restriction - Do not alter this address or field:

The address of the working storage block containing new record contents. In some instances this parameter will be zero, or will not contain the address of the working storage block (see note at the end of this topic).

param_3

Restriction - Do not alter this address or field:

The address of a 4-byte field which contains the physical size (CI size) of the VFA buffer.

param_4

Restriction - Do not alter this address or field:

The address of a 1-byte field of VFA options to use.

A bit mask indicates new VFA options as follows:

Bit
Meaning

0
ON (1) for delayed file.

1
ON (1) for permanently resident.

2
ON (1) for time initiated file.

4
ON (1) for update mode.

6
ON (1) for hiperspace backing.

param_5

Restriction - Do not alter this address or field:

The address of 1-byte record-control indicator field. Refer to the description of the RONIC macro in the *ALCS Application Programming Reference - Assembler* for details of the RONIND byte.

param_6

Restriction - Do not alter this address or field:

This is the address of an 8-byte field containing the address of the VFA buffer. You can use this parameter whether your VFA buffers are above the bar or not.

- When your VFA buffers are above the bar (SCTGEN AMODE64=VFA), the 8-byte field contains a 64-bit address.
- When your VFA buffers are below the bar, the 8-byte field contains a 31-bit address preceded by 4 bytes of binary zeros.

Note: When you use a 64-bit address for accessing the VFA buffer, you must use 64-bit binary operations and 64-bit addressing mode (AMODE). See *z/Architecture Principles of Operation* for more information about 64-bit programming techniques.

After invoking a callable service in your exit, you must re-load any 64-bit address before using it again. For example:

```
.....
DXCSERV URTN1,PARM=(UDATA1,UDATA2)
L     R01,PARMSAVE  LOAD INPUT PARAMETER LIST
L     R03,PARMx     LOAD ADDRESS OF PARAMETER FIELD
LG    R02,0(,R03)   LOAD 64-BIT ADDRESS
SAM64 ,             AMODE 64
.....
.....
SAM31 ,             AMODE 31
```

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRLOG:

R15=0
Use default logging option.

R15=4
Log the record.

R15=8
Do not log the record.

Note: USRLOG is called for records retrieved by ECB-controlled programs and also for records retrieved for its own purposes (these are called system records, ALCS system fixed file records, and application pool records containing updated release file information). When USRLOG is called by ALCS for a system record, it is in fact called twice, once before the record is changed and again after it has been changed. At the first intercept, **param_1** (or param_6) is the VFA buffer address and **param_2** is zero. At the second intercept, **param_1** is zero and **param_2** (or **param_6**) is the VFA buffer address. Your installation-wide monitor exit must handle both these instances by testing for either parameter being zero.

ZLOGN command audit exit - USRLOGN

ALCS calls this exit after a RACF user verification that is invoked by the ZLOGN command process.

The RACF user verification process can fail or not. Either way, a response message is built as a result of the process.

ALCS enters USRLOGN with the following conditions:

param_1

Restriction - Do not alter this address or field:

The address of the RACF user verification response message.

param_2

Restriction - Do not alter this address or field:

The address of the communication table entry for the resource. Refer to "Communication resource data DSECT - CO0RE" on page 312 for details of the layout of this area.

All registers (except R15) must be the same as at entry.

MQ input bridge address exit - USRMQB0

By default, the MQ input bridge assumes that the low-order 3 bytes of the MQ correlation identifier (CorrelId) contain the CRI of the originating MQ terminal for this request message. Use this exit when another technique is used to identify the originating terminal. For example:

- CRI is contained in the MQ message identifier (MsgId).
- CRI or CRN is contained in a user-defined message header.
- Message is formatted as an ALCS CM1CM input message, containing the CRI in field CM1CRI.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUMQB0. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRMQB0 with the following conditions:

EBROUT
Contains the CRI of the originating MQ queue resource.

EBW004-EBW007
Contains the length of the request message.

D0
The request message is in a block attached to level D0.

D1
The MQ message descriptor for the request message is in a block attached to level D1.

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of the communication table entry for the originating MQ queue resource. Refer to the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_3
The address of a 4-byte field containing the CRI of the originating terminal in the low-order 3 bytes. You must set this return parameter when you use return code 4.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRMQB0:

R15=0
Use the default address technique.

R15=4
Use the terminal address returned by this exit.

R15=8
Processing is complete. Discard the request message.

MQ input bridge format exit - USRMQB1

By default, the MQ input bridge assumes that the request message does not contain any special headers. Use this exit when user headers do exist.

USRMQB1 must be sensitive to the format of the request message, for example:

- User-defined message headers.
- Message structure.
- Message length.
- Character set.
- Encoding.

USRMQB1 can also use information contained in the MQ message descriptor, including:

MsgID
Message identifier, 24 hexadecimal bytes. This is a byte string that is used to distinguish one message from another.

CorrelID
Correlation identifier, 24 hexadecimal bytes. This is a byte string that is used to relate one message to another, or to other work that the application is performing.

ReplyToQ
Name of reply queue, 48 alphanumeric characters. This is the name of the message queue to which the response message should be sent.

ReplyToQMgr
Name of reply queue manager, 48 alphanumeric characters. This is the name of the queue manager to which the response message should be sent.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUMQB1. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRMQB1 with the following conditions:

EBROUT
Contains the CRI of the originating MQ queue resource.

EBW004-EBW007
Contains the length of the request message.

D0
The request message is in a block attached to level D0.

D1
The MQ message descriptor for the request message is in a block attached to level D1.

param_1
The address of the ECB. Refer to the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of the communication table entry for the originating MQ queue resource. Refer to the C00RE description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_3

Restriction - Do not alter this address or field:

The address of the communication table entry for the originating MQ terminal. Refer to the C00RE description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

All registers (except R15) must be the same as at entry.

You can use this exit to modify the content of the request message, in which case you must also update EBW004-EBW007 to contain the new length of the message. Otherwise EBW004-EBW007 must contain its original value on return from the exit.

ALCS tests the following return conditions from USRMQB1:

R15=0
Use the default message format. The message data starts at displacement 0 in the storage block. The MQ input bridge reformats the request message into the format expected by the application to which the originating terminal is routed (determined by the COMDEF SFORM parameter).

R15=4
Use the reformatted message returned by this exit. The message data starts at displacement 0 in the storage block attached on level D0. For legacy applications, the message must fit in a size L1 or L3 storage block.

R15=8
Processing is complete. Discard the request message.

MQ output bridge format exit - USRMQB2

By default, the MQ output bridge assumes that the response message does not contain any special headers. Use this exit when user headers need to be added to the response message.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUMQB2. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRMQB2 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of the communication table entry for the originating MQ queue resource. Refer to the C00RE description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_3
The address of the output message block attached to the ECB.

param_4
The address of the MQ message descriptor for the request message, or zero if there is none.

param_5
The address of a 4-byte field containing the length of the reformatted message data. You must set this return parameter when you use return code 4.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRMQB2:

R15=0
Use the default message format. The message is assumed to be in CM1CM output message format. The message length is in field CM1CCT. and the message data starts at field CM1CMW in the storage block.

R15=4
Use the reformatted message. The message data starts at displacement 0 in the storage block.

R15=8
Processing is complete. Discard the response message.

MQ output bridge message descriptor exit - USRMQB3

The MQ output bridge builds an MQ message descriptor for the response message, using information from the message descriptor for the request message if it is available. Use this exit when additional information needs to be included in the message descriptor. For example, you may wish to modify the *Expiry* field to set an expiration time for the message.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUMQB3. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRMQB3 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of the communication table entry for the originating MQ queue resource. Refer to the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_3
The address of the output message block attached to the ECB.

param_4
The address of the MQ message descriptor for the request message, or zero if there is none.

param_5
The address of the MQ message descriptor for the response message.

param_6
The address of a 4-byte field containing the address of the response message data which ALCS will put onto the MQ queue. The data includes any reformatting carried out by USRMQB2.

param_7
The address of a 4-byte field containing the length of the response message data which ALCS will put onto the MQ queue. The data includes any reformatting carried out by USRMQB2.

param_8
The address of a 4-byte field containing the PUT options. The ALCS required options are set on entry. Any required options that are changed in this exit are reset on return to DXCMQB.

ALCS tests the following return conditions from USRMQB3:

R15=0
Continue as normal.

R15=4
Processing is complete. Discard the response message.

MQI authorization exit - USRMQI1

ALCS calls this exit before it processes any MQI call. Use this exit to validate or restrict (or both) authorization for the MQI call, according to the originating terminal address.

You may also use this exit to identify an input or output message. This causes the count of input or output MQ messages to be incremented. ZSTAT MSG or ZSTAT ALL displays these counts. If data collection is currently collecting statistics about input and output messages, this also causes the statistics for input or output MQ messages to be incremented.

A sample installation-wide monitor exit is provided with the product in the installation-wide exit library, which shows how to identify input and output MQ messages. This is called DXCUMQI1. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRMQI1 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of a 1-byte CMQIC macro request type.

The following request types are supported:

Value	MQI call type
1	MQCONN
2	MQDISC
3	MQOPEN
4	MQCLOSE
5	MQPUT1
6	MQPUT
7	MQGET
8	MQINQ
9	(reserved)
10	MQSET
11	MQCMIT or CSQBCMT
12	MQBACK or CSQBBAK
13	MQAWAIT

param_3

Restriction - Do not alter this address or field:

The address of the MQI call parameter list.

For more information, refer to *WebSphere MQ for z/OS Application Programming Reference*.

param_4

Restriction - Do not alter this address or field:

The address of a 48-byte WebSphere MQ for z/OS queue manager name padded to the right with blanks. This is the name of the queue manager to which ALCS is currently connected. Use the MQMM=parameter of the SCTGEN generation macro to specify a default name. Use the ZCMQI command to connect ALCS to the queue manager.

param_5

Restriction - Do not alter this address or field:

The address of a 3-byte field which contains the CRI of the originating terminal for the entry that issued the MQI call.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRMQI1:

R15=0
Continue processing

R15=4
Not authorized. Return an error to application. The completion code in the MQI-call parameter list is set to MQCC_FAILED. The reason code (RC) in the MQI-call parameter list is set to MQRC_NOT_AUTHORIZED.

R15=8
Not authorized. Terminate the entry with a CTL-0004A3 dump.

R15=12
Increment the input message count and continue processing.

R15=16
Increment the output message count and continue processing.

MQI call completion exit - USRMQI2

ALCS calls this exit after it finishes processing an MQI call from an application program. Use this exit to take specific actions when certain MQI calls complete (for example, MQCLOSE) or to record data about specific MQI calls. This exit can examine the contents of the input parameters to determine the type of MQI call that has just completed.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUMQI2. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRMQI2 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of a 1-byte CMQIC macro request type.

The following request types are supported:

Value
MQI call type

1
MQCONN

2
MQDISC

3
MQOPEN

4
MQCLOSE

5
MQPUT1

6
MQPUT

7
MQGET

8
MQINQ

9
(reserved)

10
MQSET

11
MQCMIT or CSQBCMT

12
MQBACK or CSQBBAK

13
MQAWAIT

param_3

Restriction - Do not alter this address or field:

The address of the MQI call parameter list.

For more information, refer to *WebSphere MQ for z/OS Application Programming Reference*.

param_4

Restriction - Do not alter this address or field:

The address of a 48-byte WebSphere MQ for z/OS queue manager name padded to the right with blanks. This is the name of the queue manager to which ALCS is currently connected. Use the MQMM= parameter of the SCTGEN generation macro to specify a default name. Use the ZCMQI command to connect ALCS to the queue manager.

param_5

Restriction - Do not alter this address or field:

The address of a 3-byte field which contains the CRI of the originating terminal for the entry that issued the MQI call.

All registers (except R15) must be the same as at entry.

MQSeries interface monitoring exit - USRMQI3

ALCS calls this exit when the MQSeries interface fails to return control to ALCS after more than 1 second. Use this exit to assess the situation and request ALCS to disable MQSeries calls if required.

While MQSeries calls are disabled, any entries that issue MQSeries calls get control back immediately with completion code MQCC_FAILED and reason code MQRC_CONNECTION_BROKEN. This can prevent a queue of entries building up on the MQSeries work list.

The ALCS entry dispatcher calls this exit at 1-second intervals until either MQSeries responds or the exit requests that MQSeries calls should be disabled. MQSeries calls are automatically enabled again as soon as the queue manager responds.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUMQI3. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRMQI3 with the following conditions:

param_1
The address of a doubleword field containing the time-of-day (TOD) clock value when the latest call to MQSeries started.

param_2
The address of a doubleword field containing the time-of-day (TOD) clock value now.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRMQI3:

R15=0
No action required.

R15=4
ALCS will suspend its MQSeries activity.

System error processing exit - USRPCH

The system error routine calls USRPCH before it decides whether to take a system error dump.

Note: The system error routine does not call this exit if the system error occurs in another installation-wide monitor exit.

Use this exit to override default dump options, or to implement an online dump facility (to capture an image of the dump), or both.

USRPCH can do the following:

- Perform user error handling functions (such as online dump facility)
- Override the current DUMP/NODUMP and DUP/NODUP options
  - Inhibit the system error dump for this error (force NODUMP)
  - Conditionally take a system dump (force DUMP)

Some ALCS users have exploited this installation-wide exit to implement an online dump facility.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUPCH. IBM advises that you use this as the basis for your installation-wide exit.

Do not use the CPDMP macro in this exit.

ALCS enters USRPCH with the following conditions:

param_1

Restriction - Do not alter this address or field:

The address of the interrupt work area. Refer to the IR0WA description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area. If you use the IR0WA DSECT macro, you must also code a CPSEQ macro to define the symbol #CAR.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRPCH:

R15=0
Use default option.

R15=4
Force nodump option.

R15=8
Force dump option.

R15=12
Force catastrophic dump option.

Long-term pool ID or RCC change exit - USRPIDC

About this task

ALCS calls this exit during a file operation if there is a mismatch in the record ID or the RCC (or both) of a long-term pool record. The following conditions apply:

Procedure

1. The long-term pool record was filed at least once previously (so it must have been dispensed as well)
2. The record ID or RCC of the record (to be filed) does not match the record ID or RCC of the long-term pool record.

Results

Use this exit to determine what action ALCS takes.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUPIDC. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRPIDC with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of a 4-byte field containing the file address.

param_3

Restriction - Do not alter this address or field:

The address of a 4-byte field containing the program name.

param_4

Restriction - Do not alter this address or field:

The address of the new copy of the record.

param_5

Restriction - Do not alter this address or field:

The address of the old copy of the record. This is the 31-bit address of the VFA buffer. You can not use this address if you specified SCTGEN AMODE64=VFA in your system configuration; use **param_7** instead.

param_6

Restriction - Do not alter this address or field:

The address of the pool stamps of the old copy of the record. Refer to the RS0RS description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area. This is the 31-bit address. You can not use this address if you specified SCTGEN AMODE64=VFA in your system configuration; use **param_8** instead.

param_7

Restriction - Do not alter this address or field:

This is the address of an 8-byte field containing the address of the old copy of the record. You can use this parameter whether your VFA buffers are above the bar or not.

- When your VFA buffers are above the bar (SCTGEN AMODE64=VFA), the 8-byte field contains a 64-bit address.
- When your VFA buffers are below the bar, the 8-byte field contains a 31-bit address preceded by 4 bytes of binary zeros.

param_8

Restriction - Do not alter this address or field:

The address of an 8-byte field containing the address of the pool stamps of the old copy of the record. Refer to the RS0RS description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area. You can use this parameter whether your VFA buffers are above the bar or not.

- When your VFA buffers are above the bar (SCTGEN AMODE64=VFA), the 8-byte field contains a 64-bit address.
- When your VFA buffers are below the bar, the 8-byte field contains a 31-bit address preceded by 4 bytes of binary zeros.

Note: When you use a 64-bit address for accessing the pool record or pool control fields, you must use 64-bit binary operations and 64-bit addressing mode (AMODE). See *z/Architecture Principles of Operation* for more information about 64-bit programming techniques.

After invoking a callable service in your exit, you must re-load any 64-bit address before using it again. For example:

```
.....
DXCSERV URTN1,PARM=(UDATA1,UDATA2)
L     R01,PARMSAVE  LOAD INPUT PARAMETER LIST
L     R03,PARMx     LOAD ADDRESS OF PARAMETER FIELD
LG    R02,0(,R03)   LOAD 64-BIT ADDRESS
SAM64 ,             AMODE 64
.....
.....
SAM31 ,             AMODE 31
```

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRPIDC:

R15=0
Allow the file operation.

R15=4
Terminate the entry with a CTL-00002A dump.

User routine - USRRTN1

ALCS calls this exit when an installation-wide monitor exit calls the URTN1 callable service. Use this exit to provide user-written processing.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCURTN1. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRRTN1 with the following conditions:

R01
The address of the parameter list provided by the URTN1 call.

All registers (except R15) must be the same as at entry.

ALCS passes the following return conditions to URTN1:

R15=*x*
Set by this installation-wide exit. Can be any value except 20

R15=20
ALCS reserves this value to indicate that this exit is not loaded.

User routine - USRRTN2

ALCS calls this exit when an installation-wide monitor exit calls the URTN2 callable service. Use this exit to provide user-written processing.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCURTN2. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRRTN2 with the following conditions:

R01
The address of the parameter list provided by the URTN2 call.

All registers (except R15) must be the same as at entry.

ALCS passes the following return conditions to URTN2:

R15=*x*
Set by this installation-wide exit. Can be any value except 20.

R15=20
ALCS reserves this value to indicate that this exit is not loaded.

External security manager exit - USRSAF

ALCS calls this exit during AUTHC DATA monitor-request macro processing. If your installation uses a security product other than IBM's RACF (see the description of the SCTGEN RACF parameter in “SCTGEN macro” on page 72), use this exit to extract the appropriate installation-defined data from your external security manager.

Alternatively, you can use this exit if you use RACF but wish to code your own method of retrieving the installation-defined data.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUSAF. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRSAF with the following conditions.

param_1
The address of a 1-byte AUTHC macro request type.

Value
AUTHC call type

03
DATA=USER

04
DATA=GROUP

05
DATA=ANY

param_2
The address of the ECB. Refer to the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_3

Restriction - Do not alter this address or field:

You can alter the **user area** of this communication-table item.

The address of the communication table entry for the originating terminal. Refer to the C00RE description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_4

Restriction - Do not alter this address or field:

The address of the accessor environment element (ACEE) for the end user that is logged on to the originating terminal or the default user ID for the originating resource.

The ACEE is a control block that contains a description of the current user ID, connected GROUP, user attributes and group authorities. It is constructed by the external security manager when ALCS verifies the user ID. Refer to the description of the ACEE in the documentation for your security product for details of the layout of this area.

param_5
The address of a fullword field containing the address of the application program's storage area where AUTHC returns data. The exit must change to entry storage PSW key before updating this storage area.

param_6
The address of a fullword field containing the length of the storage area where AUTHC returns data.

The exit can return in entry or tables storage PSW key.

All registers (except R15 and R00) must be the same as at entry.

ALCS tests the following return conditions from USRSAF:

R15=0
Processing completed without serious error. The exit must copy the installation-defined data into the storage area pointed to by **param_5** and store the length of the data in the field pointed to by **param_6**. If there is no installation-defined data then store zero in the length field.

R15=8
Processing failed.

External security manager exit - USRSAF2

ALCS calls this exit during GSAFC monitor-request macro processing. If your installation uses a security product other than IBM's RACF, use this exit to construct a callable service command image for your ESM. An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUSAF2. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRSAF2 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_2
The address of the command image parameter as built for RACF - points to the two byte length before the command.

param_3
The address of the command itself as built for RACF - points to the command itself.

param_4
The address of a 1-byte GSAFC macro request type.

Value GSAFC request type
01
DEFINE

02
DELETE

05
RESET

06
DATA

Return codes
R15=0 - the length of the command image has not changed

R15<>0 - the new length of the command image

Migrated sequential file exit - USRSEQ1

ALCS calls this exit during the processing of each sequential file TOPNC monitor-request macro or `topnc` C function. An input general sequential file may have been migrated by DFHSM, therefore if your ALCS system allows TOPNC to wait for migrated general sequential files (see the description of the SCTGEN MIGSEQ parameter in "SCTGEN macro" on page 72), you must use this exit to request ALCS to wait for the recall by DFHSM of specific general sequential files (if they have been migrated).

You can also use this exit to force the general sequential file buffers below the 16MB line.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUSEQ1. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRSEQ1 with the following conditions.

param_1
The address of a 3-byte field containing the symbolic name of a general sequential file.

param_2
The address of a 50-byte work area that the exit can use.

Do not use the CPDMP macroinstruction in this exit.

Do not use ALCS callable services in this exit.

Do not use PUSH/POP parameters on the DXCSAVE macroinstruction in this exit.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRSEQ1:

R15=0
Do not wait for DFHSM to recall this sequential file if it has been migrated. Use sequential file buffers above 16MB line.

R15=4
Wait for DFHSM to recall this sequential file if it if it has been migrated. Use sequential file buffers above 16MB line.

R15=8
Do not wait for DFHSM to recall this sequential file if it has been migrated . Force sequential file buffers below 16MB line.

R15=12
Wait for DFHSM to recall this sequential file if it has been migrated . Force sequential file buffers below 16MB line.

SLC communications receive exit - USRSLC3

ALCS calls this exit when it receives a message on an SLC link.

Use this exit to:

- Add to the normal input message processing
- Replace the normal input message processing
- Discard the message

Use this exit (for example) to perform host-to-host input message processing.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUSLC3. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRSLC3 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2
The address of the attached message block. Refer to the CM8CM description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_3

Restriction - Do not alter this address or field:

You can alter the **user area** of this communication-table item.

The address of the communication table entry for the SLC link. Refer to the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

All registers (except R15) must be same as at entry.

ALCS tests the following return conditions from USRSLC3:

R15=0
Continue processing.

R15=4
Release the ECB and bypass the processing.

R15=8
Bypass the processing.

SLC communications send exit - USRSLC4

ALCS calls this exit point before translating and reformatting an output message prior to adding it to the output message queue for the SLC link. Use this exit to:

- Add to the normal output message processing
- Replace the normal output message processing
- Discard the message.

Use this exit (for example) to perform user translation or data compaction, or to perform host-to-host output message processing.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUSLC4. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRSLC4 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2
The address of the message block. Refer to the CM1CM description in *ALCS Application Programming Reference - Assembler* for details of the layout of this area.

param_3

Restriction - Do not alter this address or field:

You can alter the **user area** of this communication-table item.

The address of the communication table entry for the SLC link. Refer to the C0ORE description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_4

Restriction - Do not alter this address or field:

You can alter the **user area** of this communication-table item.

The address of the communication table entry for the destination ALC terminal (if there is one). Refer to the C0ORE description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_5

The address of the reformatted message block. Refer to the CM8CM description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRSLC4:

R15=0

Continue processing.

R15=4

Continue processing without translating the message data.

DB2 application plan selection and authorization exit - USRSQL1

ALCS SQL support calls this exit immediately before ALCS decides which application plan to associate with the entry (that is, which plan ALCS will specify with the CAF OPEN that it issues for the entry).

Use this exit to override the default application plan name chosen by ALCS (the name of the load module containing the application program). You can also use this exit to allow or deny use of a particular plan by a particular transaction based on, for example, the originating terminal address.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUSQL1. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRSQL1 with the following conditions:

param_1

The address of the ECB. Refer to the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_2

The address of an 8-byte field containing the application program load module name (this is the default plan name).

param_3

Restriction - Do not alter this address or field:

The address of a 4-byte field containing the application program name.

param_4

Restriction - Do not alter this address or field:

The address of an 8-byte field containing the DB2 DBRM name.

param_5

Restriction - Do not alter this address or field:

The address of a 3-byte field containing the CRI of the originating terminal.

param_6

Restriction - Do not alter this address or field:

The address of a 4-byte field containing the DB2 subsystem name (SSNM).

All registers (except R15) must be the same as at entry.

param_2 can be returned unchanged (to use the default plan name), or can be modified to point to a different plan name (eight characters, left-justified, and padded with spaces).

ALCS tests the following return conditions from USRSQL1:

R15=0
Allow the entry to use the plan (either the default plan name, or a new plan name).

R15=4
Do not allow the entry to use the plan. Return to the application program.

R15=8
Do not allow the entry to use the plan. Terminate the entry with a CTL-000454 dump.

SQL authorization exit - USRSQL2

ALCS SQL support calls this exit when an application program issues an SQL call. Use this exit to:

- Verify that the originator has sufficient authority to use DB2
- Indicate that the requesting entry should be terminated

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUSQL2. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRSQL2 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2
The address of an 8-byte field containing the DB2 application plan name.

param_3
The address of a 4-byte field containing the application program name.

param_4
A 4-byte field containing the offset from the start of the application program to the return point from the SQL call.

param_5
The address of a 3-byte field containing the CRI of the originating terminal.

param_6
A 4-byte field containing the index number for the attached subtask that is processing this SQL call. The first attached subtask has an index number of 1, the second attached subtask has an index number of 2, and so on.

The total number of attached subtasks corresponds to the number of concurrent database access threads defined on the DB2 parameter of the SCTGEN generation macro.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRSQL2:

R15=0
Allow the entry to perform the SQL call.

R15=4
Do not allow the entry to perform the SQL call. Return to the application program.

R15=8
Do not allow the entry to perform the SQL call. Terminate the entry with a CTL-000454 dump.

DB2 application plan change exit - USRSQL3

ALCS SQL support calls this exit immediately before processing an SQL call, whenever ALCS detects that the program load module name has changed since the previous SQL call for the entry.

Use this exit to change the application plan name for SQL calls from this entry, if required. If you specify a different plan from the one currently in use for the entry, ALCS issues CAF CLOSE followed by CAF OPEN for the new plan, before continuing to process the SQL call.

Note: CAF CLOSE commits any database changes and deallocates the current plan; it also closes any held or non-held cursors that your application was using.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUSQL3. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRSQL3 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2
The address of an 8-byte field containing the current application plan name.

param_3
The address of an 8-byte field containing the previous application program load module name.

param_4
The address of an 8-byte field containing the current application program load module name.

param_5

Restriction - Do not alter this address or field:

The address of a 4-byte field containing the application program name.

param_6

Restriction - Do not alter this address or field:

The address of an 8-byte field containing the DB2 DBRM name.

param_7

Restriction - Do not alter this address or field:

The address of a 3-byte field containing the CRI of the originating terminal.

param_8

Restriction - Do not alter this address or field:

The address of a 4-byte field containing the DB2 subsystem name (SSNM).

All registers (except R15) must be the same as at entry.

param_2 can be returned unchanged (to continue using the current application plan name), or can be modified to point to a different plan name (eight characters, left-justified, and padded with spaces).

ALCS tests the following return conditions from USRSQL3:

R15=0
Allow the entry to use the plan (either the same plan name, or a new plan name).

R15=4
Do not allow the entry to use the plan. Return to the application program.

R15=8
Do not allow the entry to use the plan. Terminate the entry with a CTL-000454 dump.

Short-term pool redispense after timeout exit - USRSTD

Note: This exit applies only to the type 2 short-term pool dispense mechanism.

ALCS provides a mechanism to redispense a short-term pool record when an application does not release it.

When ALCS needs to dispense a short-term pool record it cycles through the short-term pool directory. If a record is not available after *N1* cycles, the record is *timed out* and made available for redispensing. Use this exit to set a different value for *N1*.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUSTD. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRSTD (when the record is dispensed) with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of a 2-byte field containing the record ID (if specified) or zero.

param_3
The address of a 2-byte field which contains the default value for *N1*. You can change this value between 1 and 189. The default value is 100, but you can use the ZP00L command to change this value.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRSTD:

R15=0
Use the default value.

R15=4
param_3 contains the address of the new value.

Short-term pool redispense after release exit - USRSTR

Note: This exit applies only to the type 2 short-term pool dispense mechanism.

When ALCS needs to dispense a short-term pool record it cycles through the short-term pool directory. Following the release of a short-term pool record, the record is made available for redispensing only when ALCS cycles through the short-term pool directory *N2* times. Use this exit to set a different value for *N2*.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUSTR. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRSTR with the following conditions:

param_1

Restriction - Do not alter this address or field:

The address of a 4-byte field containing the name of the program which released this short-term pool record.

param_2

Restriction - Do not alter this address or field:

The address of a 2-byte field containing the record ID (if specified).

param_3
The address of a 2-byte field which contains the default value for *N2*. You can change this value between 1 and 25. The default value is 1, but you can use the ZP00L command to change this value.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRSTR:

R15=0
Use the default value.

R15=4
param_3 contains the address of the new value.

Monitor-request macro service exit - USRSVC

Use this exit to implement monitor-requests (SVCs in TPF). All odd-numbered request codes (SVC numbers in TPF) are reserved for user requests (ALCS uses only even-numbered codes).

TPF compatibility:

Attention

Applications that include user-written monitor-request macros can be difficult to port to TPF or other ALCS installations.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUSVC. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRSVC with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area. ALCS sets the following fields in the ECB before calling this exit:

CE1REQ
A 2-byte request code.

CE1SVP
Application program base (4 bytes).

CE1EXT
The address of a parameter list (4-byte). The parameter list starts at offset 2 from the address in CE1EXT.

The online monitor saves application registers R14 through R07 in CE1SVDA through CE1SVF in the ECB (address in **param_1**) before it calls USRSVC.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRSVC:

R15=0
Return to the next instruction.

R15=4
The ECB is now queued, go to the CPU loop.

R15=8
Retry the monitor-request macro.

R15=12
Unknown monitor-request code. Terminate the entry with a CTL-000005 dump.

You must code the following assembler instructions to return:

```
        LA      R00,number_of_parameter_bytes+2
        STH     R00,CE1IDX
```

```
         SPACE 1
         DXCSAVE POP,RESTORE,    RESTORE THE CALLER'S REGISTERS        X
               EXCEPT=((R15))    EXCEPT THE RETURN INFORMATION.
         BR    R14               RETURN TO THE ALCS MONITOR
```

This sequence of instructions returns control to ALCS which in turn returns control to the application program that issued the monitor request macro.

ALCS restores application program registers R14 through R07 from CE1SVDA through CE1SVF in the ECB before it returns control to the application program. If this exit needs to modify the application registers, it must store the new register contents in the corresponding ECB field.

User translate tables - USRTAB1 through USRTAB6

You can use these six exits to provide custom translate tables for the CCITT#2, CCITT#5, and ALC transmission codes. ALCS uses the table (or tables) that you load instead of its own standard tables.

For input messages, ALCS converts:

- Padded ALC to ALC
- Padded CCITT#2 to CCITT#2.

before it uses your custom tables.

For output messages, ALCS converts:

- ALC to padded ALC
- CCITT#2 to padded CCITT#2.

after it uses your custom translate tables.

Empty installation-wide monitor exit shells are provided with the product in the installation-wide exit library. They are called DXCUTAB1 through DXCUTAB6. IBM advises that you use them as the basis for your installation-wide exits.

The custom translation takes place automatically if a user table is loaded, otherwise ALCS uses its internal translate table.

Note: Each table can be loaded separately or as a part of a group using the ZPCTL command with the U option. Each exit must contain only a table, it must **not** contain any executable code.

Table 28 on page 292 lists the table names and translate function.

Table 28. ALCS user translation tables

table name	From	To
USRTAB1	EBCDIC	CCITT#2
USRTAB2	CCITT#2	EBCDIC
USRTAB3	EBCDIC	CCITT#5
USRTAB4	CCITT#5	EBCDIC
USRTAB5	EBCDIC	ALC
USRTAB6	ALC	EBCDIC

This table gives an overview of a translate scheme which uses the ALCS internal table to translate from EBCDIC to CCITT#2, but uses a user-provided table to translate from CCITT#2 to EBCDIC.

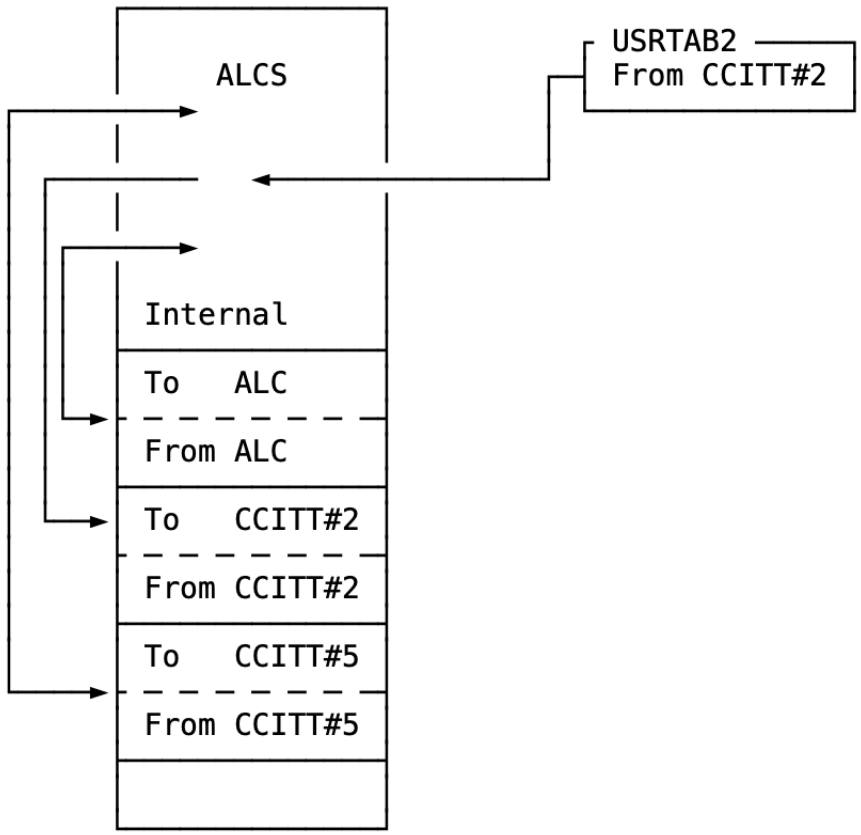


Figure 58. User translation tables

User translate tables for ASCIC - USRTAB7 through USRTAB10

You can use these four exits to provide custom translate tables for the ASCIC monitor-request macro. ALCS uses the table (or tables) that you load instead of its own standard tables.

The ALCS standard translate tables for ASCIC are derived from EBCDIC code page 00037 Version 1 (IBM USA/CANADA - CECP) and ASCII ISO 8859-1 (Latin 1).

Empty installation-wide monitor exit shells are provided with the product in the installation-wide exit library. They are called DXCUTAB7 through DXCUTAB9 and DXCUTABA. IBM advises that you use them as the basis for your installation-wide exits.

The custom translation takes place automatically if a user table is loaded, otherwise ALCS uses its internal translate table.

Note: Each table can be loaded separately or as part of a group using the ZPCTL command with the U option. Each exit must contain only a table, it must not contain any executable code.

USRTAB7
is for user translation from EBCDIC to ASCII when ASCIC ALT=NO is specified or defaulted.

USRTAB8
is for user translation from ASCII to EBCDIC when ASCIC ALT=NO is specified or defaulted.

USRTAB9
is for user translation from EBCDIC to ASCII when ASCIC ALT=YES is specified.

USRTAB10
is for user translation from ASCII to EBCDIC when ASCIC ALT=YES is specified.

TCP/IP blocked send timeout exit - USRTCPA

Use this exit to affect the action that ALCS takes when the TCP/IP blocked send timeout expires.

Specify a blocked send timeout value for a TCP/IP connection on the TIMEOUT parameter of the COMDEF generation macro.

ALCS starts the blocked send timeout when the TCP/IP stack rejects data for transmission with error number EWOULDBLOCK. ALCS continues to try transmitting the same data until either it is accepted by the IP stack, or the timeout expires. When the timeout expires, ALCS conditionally calls USRTCPA.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUTCPA. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRTCPA with the following conditions:

param_1

Restriction - Do not alter this field:

You can only alter the user area of this communication-table item.

The address of communication table entry for the originating TCP/IP resource.

Refer to the C0ORE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2
The address of a 512-byte work area that the exit can use.

Do not use the CPDMP macroinstruction in this exit. Do not use ALCS callable services in this exit. All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRTCPA:

R15=0
Restart the blocked send timeout and continue trying to transmit the data. This is also the default action that ALCS takes if the exit is not loaded.

R15=4
Close the connection immediately.

TCP/IP authorization exit - USRTCP1

ALCS TCP/IP sockets support calls this exit when an application program issues a TCP/IP sockets call.

Use this exit to allow or deny this call based on, for example, the originating terminal address.

You may also use this exit to identify an input or output message. This causes the count of input or output TCP/IP messages to be incremented. ZSTAT MSG or ZSTAT ALL displays these counts. If data collection is currently collecting statistics about input and output messages, this also causes the statistics for input or output TCP/IP messages to be incremented.

A sample installation-wide monitor exit is provided with the product in the installation-wide exit library, which shows how to identify input and output TCP/IP messages. This is called DXCUTCP1. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRTCP1 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of a 1-byte SOCKC macro request type.

The following request types are supported:

Value
TCP/IP sockets call type

1
EZASOKET

2
EZACIC04

3
EZACIC05

4
EZACIC06

5
EZACIC08

param_3

Restriction - Do not alter this address or field:

The address of the TCP/IP sockets call parameter list.

For more information refer to the *z/OS Communications Server IP API Guide*.

param_4
The address of the 3-byte field containing the CRI of the originating terminal.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRTCP1:

R15=0
Continue processing.

R15=4
Not authorized. Terminate the entry with a CTL-0004E3 dump.

R15=8
Not authorized. Terminate the entry with a CTL-0004E4 dump.

R15=12
Increment the input message count and continue processing.

R15=16
Increment the output message count and continue processing.

TCP/IP application protocol exit - USRTCP2

Use this exit to determine if a TCP/IP input message has been completely received.

For TCP/IP connections that use stream sockets, messages may be fragmented during transmission; if that happens, the message fragments are guaranteed to arrive in the correct order but they must be reassembled into the complete message. Alternatively, more than one message can be received in a single read operation.

For TCP/IP connections that implement HTTP (Hypertext Transfer Protocol), you can tell ALCS to handle fragmented input messages by specifying `TERM=(SERVER,HTTP)` in the communication generation.

For other TCP/IP connections, you can use the USRTCP2 exit to check for incomplete input messages or for multiple input messages.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called `DXCUTCP2`. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRTCP2 with the following conditions:

param_1
The address of the received data or zeros. This parameter will be zero if you specified the `COMDEF IPMGSZ` parameter for this TCP/IP connection in your communication generation, use **param_6** instead.

param_2
The address of a fullword containing the length of the received data.

param_3

Restriction - Do not alter this field:

You can only alter the user area of this communication-table item.

The address of communication table entry for the originating TCP/IP resource. Refer to the C0ORE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_4
The address of a 512-byte work area that the exit can use.

param_5
The address of the received data immediately following the first complete message (for return code 12) or the address of the received data immediately following the data to be discarded (for return code 16). You must not use **param_5** if **param_1** is zero; use **param_7** instead.

param_6
The address of an 8-byte field containing the address of the received data or zeros. If you specified the COMDEF IPMGSZ parameter for this TCP/IP connection in your communication generation, the 8-byte field contains a 64-byte address, otherwise this parameter will be zero.

param_7
The address of an 8-byte field containing the 64-bit address of the received data immediately following the first complete message (for return code 12) or the address of the received data immediately following the data to be discarded (for return code 16). Use this parameter if **param_1** is zero; otherwise use **param_5**.

Note: When you use a 64-bit address for accessing the VFA buffer or control fields, you must use 64-bit binary operations and 64-bit addressing mode (AMODE). See *z/Architecture Principles of Operation* for more information about 64-bit programming techniques.

After invoking a callable service in your exit, you must re-load any 64-bit address before using it again. For example:

```
.....
DXCSERV URTN1,PARM=(UDATA1,UDATA2)
L     R01,PARMSAVE  LOAD INPUT PARAMETER LIST
L     R03,PARMx     LOAD ADDRESS OF PARAMETER FIELD
LG    R02,0(,R03)   LOAD 64-BIT ADDRESS
SAM64 ,             AMODE 64
.....
.....
SAM31 ,             AMODE 31
```

Do not use the CPDMP macroinstruction in this exit. Do not use ALCS callable services (except UMSGT2) in this exit. All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRTCP2:

R15=0
All expected data received.

R15=4
More data expected.

R15=8
Discard the data and close the TCP/IP connection immediately.

R15=12
More than one message was received.

R15=16
Discard some of the data.

ALCS calls this exit again when R15=12 or R15=16. This allows you to extract all the messages from the received data.

TCP/IP output message exit - USRTCP3

ALCS calls this exit for all outgoing messages to TCP/IP ALC terminals whose owning TCP/IP connections are defined with TERM=SERVER. ALCS calls this exit immediately before it sends the message. The message text has been translated from EBCDIC according to the translate code defined for the terminal.

You can use this exit to validate and (or) change outgoing messages.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUTCP3. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRTCP3 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2
The address of the output message. If you use TCP/IP large messages, then you must use the 4-byte field CM1CCX instead of the 2-byte field CM1CCT if you inspect the message block (refer to CM1CM in *ALCS Application Programming Reference - Assembler* for a description of the message format). The message block for TCP/IP messages can be either in the extended or in the standard format. When in standard format it is guaranteed that the two characters preceding CM1CCT are binary zeros.

param_3

Restriction - Do not alter this field:

You can only alter the user area of this communication-table item.

The address of communication table entry for the destination terminal.

Refer to the C0ORE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_4

Restriction - Do not alter this field:

You can only alter the user area of this communication-table item.

The address of communication table entry for the destination TCP/IP resource.

Refer to the C0ORE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

All registers must be the same as at entry.

TCP/IP input message exit - USRTCP4

ALCS calls this exit for all incoming messages from TCP/IP connections that are defined with TERM=SERVER. ALCS calls this exit immediately after it receives a complete message.

No ALCS processing or reformatting has been done on the message. You can use this exit to validate and (or) change incoming messages.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUTCP4. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRTCP4 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this field:

You can only alter the user area of this communication-table item.

The address of communication table entry for the originating TCP/IP resource.

Refer to the C0ORE description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_3
The address of a 512-byte work area that the exit can use.

param_4
Zero

Do not use the CPDMP macroinstruction in this exit. Do not use ALCS callable services (except UMSGT2) in this exit.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRTCP4:

R15=0
Continue processing.

R15=4
Origin is a terminal. ALCS processes the message as if it originated from this display or printer terminal. The user must set the following return parameters when R15=4:

param_4
The address of an 8-byte field containing the CRN of the originating terminal.

R15=8
Origin is a terminal. ALCS processes the message as if it originated from this display or printer terminal. The user must set the following return parameters when R15=8:

param_4
The address of a 4-byte field containing the CRI of the originating terminal in the low-order 3 bytes.

R15=12
Discard the data.

TCP/IP connection start exit - USRTCP5

Use this exit to do user processing when a new TCP/IP connection starts.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUTCP5. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRTCP5 with the following conditions:

param_1

Restriction - Do not alter this field:

You can only alter the user area of this communication-table item.

The address of communication table entry for the TCP/IP resource.

Refer to the C0ORE description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

All registers must be the same as at entry.

TCP/IP connection stop exit - USRTCP6

Use this exit to do user processing when a TCP/IP connection stops.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUTCP6. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRTCP6 with the following conditions:

param_1

Restriction - Do not alter this field:

You can only alter the user area of this communication-table item.

The address of communication table entry for the TCP/IP resource.

Refer to the C0ORE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

All registers must be the same as at entry.

TCP/IP ACSA OPEN exit - USRTCP7

ALCS calls this exit when it receives an OPEN request from an ACSA terminal emulator that is connected over TCP/IP. ALCS calls this exit before it checks for a valid CRN in the OPEN request.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUTCP7. IBM advises that you use this as the basis for your installation-wide exit. ALCS enters USRTCP7 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this field:

You can only alter the user area of this communication-table item.

The address of the communication table entry for the originating TCP/IP resource.

Refer to the C0ORE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_3
The address of an 8-byte field that contains the CRN from the OPEN request. The CRN has been converted from ASCII to EBCDIC and padded with spaces to 8 characters.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRTCP7:

R15=0
Continue processing.

R15=4
OPEN request has been completely processed.

R15=8
Reject the OPEN request.

TCP/IP trace exit - USRTCP8

When TCP/IP trace is active ALCS calls this exit before writing each record to the diagnostic sequential file. Use this exit to write the record to a different location or to prevent the record from being written.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUTCP8. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRTCP8 in table storage PSW key with the following conditions:

param_1
The address of the TCP/IP trace record mapped by SD0DR.

param_2
The address of a 4-byte field that contaisnd the lengh of the TCP/IP trace record mapped by SD0DR.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRTCP8:

R15=0
Write the record to the diagnostic sequential file.

R15=4
Do not write the record. Set this return condition if this exit writes the record to a different location or if you want to prevent the record from being written.

The following example shows how you could write TCP/IP trace records to the user sequential file:

```
⋮
     L     R05,PARM1              LOAD ADDRESS OF DATA
     L     R04,PARM2              LOAD ADDRESS OF LENGTH FIELD
     DXCSERV UWSEQ,PARM=((R05),(R04),=AL1(IW0USR))
     B     *+4(R15)               CHECK THE RETURN CODE
     B     RC0                      0 - DATA WRITTEN OUT OK
     B     RC4                      4 - SEQ FILE NOT AVAILABLE
     B     RC8                      8 - ERROR
⋮
```

Stop exit - USRTERM

ALCS calls this exit when the ALCS job or started task stops. Use this exit for user functions, for example to close data sets or to extract sample data.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUTERM. IBM advises that you use this as the basis for your installation-wide exit.

All registers (except R15) must be the same as at entry. Set R15 to 0.

Time-initiated function exit - USRTIM1

Use this exit to perform time-initiated functions. For example, to update tables or to extract sample data. The ALCS timer interrupt routine calls this exit at 0.2-second intervals.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUTIM1. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRTIM1 in tables key and supervisor state.

This exit must return in tables key and supervisor state.

All registers (except R15) must be the same as at entry. Set R15 to 0.

Time-initiated function exit - USRTIM2

Use this exit to perform time-initiated functions. For example, to update tables or to extract sample data. The ALCS entry dispatcher calls this exit at 1-second intervals.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUTIM2. IBM advises that you use this as the basis for your installation-wide exit.

All registers (except R15) must be the same as at entry. Set R15 to 0.

Third party initialization exit - USRTSK1

This exit is called when the ALCS job or started task starts and the SCTGEN parameter USRTSK1=YES is coded.

This exit is intended for use with third party software. For more information contact the ALCS Support Group (email alcs@uk.ibm.com).

VFA option set or option override exit - USRVFA

Use this exit to override the normal (ALCS generation time) VFA processing options during a find or file operation. For example, VFA options normally apply to all the records of one type (all #WAARI records have the same processing options). This exit point can set different options for one record.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUVFA. IBM advises that you use this as the basis for your installation-wide exit. USRVFA is not activated for system records such as:

- Application fixed file #KPTRI
- Files initiated by the ALCS monitor.

ALCS enters USRVFA with the following conditions:

param_1

Restriction - Do not alter this address or field:

The VFA buffer address. This is the 31-bit address of the VFA buffer. You can not use this address if you specified SCTGEN AMODE64=VFA in your system configuration; use **param_6** instead.

param_2

Restriction - Do not alter this address or field:

The address of the record control fields in the record. Refer to the RS0RS description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area. This is the 31-bit address. You can not use this address if you specified SCTGEN AMODE64=VFA in your system configuration; use **param_7** instead.

param_3

The address of a 1-byte field of VFA options to use. A bit mask indicates new VFA options as follows:

Bit

Meaning

0

ON (1) for delayed file.

1

ON (1) for permanently resident.

2

ON (1) for time initiated file.

4

ON (1) for update mode.

6

ON (1) for hiperspace backing.

param_4

Restriction - Do not alter this address or field:

The address of a 2-byte field which contains either 0 (read request) or 1 (write request).

param_5

Restriction - Do not alter this address or field:

The address of 1-byte record-control indicator field. Refer to the description of the RONIC macro in the *ALCS Application Programming Reference - Assembler* for details of the RONIND byte.

param_6

Restriction - Do not alter this address or field:

This is address of an 8-byte field containing the address of the VFA buffer. You can use this parameter whether your VFA buffers are above the bar or not.

- When your VFA buffers are above the bar (SCTGEN AMODE64=VFA), the 8-byte field contains a 64-bit address.
- When your VFA buffers are below the bar, the 8-byte field contains a 31-bit address preceded by 4 bytes of binary zeros.

param_7

Restriction - Do not alter this address or field:

The address of an 8-byte field containing the address of the control fields of the old copy of the record. Refer to the RS0RS description in "Interfaces for installation-wide monitor exits" “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area. You can use this parameter whether your VFA buffers are above the bar or not.

- When your VFA buffers are above the bar (SCTGEN AMODE64=VFA), the 8-byte field contains a 64-bit address.
- When your VFA buffers are below the bar, the 8-byte field contains a 31-bit address preceded by 4 bytes of binary zeros.

Note: When you use a 64-bit address for accessing the VFA buffer or control fields, you must use 64-bit binary operations and 64-bit addressing mode (AMODE). See *z/Architecture Principles of Operation* for more information about 64-bit programming techniques.

After invoking a callable service in your exit, you must re-load any 64-bit address before using it again. For example:

```
.....
DXCSERV URTN1,PARM=(UDATA1,UDATA2)
L     R01,PARMSAVE  LOAD INPUT PARAMETER LIST
L     R03,PARMx     LOAD ADDRESS OF PARAMETER FIELD
LG    R02,0(,R03)   LOAD 64-BIT ADDRESS
SAM64 ,             AMODE 64
.....
.....
SAM31 ,             AMODE 31
```

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRVFA:

R15=0
Use the default values.

R15=4
Use the values in **param_3**.

Implied wait exit - USRWAIT

ALCS calls the USRWAIT installation-wide monitor exit during an implied wait operation (for example WAITC, FINWC, FIPWC, or FIWHC) each time the number of reads for this entry exceeds one of the entry read thresholds. The thresholds are defined by the SCTGEN ENTREAD parameter and can be changed by the SLIMC monitor-request macro.

Use this exit to indicate that the entry should be terminated.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUWAIT. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRWAIT with the following conditions:

param_1

Restriction - Do not alter this address or field:

The address of the ECB. Refer to the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of a 4-byte field that contains the total number of reads for this entry (includes reads as a result of FINDs, reads as a result from FILEs for LT pool, and reads for records in update mode.

param_3

Restriction - Do not alter this address or field:

The address of a 4-byte field that contains the total number of reads saved by VFA and/or Hiperspace.

param_4

Restriction - Do not alter this address or field:

The address of a 4-byte field that contains the total number of USRWAIT invocations for this entry.

param_5

Restriction - Do not alter this address or field:

The address of a 8-byte field that contains the ECB dispense time (TOD clock value).

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRWAIT:

R15=0
Continue processing.

R15¬=0
Terminate the entry.

WAS authorization exit - USRWAS1

ALCS calls this exit before it processes any optimized local adapter (OLA) for WebSphere Application Server for z/OS (WAS) call. Use this exit to validate or restrict (or both) authorization for the OLA call, according the originating terminal address.

You may also use this exit to identify an "input" or "output" message. This causes the count of input or output WAS messages to be incremented. `ZSTAT MSG` or `ZSTAT ALL` displays these counts. If data collection is currently collecting statistics about input and output messages, this also causes the statistics for input or output WAS messages to be incremented. A sample installation-wide monitor exit is provided with the product in the installation-wide exit library, which shows how to identify "input" and "output" WAS messages. This is called `DXCUWAS1`. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRWAS1 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of a 1-byte `WASCC` macro request type.

The following request types are supported:

Value	OLA call type	Description
30	BBOA1REG	REGISTER
31	BBOA1URG	UN-REGISTER
32	BBOA1CNG	CONNECTION GET
33	BBOA1CNR	CONNECTION RELEASE
34	BBOA1SRQ	SEND A REQUEST
35	BBOA1SRP	SEND A RESPONSE
36	BBOA1RCA	RECEIVE A REQUEST (ANY)
37	BBOA1RCS	RECEIVE A REQUEST (SPECIFIC)
38	BBOA1RCL	RECEIVE RESPONSE LENGTH
39	BBOA1GET	GET MESSAGE DATA
40	BBOA1INV	INVOKE A METHOD
41	BBOA1SRV	HOST A SERVICE
42	BBOA1SRX	SEND RESPONSE EXCEPTION DATA
43	BBOA1GTX	GET MESSAGE CONTEXT DATA
44	BBOA1INF	GET LOCAL ADAPTERS INFORMATION

param_3

Restriction - Do not alter this address or field:

The address of the WAS control and vector area. Refer to the DXCWASA description “ALCS WAS control and vector area DXCWASA - DXCWASA” on page 322 for details of the layout of this area.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRWAS1:

R15=0
Continue processing

R15=4
Not authorized. Return an error to the application:

- R15 contains 16 on return from an assembler call and
- a C/C++ function call returns the value 16

R15=8
Not authorized. Terminate the entry with a CTL-000068 dump.

R15=12
Increment the input message count and continue processing.

R15=16
Increment the output message count and continue processing.

R15=20
Increment the input and output message counts and continue processing.

WAS input bridge address exit - USRWAS3

Use this exit to identify the originating terminal if this is protocol type 1 WAS Bridge. The CRI or CRN can be contained in an 8-byte correlator preceding the message. ALCS will remove this 8-byte correlator on

return from this exit. By default, the WAS input bridge assumes there is no originating terminal; it uses the WAS resource as the originator instead.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUWAS3. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRWAS3 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of the communication table entry for the originating WAS resource. Refer to the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_3
Zero

param_4
The address of the input message block attached to this ECB or the address of the input message heap storage area.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRWAS3:

R15=0
Use the default address technique.

R15=4
Use the terminal address returned by this exit and remove the 8-byte correlator. Note that if a correlator is present then it preceeds the actual message text.

R15=8
Processing is complete. Discard the request message.

You must set the following return parameter when R15=4:

param_5
The address of a 4-byte field containing the CRI of the originating terminal in the low-order 3 bytes.

WAS input bridge format exit - USRWAS4

By default, the WAS input bridge assumes that the message does not require any reformatting. Use this exit when reformatting is required. USRWAS4 must be sensitive to the format of the message.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUWAS4. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRWAS4 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of the communication table entry for the originating WAS resource. Refer to the C00RE description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area.

param_3

Restriction - Do not alter this address or field:

The address of the communication table entry for the originating WAS terminal resource (if any, otherwise zero). Refer to the C0ORE description in “Intended interfaces for installation-wide monitor exits” on page 308 for details of the layout of this area.

param_4
The address of the input message block attached to this ECB or the address of the input message heap storage area.

param_5
The address of a 4-byte field containing the length of the message.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRWAS4:

R15=0
Use the default message format.

R15=4
Use the reformatted message

R15=8
Processing is complete. Discard the request message.

You must set the following return parameter when R15=4:

param_6
The address of a 4-byte field containing the new length of the message.

WAS output bridge address exit - USRWAS5

Use this exit to build an 8-byte correlator which identifies the destination terminal, if this is protocol type 1 WAS Bridge. ALCS will add this 8-byte correlator in front of the message on return from this exit.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUWAS5. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRWAS5 with the following conditions:

param_1
The address of the ECB. Refer to the EB0EB description in “Entry control block - EB0EB” on page 323 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of the communication table entry for the destination WAS resource. Refer to “Communication resource data DSECT - CO0RE” on page 312 for details of the layout of this area.

param_3

Restriction - Do not alter this address or field:

The address of the communication table entry for the destination WAS terminal resource (if any, otherwise zero). Refer to “Communication resource data DSECT - CO0RE” on page 312 for details of the layout of this area.

param_4
The address of the output message block attached to this ECB or the address of the output message heap storage area.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRWAS5:

R15=0
Use the address of the WAS resource.

R15=4
Use the 8-byte correlator. Note that the correlator preceeds the actual message text.

R15=8
Processing is complete. Discard the request message.

You must set the following return parameter when R15=4:

param_5
The address of the 8-byte correlator to be added to the message.

WAS output bridge format exit - USRWAS6

By default, the WAS output bridge assumes that the message does not require any reformatting. Use this exit when reformatting is required. USRWAS6 must be sensitive to the format of the message.

An empty installation-wide monitor exit shell is provided with the product in the installation-wide exit library. This is called DXCUWAS6. IBM advises that you use this as the basis for your installation-wide exit.

ALCS enters USRWAS6 with the following conditions:

param_1
The address of the ECB. Refer to "Entry control block - EB0EB" on page 323 for details of the layout of this area.

param_2

Restriction - Do not alter this address or field:

The address of the communication table entry for the destination WAS resource. Refer to "Communication resource data DSECT - CO0RE" on page 312 for details of the layout of this area.

param_3

Restriction - Do not alter this address or field:

The address of the communication table entry for the destination WAS terminal resource (if any, otherwise null) Refer to "Communication resource data DSECT - CO0RE" on page 312 for details of the layout of this area.

param_4
The address of the output message block attached to this ECB or the address of the input message heap storage area.

param_5
The address of the 8-byte correlator to be added to the message.

All registers (except R15) must be the same as at entry.

ALCS tests the following return conditions from USRWAS6:

R15=0
Use the default message format.

R15=4
Use the reformatted message.

R15=8
Processing is complete. Discard the request message.

You must set the following return parameter when R15=4:

param_6
The address of a 4-byte field containing the new length of the message.

Workstation Trace exit - USRWSTR

Use this exit to determine if workstation trace should be enabled for this ECB.

An empty installation-wide monitor exit shell called DXCUWSTR is provided with the product in the installation-wide exit library. Use DXCUWSTR as the basis for your installation-wide exit.

ALCS enters USRWSTR with the following conditions:

param_1
The address of the ECB that will be traced. Refer to the EB0EB description in "Entry control block - EB0EB" on page 323 for details of the layout of this area.

USRWSTR returns with the following conditions:

R15=0
Allow Workstation Trace to continue

R15=4
Do not initialize Workstation Trace

Intended interfaces for installation-wide monitor exits

This section describes the intended interfaces for callable services used by installation-wide monitor exits.

Application message block format - CM1CM

CM1CM is an intended interface. It is described fully in *ALCS Application Programming Reference - Assembler*.

VTAM input message block DSECT - CM5CM

Format

```
CM5CM [REG=reg]
```

REG=*reg*
Base register for DSECT addressability.

Description

Use the CM5CM DSECT macro to reference fields in the VTAM input message block. The DSECT name is CM5CM.

Symbols defined for fields

CM5CM defines the following symbols for fields:

CM5CRI
3-byte origin, containing the originating CRI of the VTAM LU.

CM5DAT
Start of data in the input message block.

CM5FLI
1-byte of flags, containing the message block indication.

The following bits can be used in this byte:

CM5FLINF
Not first block. If set to 0, this is the first block of the message. If set to 1, this is not the first block of the message.

CM5FLINL
Not last block. If set to 0, this is the last block of the message. If set to 1, this is not the last block of the message.

CM5FLIPG
Possible garbled input. If set to 1, this message block might be garbled.

CM5LNA
1-byte Screen Line Address (LNA).

CM5RCL
4-byte input data length from the VTAM Routing Parameter List (RPL).

Note: The following fields are valid only for AX.25 input from a Type 1 PVC.

CM5XTA
1-byte X.25 terminal address.

CM5XTX
Start of X.25 text.

Note: The following fields are valid only for AX.25 input from a Type 2, 3, 4, or 5 PVC.

CM5XTT
Start of X.25 text.

Note: The following fields are valid only for AX.25 input from a Type 6 PVC.

CM5XIA
1-byte X.25 interchange address.

CM5XTA
1-byte X.25 terminal address.

CM5XTX
Start of X.25 text.

Note: The following fields are valid only for ALCI input.

CM5HCT
Length of the header.

CM5HTP
1-byte header type, containing flags.

The following bit can be used in this byte:

CM5HTPMT
If set to 1, the following data is message text.

CM5MIA
1-byte ALCI interchange address.

CM5MLD
3-byte origin LEID, containing the LEID of the ALC terminal from where this message originated.

CM5MTA
1-byte ALCI terminal address.

CM5MTX
Start of ALCI message text.

ALCS SLC message block DSECT - CM8CM

Format

```
        CM8CM [REG=reg]
```

REG=*reg*
Base register for DSECT addressability.

Description

Use the CM8CM DSECT macro to reference fields in the SLC message block. The DSECT name is CM8CM.

Symbols defined for fields

CM8CM defines the following symbols for fields:

CM8CCT
2-byte count, containing the length from CM8LYN to the end of message text inclusive.

CM8LCI
1-byte Link Characteristics Indicator (LCI). Refer to the ATA/IATA *iataref* for usage of this byte.

CM8LYN
1-byte SLC link number.

CM8MBI
1-byte Message Block Identifier (MBI). Refer to the ATA/IATA *iataref* for usage of this byte.

Note: The following fields are valid only for an SLC Type-1 or Type-3 link.

CM8ACH
1-byte Message Characteristics Identifier (MCI). Refer to the ATA/IATA *iatarefl* for usage of this byte.

CM8HEN
2-byte SLC entry address.

CM8HEX
2-byte SLC exit address.

CM8TXH
Start of message text.

Note: The following fields are valid only for a SLC Type-2 link.

CM8ACI
1-byte Message Characteristics Identifier (MCI). Refer to the ATA/IATA *iataref* for usage of this byte.

CM8TXT
Start of message text.

Parsed user command format - C00PR

Format

```
[label]  C00PR REG=reg
                          [,SUFFIX=s]
```

REG=*reg*
Base register for DSECT addressability.

SUFFIX=*s*
Suffix for DSECT, a single alphanumeric character. C00PR allows the same suffix character in multiple macroinstructions (only the first occurrence generates the DSECT). C00PR adds this suffix to all the symbols that it defines.

Description

ALCS optionally parses user command parameters into a work area. Use the C00PR DSECT macro to reference fields in this work area. The DSECT name is C00PR. Refer to "User command exit program - ACM0" on page 363 for details about parsing user commands.

The parsed user command work area contains a header section followed by a parameter section for each parameter that was specified on the command (in the same order).

Header section

C00PR defines the following symbols for fields:

PRSW1
2 bytes of flags, containing error indicators. The following bits can be used in the first byte of this field:

PR1MAX
Too many input parameters.

PR1UMP
Unmatched parenthesis.

PR1LNG
Parameter too long.

PR1MSK
Missing keyword.

PR1KLN
Keyword too long.

PR1QOT
Unmatched quote.

PR1KBD
Invalid keyword.

PRNMX
2-byte field containing the total number of parameter sections in the work area (not all the sections may contain parameter information).

PRNUM
2-byte field containing the number of parameters sections that do contain parameter information.

Parameter section

C00PR defines the following symbols for fields:

PRENT
64-byte field that describes one parameter. This section contains subfields PRKEY and PRPRM.

PRKEY
9-byte field containing the length and name of the keyword (for a keyword parameter - otherwise it contains binary zeros). This field contains subfields PRKLN and PRKWD.

PRKLN
1-byte field containing the length of the keyword.

PRKWD
8-byte field containing the name of the keyword.

PRPRM
53-byte field containing the length and value of the parameter operand. This field contains subfields PRVLN and PRVAL.

PRVLN
1-byte field containing the length of the parameter operand.

PRVAL
52-byte field containing the value of the parameter operand.

Example

The user command

```
ZUSER PARM1,PARM2=(FRED,BILL),PARM3
```

is parsed into the work area in the following format:

Table 29. Example of parsed user command work area

Displacement	Field name	Field name	Field name	Contents
0	PRENT	PRKEY	PRKLN	0
1			PRKWD	(not initialized)
9		PRPRM	PRVLN	5
10			PRVAL	PARM1
64	PRENT	PRKEY	PRKLN	5
65			PRKWD	PARM2
73		PRPRM	PRVLN	11
74			PRVAL	(FRED,BILL)
128	PRENT	PRKEY	PRKLN	0
129			PRKWD	(not initialized)
137		PRPRM	PRVLN	5
138			PRVAL	PARM3

Communication resource data DSECT - C0ORE

Format

```
        C0ORE [REG=reg][,SUFFIX=s]
```

REG=*reg*
Base register for DSECT addressability.

SUFFIX=*s*
Suffix for DSECT, a single alphanumeric character.

Description

Use the C00RE DSECT macro to reference fields in the communication table entry for a given resource. The DSECT name is CO0RE.

The macro is subdivided in a number of sections:

Common section
This section is applicable to all resources. All labels start with REC.

Terminal section
This section is applicable to terminal type resources and also for X.25 PVCs and ALCI (NEF2) LUs. All labels start with RETD.

Application section
This section is applicable to application resources. All labels start with REAA.

SLC link section
This section is applicable to SLC links. All labels start with RESL.

LU 6.1 link section
This section is applicable to LU 6.1 links. All labels start with RELK.

WTTY section
This section is applicable to WTTY links. All labels start with RETY.

APPC section
This section is applicable to APPC connections. All labels start with REAP.

TCP/IP section
This section is applicable to TCP/IP connections. All labels start with REIP.

Usage of bits defined in C00RE

C00RE defines symbols for bits. These symbols are defined so that the *value* of the symbol is the location (byte), and the *length attribute* of the symbol is the bit pattern.

To test these bits, use:

```
TM REC1xxx,L'REC1xxx
```

where *xxx* is the last three characters of the symbol for the bit being tested.
For example:

```
TM REC1PRC,L'REC1PRC
```

Common section - Symbols defined for fields in the common section

C00RE defines the following symbols for fields:

RECOARN
4-byte application name, in a field of eight bytes. If the resource is a terminal, and if it is routed to an application, this field contains the application name, left-justified and padded with spaces.

Note: This name is not usually the same as the name of the input message editor program for the application.

RECOCRI
4-byte field containing the CRI of the resource. The CRI is in the low-order three bytes. Do not use the high-order byte.

RECOCRN
8-byte CRN of the resource.

RECODV1
1-byte device type. The ALCS TRMEQ macro defines symbols for the values that this field can contain.

RECOORN
4-byte communication resource ordinal number. A unique number associated with the communication resource.

RECOOID
1-byte owning system communication ID.

RECOPID
1-byte path destination communication ID.

Symbols defined for bits in the common section

C0ORE defines the following symbols for bits:

REC1AAA
Terminal hold (also called AAA hold). This bit is available for application programs to set and test as required.

REC1ALC
ALC terminal attached via SLC, AX.25, or ALCI.

REC1APN
Alternate printer CRAS. This bit is on (1) if the resource is an alternate CRAS printer terminal.

REC1ATN
Alternate CRAS. This bit is on (1) if the resource is an alternate CRAS terminal.

REC1CST
Active. This bit is on (1) if the resource is in session with ALCS.

For an SLC link. This bit is on (1) if at least one channel on the link has been opened.

For an ALC terminal. The terminal is active if this bit is on and the owning link or LU is also active.

REC1ISS
For a terminal. This bit is on (1) if system generated messages are not sent to the terminal.

REC1LOG
The resource is a terminal and is routed to an application.

REC1NEF
ALC terminal attached through ALCI or through NEF2.

REC1PRC
Prime CRAS. This bit is on (1) if the resource is the prime CRAS terminal.

REC1ROC
ROC. This bit is on (1) if the resource is the read-only (RO) CRAS terminal.

REC1SLC
ALC terminal attached through an SLC link.

REC1STV
STV resource.

REC1TPP
Printer terminal.

REC1TPS
Display terminal.

REC1X25
ALC terminal attached through an AX.25 link.

Symbols defined for fields in the terminal section

C0ORE defines the following symbols for fields:

RETDACI
1-byte alternate CRAS index (AT number) or alternate CRAS printer index (AP number).

RETDARC
4-byte field containing the CRI of the associated resource (if any). The CRI is in the low-order three bytes. Do not use the high-order byte.

RETDBSZ
2-byte buffer size (number of bytes). For example, if the resource is a terminal with a 1920 character buffer, this field contains X'0780' (decimal 1920).

RETDCDE
1-byte translate code for an ALC terminal.

The values that this field can contain:

RETDCDCA
Translate from/to ALC line code.

RETDCDCN
No translation required.

RETDCDC5
Translate from/to CCITT#5 (ASCII) code.

RETDCDC7
Translate from/to CCITT#2 code.

RETDCOL
1-byte number of display columns (when the resource is a VTAM 3270 display terminal).

RETDCRL
4-byte field containing one of the following:

- Address of owning ALCI LU for an ALC terminal connected through ALCI
- Address of owning NEF2 LU for an ALC terminal connected through NEF2
- Address of owning SLC link for an ALC terminal connected through SLC
- Address of owning X.25 PVC for an ALC terminal connected through X.25

RETDFMC
For a VTAM 3270 display terminal, 1-byte number of display columns formatted for end user input by COMCC macro.

RETDFMR
For a VTAM 3270 display terminal, 1-byte number of display rows formatted for end user input by COMCC macro.

RETDHEX
2-byte exit address within the High Level Network (HLN) for an ALC terminal connected through SLC.

RETDLEI
3-byte LEID for an ALC terminal connected through ALCI

RETDLNA
For a VTAM 3270 display terminal, 1-byte screen line number corresponding to the cursor position in the previous input message.

RETDMTC
For a terminal. 4-byte field containing the 32-bit count of input messages received from this resource.

RETDMTD
For a terminal. 4-byte field containing the 32-bit count of input messages discarded by installation-wide monitor exit USRCOM2.

RETDMTT
For a terminal. 8-byte field containing the TOD clock value when the last input message was accepted by installation-wide monitor exit USRCOM2.

RETDOSI
1-byte other system ID.

RETDPCL
1-byte X.25 PVC type.

The values that this field can contain:

RETDPCT1
X.25 PVC is PRTCOL=TYPE1

RETDPCT2
X.25 PVC is PRTCOL=TYPE2

RETDPCT3
X.25 PVC is PRTCOL=TYPE3

RETDPCT4
X.25 PVC is PRTCOL=TYPE4

RETDPCT5
X.25 PVC is PRTCOL=TYPE5

RETDPCT6
X.25 PVC is PRTCOL=TYPE6

RETDRIA
2-byte terminal Interchange Address (IA) for an ALC terminal.

RETDROW
1-byte number of display lines (rows) if the resource is a display terminal.

RETDRTA
2-byte Terminal Address (TA) for an ALC terminal.

RETDTCD
2-byte TCID address within the High Level Network (HLN) for an ALC terminal connected through SLC.

Symbols defined for bits in the terminal section

C0ORE defines the following symbols for bits:

RETDACK
An acknowledgement arrived from a printer resource, but is not yet passed to the printer package.

RETDBBB
For an IBM 3270 terminal. This bit is on (1) if the terminal is in the SNA in-bracket state (that is, if ALCS has received a message with SNA begin bracket from the terminal).

RETDBCS
The resource is a terminal that supports the IBM 3270 Double Byte Coded Character Set.

RETDNE2
If this bit is set to 1, the resource is one of:

An ALC terminal connected through ALCI (if bit REC1NEF is set to 1)
An ALCI LU (if bit REC1NEF is set to 0)

If this bit is set to 0, the resource is one of:

An ALC terminal connected through NEF1 (if bit REC1NEF is set to 1)
An NEF1 LU (if bit REC1NEF is set to 0)

RETDSDD
For an IBM 3270 display terminal. Application has formatted the screen using SENDC D macro.

RETDUNS
The resource is unusable.

RETD3EX
The resource is a terminal that supports the IBM 3270 extended data stream.

Symbols defined for fields in the application section

C00RE defines the following symbols for fields:

REAAPRG
4-byte application program name. If the resource is an application, this field contains the name of the input message editor program for the application.

REAAMFM
1-byte message format that this application accepts.

The values that this field can contain:

REAAMFA
Message format is AMSG

REAAMFI
Message format is IMSG

REAAMFO
Message format is OMSG

REAAMFX
Message format is XMSG

REAASYS
1-byte minimum system state that in which this application can accept messages.

Symbols defined for bits in the application section

C00RE defines the following symbols for bits:

REAA1FM
The resource is an application that can process ALCS commands. That is, the application processes input messages with primary action code Z.

REAA1PE
The resource is an application that is permanently active. That is, the operator cannot make the application inactive.

Symbols defined for fields in the SLC link section

C00RE defines the following symbols for fields:

RESLCDE
1-byte translate code to be used for TYPE-B traffic over the SLC link.

The values that this field can contain:

RESLCDCA
Translate from/to ALC line code.

RESLCDCN
No translation required.

RESLCDC5
Translate from/to CCITT#5 (ASCII) code.

RESLCDC7
Translate from/to CCITT#2 code.

RESLHEN
2-byte entry address within the High Level Network (HLN) for this SLC link.

RESLPCL
1-byte SLC link protocol

The values that this field can contain:

RESLPCT1
SLC link is PRTCOL=TYPE1

RESLPCT2
SLC link is PRTCOL=TYPE2

RESLPCT3
SLC link is PRTCOL=TYPE3

Symbols defined for fields in the LU 6.1 link section

C0ORE defines the following symbols for fields:

RELKCRL
4-byte address of owning LU 6.1 link for a parallel session within an LU 6.1 link.

Symbols defined for fields in the WTTY section

C0ORE defines the following symbols for fields:

RETYCN2
8-byte LU name for the receive size of a WTTY full-duplex link.

Symbols defined for fields in the APPC section

C0ORE defines the following symbols for fields:

REAPPCL
1-byte APPC conversation protocol.

The values that this field can contain:

REAPPCT1
APPC connection is PRTCOL=TYPE1

REAPPCT2
APPC connection is PRTCOL=TYPE2

REAPPCT3
APPC connection is PRTCOL=TYPE3

REAPMOD
8-byte APPC mode name.

REAPSYM
8-byte APPC symbolic destination name.

REAPTPN
64-byte field containing the APPC partner TP name, left-justified.

REAPTPL
2-byte field containing the actual length of the partner TP name in field REAPTPN.

REAPLU1
8-byte APPC local LU name.

REAPLU2
17-byte APPC partner LU name for allocating outbound conversations.

REAPLU3
17-byte APPC partner LU name for allocating inbound conversations.

REAPRLU
17-byte APPC partner LU name - this is the current partner LU for the receive conversation.

REAPSLU
17-byte APPC partner LU name - this is the current partner LU for the send conversation.

Symbols defined for bits in the APPC section

C0ORE defines the following symbols for bits:

REAP2BA
APPC conversation type is mapped (0) or basic (1).

Symbols defined for fields in the TCP/IP section

C0ORE defines the following symbols for fields:

REIPCRL
4-byte field containing the address of the owning TCP/IP server connection.

REIPLPO
2-byte TCP/IP local port number.

REIPRPO
2-byte TCP/IP remote port number.

REIPRHA
16-byte field containing the TCP/IP remote host address in printable EBCDIC characters, left-justified and padded with blanks.

REIPRHB
4-byte TCP/IP remote host address in binary.

REIPMXC
4-byte field containing the maximum number of TCP/IP server connections (`COMDEF MAXCONN` parameter value).

REIPTMO
2-byte field containing the idle connection timeout (`COMDEF TIMEOUT` parameter value).

REIPPCL
1-byte TCP/IP application protocol.

The values that this field can contain:

REIPPCT0
TCP/IP connection is TERM=SERVER or TERM=CLIENT

REIPPCT1
TCP/IP connection is TERM=(SERVER,HTTP)

Symbols defined for bits in the TCP/IP section

C0ORE defines the following symbols for bits:

REIPCLI
This bit is on (1) for a TCP/IP client connection (TERM=CLIENT).

REIPSER
This bit is on (1) for a TCP/IP server connection (TERM=SERVER).

REIPDYN
This bit is on (1) for a TCP/IP dynamic server connection.

ALCS directly addressed monitor fields - CP0DA

Format

```
CP0DA [REG=reg]
```

REG=*reg*
Base register for DSECT addressability.

Description

Use the CP0DA DSECT macro to reference fields in the directly addressable monitor area. The DSECT name is CP0DA.

Symbols defined for fields

CP0DA defines the following symbols for fields:

CP0GLB
4-byte field containing the address of global area 1.

CP0G1L
4-byte field containing the size of global area 1.

CP0GA2
4-byte field containing the address of global area 2.

CP0G2L
4-byte field containing the size of global area 2.

CP0GA3
4-byte field containing the address of global area 3.

CP0G3L
4-byte field containing the size of global area 3.

CP0GA1
4-byte field containing the address of the directory of directories for the global area.

CP0STC
1-byte field containing the system state that we are currently changing to or zero if no state change is in progress.

The byte is split in to 2 parts of 4 bits each which have the identical contents. The first 4 bits define the FROM system state and the last 4 bits define the TO system state.

The following symbols can be used to test the first 4 bits of this byte.

CCCRAS
CRAS state

CCHALT
HALT state

CCIDLE
IDLE state

CCMESW
MESW state

CCNORM
NORM state

To use the symbols on the TO system state their values must be shifted 4 bits to the right.

CP0STI
1-byte field containing the current system state. The bits as defined for CP0STC are applicable here. The second 4 bits are always set to zero.

CP0USER
Start of an 128-byte area for user purposes. The user can define a macro to map the area.

ALCS APPC/MVS data area - DXCAPPCA

Format

```
DXCAPPCA [REG=reg]
```

REG=*reg*
Base register for DSECT addressability.

Description

Use the DXCAPPCA DSECT macro to reference fields in the APPC/MVS data area.. The DSECT name is DXCAPPCA.

Symbols defined for fields

DXCAPPCA defines the following symbols for fields:

APPCSSN
8-byte field containing the ALCS transaction scheduler name.

ALCS ECB descriptor - DXCECBD

If you need to access fields in the ECB descriptor, you can compute its address from the ECB address. To do this, you use the field IW0ECBD in the exit information table ("ALCS Installation-wide monitor exit information table - IW0IT" on page 328 describes this table). Note that you do not need to call the IW0IT macro.

The following example shows how you might do this in the installation-wide exit routine USRAPPC. Refer to "APPC/MVS (LU 6.2) exit - USRAPPC" on page 237 for more details.

```
L     R02,0(,R01)         LOAD ECB ADDRESS FROM PLIST
SH    R02,IW0ECBD         COMPUTE DESCRIPTOR ADDRESS
DXCECBD REG=R02           USE R02 AS DESCRIPTOR BASE
 :
 :  (you can access ECB descriptor fields here)
 :
DROP  R02                 DROP DESCRIPTOR BASE
```

Format

```
DXCECBD [REG=reg]
```

REG=*reg*
Base register for DSECT addressability.

Description

Use the DXCECBD DSECT macro to reference fields in the ECB descriptor. The DSECT name is DXCECBD.

Symbols defined for fields

DXCECBD defines the following symbols for fields:

CE3AUTH
1-byte field of bits indicating the authorization level of this ECB.

The following bits can be used in this byte:

CE3AUTHP
Prime CRAS

CE3AUTHS
System authorization

This is for ECBs which are internally created by ALCS and are not related to any resource.

CE3AUTH1
Alernate CRAS AT1-AT16

CE3AUTH2
Alernate CRAS AT17-AT255

CE3CHN
4-byte resource hold count. This excludes the record hold count (field CE3RHN).

CE3CRI
3-byte CRI indicating the origin of the ECB.

CE3PIA
4-byte Post Interrupt (PI) address. This is the address where the suspended ECB will be restarted.

CE3PLUNM
17-byte APPC/MVS partner LU name field.

CE3RHN
4-byte record hold count.

CE3SEQ
4-byte count of the number of sequential files currently assigned to this ECB.

CE3TOK
APPC/MVS user security token. The length of this field is available in field CE3TOKL.

CE3TOKL
2-byte field containing the length of the APPC/MVS user security token field CE3TOK.

CE3TPID
8-byte APPC/MVS TP_ID

CE3TPIDS
1-byte field of bits related to the APPC/MVS TP_ID.

The following bits can be used in this byte:

CE3TPIDI
TP-ID is in use.

CE3TPIDC
Inbound conversation TP-ID

CE3TPPRF
8-byte APPC/MVS TP profile.

CE3WIC
4-byte wait I/O counter. This is the number of I/Os currently in progress for this ECB.

ALCS WAS control and vector area DXCWASA - DXCWASA

Format

```
DXCWASA [REG=reg]
```

REG=*reg*
Base register for DSECT addressability.

Use the DXCWASA DSECT macro to test fields in the WAS control and vector table. The DSECT name is DXCWASA.

Entry control block - EB0EB

EB0EB is an intended interface. It is described fully in *ALCS Application Programming Reference - Assembler*.

ALCS I/O control block - IO0CB

Format

```
IO0CB [REG=reg]
```

REG=*reg*
Base register for DSECT addressability.

Description

Use the IO0CB DSECT macro to reference fields in the I/O control block. The DSECT name is IO0CB.

Symbols defined for fields

IO0CB defines the following symbols for fields:

IO0ECB
4-byte ECB address. Bit IO0FLGE should be set if this field is used.

IO0EVC
4-byte field containing either the MVS event control block or the address of the MVS event control block. This is dependent on bit IO0FLGA.

IO0EXT
4-byte post interrupt address.

IO0FLG
1-byte field of bits indicating the status of the fields within the IOCB.

The following bits can be used in this byte:

IO0FLGA
Field IO0EVC contains the address of the MVS event control block.

If this bit is not set, ALCS assumes that IO0EVC is the MVS event control block.

IO0FLGE
Field IO0ECB contains a valid ECB address.

IO0FLGR
Register save area IO0SAV is in use.

IO0SAV
72-byte (18 fullwords) standard MVS register save area.

If this is used bit IO0FLGR should be set.

IO0VAR
400-byte area for user usage.

You can define a user DSECT to map this area.

ALCS interrupt work area - IR0WA

Format

```
[label]  IR0WA [REG=reg]
```

REG=*reg*
Base register for DSECT addressability.

Description

Use the IR0WA DSECT macro to reference fields in the ALCS interrupt work area. The DSECT name is IR0WA.

Symbols defined for fields

IR0WA defines the following symbols for fields:

IR0BDA
4-byte field containing the address of the storage block list descriptor for the active ECB, or zero if no active ECB is related to the program interrupt.

IR0DO2
1-byte field of bits indicating the dump options in effect.

The following bits can be used in this byte:

IR0DO2AE
All entry storage indicator. If on, all the storage units will be dumped. If off, only the storage units belonging to the active ECB, if any, will be dumped.

IR0DO2GL
Global area indicator. If on, the global area will be dumped. If off, the global area will not be dumped.

IR0DO2IO
IOCB indicator. If on, the I/O Control Blocks (IOCBs) will be dumped. If off, the IOCBs will not be dumped.

IR0DO2PR
Program tables indicator. If on, the program tables will be dumped. If off, the program tables will not be dumped.

IR0DO2TF
Formatted monitor tables indicator. If on, the formatted monitor tables will be dumped. If off, the formatted monitor tables will not be dumped.

IR0DO2TU
Unformatted monitor tables indicator. If on, the unformatted monitor tables will be dumped. If off, the unformatted monitor tables will not be dumped.

IR0DO2VN
VFA buffers indicator. If on, all the VFA buffers will be dumped. If off, the VFA buffers will not be dumped.

IR0DO2VC
VFA control area indicator. If on, the VFA control area will be dumped. If off, the VFA control area will not be dumped.

IR0ECB
4-byte field containing the address of the active ECB or zero if no active ECB is related to the program interrupt.

IR0EOP
1-byte field of error options flag bits.

The following bits can be used in this byte:

IR0EOPC
Control dump indicator. If on, the program exception is an ALCS detected dump (CTL dump). If off, this program exception is an application detected dump (OPR dump).

IR0EOPM
User dump message indicator. If on, a user dump message is supplied with the dump. General purpose register 0 (R00) at the time of the exception (in IR0R00) will hold the address of the message. The first byte addressed by R00 is the length of the message followed by the message text.

IR0EOPR
Return dump indicator. If on, this dump is a return dump, if off this dump is an exit dump.

IR0EOPS
SERRC indicator. If on, the program exception is caused by an SERRC, SNAPC, or CPDMP issued by ALCS or an application program. If off, this program exception is a hardware generated program exception like an protection exception or an addressing exception.

IR0EOPX
Catastrophic dump indicator. If on, this program exception is catastrophic (ALCS terminates). If off ALCS will continue normally after the program exception.

IR0FPR
32-byte field containing the contents of all 4 floating point registers at the time of the program interrupt.

IR0FP0
8-byte field containing the contents of floating point register 0 (FP0) at the time of the program interrupt.

IR0FP2
8-byte field containing the contents of floating point register 2 (FP2) at the time of the program interrupt.

IR0FP4
8-byte field containing the contents of floating point register 4 (FP4) at the time of the program interrupt.

IR0FP6
8-byte field containing the contents of floating point register 6 (FP6) at the time of the program interrupt.

IR0ILC
2-byte field containing the instruction length code (ILC) at the time of the program interrupt.

IR0IRC
2-byte field containing the interruption code (IRC) at the time of the program interrupt.

IR0MSG
A variable length field containing the address of the user dump message if any. The length of this field is given by label IR0MSL.

IR0MSGDD
44-byte field containing the data set name (DSN) of the dump data set where this dump is written in printable text, or blanks if the dump is a NODUMP.

IR0MSGNR
6-byte field containing the system error sequence number in printable text.

IR0MSGOX
4-byte offset within the program related to the active ECB at the time of the program interrupt (in printable text). This field is only relevant if field IR0MSGPG is set.

IR0MSGSC
6-byte field containing the system error number in printable text.

IR0MSGPG
5-byte field which can contain the fixed text PR0G-. If this text is set some other fields are relevant.

IR0MSGPN
4-byte program name related to the active ECB at the time of the program interrupt. This field is only relevant if field IR0MSGPG is set.

IR0MSGPS
4-byte field which can contain the fixed text PSW-. If this text is set some other fields are relevant.

IR0MSGPW
16-byte Program Status Word (PSW) containing the PSW at the time of the program interrupt, in printable text. This field is only relevant if field IR0MSGPS is set.

IR0MSGTA
6-byte field containing the CRI of the resource related to the program exception in printable text. This field is only applicable depending on the contents of field IR0MSGTT.

IR0MSGTM
8-byte field containing the time of the program exception in printable text in the format HH.MM.SS.

IR0MSGTN
8-byte field containing the CRN of the resource related to the program exception. This field is only applicable depending on the contents of field IR0MSGTT.

IR0MSGTT
4-byte field containing the text of either CRN- in which case the field IR0MSGTN is applicable or CRI- in which case field IR0MSGTA is applicable.

IR0MSGTY
4-byte field containing the text of either CTL- in case of a monitor-detected error or 0PR- in case of an application-detected error.

IR0MSGUT
A variable length field containing the user dump message if any. The length of this field is given by label IR0MSL.

IR0MSGVD
6-byte field containing the volume serial number (VSN) of the volume where this dump is written in printable text, or blanks if the dump is a NODUMP.

IR0MSL
2-byte field containing the length of the dump message.

IR0PSW
8-byte field containing the contents of the Program Status Word (PSW) at the time of the program interrupt.

IR0REG
64-byte field containing the contents of all 16 general purpose registers at the time of the program interrupt.

IR0R00
4-byte field containing the contents of general purpose register 0 (R00) at the time of the program interrupt.

IR0R01
4-byte field containing the contents of general purpose register 1 (R01) at the time of the program interrupt.

IR0R02
4-byte field containing the contents of general purpose register 2 (R02) at the time of the program interrupt.

IR0R03
4-byte field containing the contents of general purpose register 3 (R03) at the time of the program interrupt.

IR0R04
4-byte field containing the contents of general purpose register 4 (R04) at the time of the program interrupt.

IR0R05
4-byte field containing the contents of general purpose register 5 (R05) at the time of the program interrupt.

IR0R06
4-byte field containing the contents of general purpose register 6 (R06) at the time of the program interrupt.

IR0R07
4-byte field containing the contents of general purpose register 7 (R07) at the time of the program interrupt.

IR0R08
4-byte field containing the contents of general purpose register 8 (R08) at the time of the program interrupt.

IR0R09
4-byte field containing the contents of general purpose register 9 (R09) at the time of the program interrupt.

IR0R10
4-byte field containing the contents of general purpose register 10 (R10) at the time of the program interrupt.

IR0R11
4-byte field containing the contents of general purpose register 11 (R11) at the time of the program interrupt.

IR0R12
4-byte field containing the contents of general purpose register 12 (R12) at the time of the program interrupt.

IR0R13
4-byte field containing the contents of general purpose register 13 (R13) at the time of the program interrupt.

IR0R14
4-byte field containing the contents of general purpose register 14 (R14) at the time of the program interrupt.

IR0R15
4-byte field containing the contents of general purpose register 15 (R15) at the time of the program interrupt.

IR0SDWA
4-byte field containing the address of the MVS System Diagnostic Work Area (SDWA) or zero if no SDWA was obtained.

IR0SRC
3-byte field containing the system error number in hexadecimal.

ALCS Installation-wide monitor exit information table - IW0IT

Format

```
        IW0IT [REG=reg]
```

REG=R12
Base register for DSECT addressability.

Be aware that this DSECT is already generated by the DXCUHDR macro with R12 as base register.

Description

Use the IW0IT DSECT macro to reference fields in the installation-wide monitor exit table. The DSECT name is IW0IT.

Symbols defined for fields

IW0IT defines the following symbols for fields:

IW0CDS
4-byte field containing the address of the system configuration table (CS0DT).

IW0DATAB
4-byte field containing the address of the direct addressable storage table (CP0DA). This is described in “ALCS directly addressed monitor fields - CP0DA” on page 319.

IW0DESP
2-byte field containing the displacement between the ECB prefix and the ECB descriptor which is situated before the ECB prefix.

IW0DESS
2-byte field containing the displacement between the start of the storage unit holding the ECB and the ECB descriptor which is situated before the storage unit.

IW0ECBCT
4-byte field containing the ACTIVE/DEFER ECB COUNTER ADDRESS

IW0ECBD
2-byte field containing the displacement between the ECB and the ECB descriptor which is situated before the ECB.

IW0ECBP
2-byte field containing the displacement between the ECB and the ECB prefix which is situated before the ECB.

IW0ECBS
2-byte field containing the displacement between the ECB and the start of the storage unit in which the ECB is located.

IW0IONDX
4-byte field containing the I/O CONTROL BLOCK INDEX ADDRESS

IW0SUNDX
4-byte field containing the STORAGE UNIT INDEX ADDRESS

IW0USER
Start of an 2048 (2K) area for usage by the installation-wide exits. The user can define a macro to map this area.

The following symbols can be used in combination with callable service UCOMGET:

- IW0CRI
- IW0CRN
- IW0LEID
- IW0NEXT
- IW0SLCID
- IW0UDATA

The following symbols can be used in combination with callable service UCNTINC:

- IW0IMSG
- IW0OMSG

The following symbols can be used in combination with callable services UMSGT1 and UMSGT2:

- IW0INP

The following fields can be used in combination with callable service UECBGET:

- IW0CREAT
- IW0NONE
- IW0RECEI

The following fields can be used in combination with callable service UECBQUE:

- IW0DEFER
- IW0DELAY
- IW0INPUT
- IW0READY

The following fields can be used in combination with callable service UIOBGET:

- IW0LOAD
- IW0NONE

The following symbols can be used in combination with callable service USTRGET and USTRREL:

- IW0PFIX
- IW0PROT
- IW0RM24

The following symbols can be used in combination with callable service USTRVAL:

- IW0ENTRY
- IW0TABLE
- IW0GLOBL

The following symbols can be used in combination with callable service UWSEQ:

- IW0DCR
- IW0DIA
- IW0LOG
- IW0LOGB
- IW0USR

Get information about a DASD record or record type - RONIC

RONIC is an intended interface. It is described fully in *ALCS Application Programming Reference - Assembler*.

ALCS record-control fields - RS0RS

RS0RS is an intended interface. It is described fully in *ALCS Application Programming Reference - Assembler*.

ALCS services for installation-wide monitor exits

ALCS provides macros and callable services to implement commonly required functions (get an ECB, get an IOCB, and so on). You must use these macros and callable services in the installation-wide monitor exits, because the ALCS monitor services are **not** directly available to installation-wide monitor exits.

Macros you can call in installation-wide monitor exits

The available macros are:

APIDC
Performance monitor interface.

COMCC
Update communication resource information.

CPDMP
Take a control system error dump.

DECBC
DECB management.

DXCPKEY
Change PSW key.

DXCSAVE
Save area management macro.

GLOBZ
Access a global area.

TIMEC
Get the ALCS time and date.

WTOPC
Send a message.

ALCS performance monitor interface - APIDC

Use this macro as described in *ALCS Application Programming Reference - Assembler* with the following restriction.

Restriction

ECBADDR=YES is not supported. Register 0 (RAC) is corrupted.

Update communication resource entry - COMCC

ALCS Application Programming Reference - Assembler describes this macro.

You can use this macro with some restrictions. You can use the following for fields or bits:

- RECOARN
- REC1AAA
- REC1CST
- REC1LOG
- RETDACK
- RETDARC

- RETDSDD

Restriction

The FIELD= parameter is not supported in register notation.

Take system error dump - CPDMP

Use the CPDMP macro to take a system error dump. See "Problem determination in ALCS" in *ALCS Operation and Maintenance* for information about system error dumps.

Format

```
[label]  CPDMP error_code
                        [,EXIT=NO]
                        [,ECB={YES|NO}]
                        [,MSG={NO|YES|'text'}]
                        [,DUMP={SEL|ALL|NO}]
```

Where:

label
Any valid Assembler label.

error_code
A unique six hexadecimal-digit code that indicates which condition caused the dump to be taken. Specify a code in the range X'000000' through X'000FFF'.

See "System error codes: 000000-000FFF" in *ALCS Messages and Codes* for a list of system error codes that ALCS uses.

EXIT=NO
The action to take following the dump.

NO
Return control to the next sequential instruction after the CPDMP macroinstruction.

ECB={YES|NO}

YES
The system error dump is associated with a particular active entry.

NO
The system error dump is not associated with a particular active entry.

ECB=YES is allowed only for the following installation-wide monitor exits:

- USRCOM2
- USRCOM4
- USRMQI1
- USRSQL1
- USRSQL2
- USRSVC
- USRTCP1

MSG={NO|YES|'*text*'}
Indicates whether the online monitor appends an optional user message to the standard message that it generates when it takes the system error dump.

NO
No message is appended.

YES
General register 0 (RAC) contains the address of a field that contains the message with the following format:

Byte 0
Contains the length of the field (in binary).

Successive bytes
Contain the text of the message. The message text must not contain control characters such as new line (#CAR).

text
Message text.

DUMP={SEL|ALL|NO}

SEL
Dump according to the current CTL system error option settings (the ZDSER command displays the current settings).

ALL
Override the current system error options and dump as if all options are selected. Also include the online program tables.

NO
Generate the system error dump message but do not include any dump data.

Loss of Control

If there is an active entry, this macro does not cause it to lose control.

Register Usage

If you specify MSG=YES, no registers are used. If you specify MSG=*text*, ALCS uses general register 0 (RAC) and does not restore it.

Usage Notes

Use CPDMP when an error condition is detected. CPDMP expands to give an specification exception that the online monitor recognizes as a CPDMP request.

Use the diagnostic file processor offline program (DXCDTP) to format and print the system error dump (see "Running the ALCS diagnostic file processor" in *ALCS Operation and Maintenance*).

Note: Some installation-wide monitor exits do not tolerate CPDMP. Do not include CPDMP in the following exits:

- USRAPPC
- USRCOM6
- USRCOM7
- USRDMP
- USRPCH
- USRTCP2
- USRTCP4

DECB management - DECBC

Use this macro as described in *ALCS Application Programming Reference - Assembler*, with the following restriction.

Restriction

ALCS uses general registers 1 and 2 (RG1 and RGA) and does not restore them.

Change PSW key - DXCPKEY

Format

```
[label]  DXCPKEY PSW,SET,KEY={TABLES|ENTRY|GLOBAL1}
```

Where:

KEY={TABLES|ENTRY|GLOBAL1}
Key to be moved to the PSW. Valid keys are:

TABLES
To update monitor controlled data fields.

ENTRY
To update storage units which hold the ECBs and associated blocks.

GLOBAL1
To update global areas 1 or 3.

Description

Use the DXCPKEY macro to change the Program Status Word (PSW) key.

ALCS save area management macro - DXCSAVE

Restriction:

You must use the forms of DXCSAVE in the sample installation-wide monitor exits. Do not use any other form of this macro.

Format

```
[label] DXCSAVE PUSH
                          ,ID={USER|UPCH|UDMP|USR1|USR2}
                          ,SIZE=512,WORKREG=R02
```

Stacks the save area and registers of the calling routine in ALCS.

```
[label]  DXCSAVE POP,RESTORE,EXCEPT=((R15))
```

Restores the save area and registers (except R15) of the calling routine in ALCS.

```
[label]  DXCSAVE SRBSAVE,ID=USER,SIZE=512,WORKREG=R06
```

Stacks the save area and registers of the calling routine in ALCS.

```
[label]  DXCSAVE FREESRB,RESTORE,EXCEPT=((R15))
```

Restores the save area and registers (except R15) of the calling routine in ALCS.

Where:

ID=USER
Most installation-wide monitor exits use this form of DXCSAVE.

ID=UPCH
USRPCH uses this form of DXCSAVE.

ID=UDMP
USRDMP uses this form of DXCSAVE.

ID=USR1
USRRTN1 uses this form of DXCSAVE.

ID=USR2
USRRTN2 uses this form of DXCSAVE.

The following installation-wide monitor exits do not use the DXCSAVE macro:

- USRCOM5
- USRTAB1 through USRTAB6
- USRFAR

The following installation-wide monitor exits use the SRBSAVE and FREESRB parameters:

- USRCOM6
- USRCOM7

Access a global area - GLOBZ

Use this macro as described in *ALCS Application Programming Reference - Assembler*.

Get the ALCS time and date - TIMEC

Use this macro as described in *ALCS Application Programming Reference - Assembler*, with the following restriction.

Restriction

The AREA= parameter is mandatory and must point to a field of the appropriate size in tables key.

Write to operator - WTOPC

Use this macro as described in *ALCS Application Programming Reference - Assembler*, with the following restriction.

Restriction

In execute form, the *plist* label is mandatory.

Entry conditions for callable services

All callable services are invoked as follows:

```
DXCSERV Uservice_name,PARM=({(Rnn)|label|0},...)
```

Where

U*service_name*
One of:

UCNTINC
Increment system counter.

UCOMCHG
Change a CRN in a communication table item.

UCOMGET
Obtain a communication table item address.

UDLEVGET
Obtain a storage block and attach it to a DECB.

UDLEVREL
Release a storage block attached to a DECB.

UDLEVVAL
Validate a DECB storage level.

UDISP
Branch to dispatcher.

UECBGET
Obtain an ECB.

UECBQUE
Queue an ECB.

UECBREL
Release an ECB.

UECBVAL
Validate an ECB address.

UFREE
Free heap storage

UHEAP
Allocate heap storage

UIOBGET
Obtain an IOCB.

UIOBQUE
Queue an IOCB.

UIOBREL
Release an IOCB.

ULEVGET
Obtain a storage block and attach it to an ECB.

ULEVREL
Release a storage block attached to an ECB.

ULEVVAL
Validate an ECB storage level.

UMLEVVAL
Validate an ECB monitor storage level.

UMSGT1
Trace a message of a particular communications resource

UMSGT2
Trace a message of a particular communications resource

UPROGF
Find program address.

URTN1
Call USRRTN1.

URTN2
Call USRRTN2.

USTRECB
Validate storage belonging to an ECB

USTRGET
Obtain MVS virtual storage.

USTRREL
Release MVS virtual storage.

USTRVAL
Validate an MVS virtual storage address.

UTAB1
Find a translate table.

UTAB6
Find a translate table.

UTAB7
Find a translate table for `ASCIC`.

UTAB10
Find a translate table for `ASCIC`.

UWSEQ
Write to a system-sequential file.

PARM=
One or more parameters to pass between the callable service and the user code. The parameters include the passed and returned parameters. Each parameter occupies a fullword. Specify `0` to initialize each unused parameter.

Attention

You must specify **all** passed and returned parameters required by a callable service, otherwise the call may fail unpredictably. The format is a standard MVS `CALL` parameter list.

Figure 59 on page 337 gives an overview of the calling process for a service with two input parameters and three output parameters.

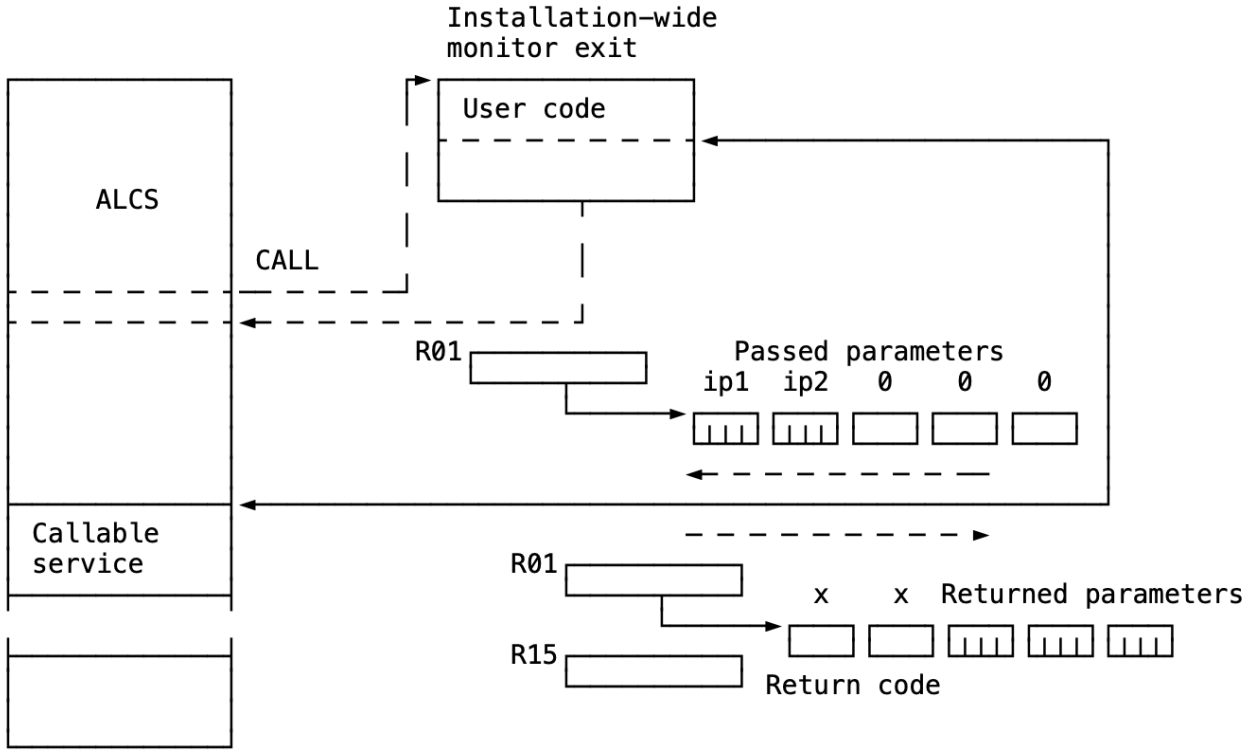


Figure 59. Callable service linkage conventions

The callable services use the standard MVS CALL macro as follows:

- R14 for the return address
- R15 for the return code
- R01 to point to the start of the parameter list
- R00 can be corrupted

Notes:

1. The output parameters are offset by one fullword for each input parameter.
2. Unless specifically mentioned, all callable services are entered (and return) in tables storage key.

Register usage on return from the callable service

R00
Can be overwritten by the MVS CALL macro.

R01
Points to the start of the parameter list.

R14
The return address to the user code.

R15
Return code from the service, it is set to:

0
Successful completion

<>0
Unsuccessful completion.

All other registers are the same as on entry to the callable service.

Increment system counter -- UCNTINC

param_1
The address of a 1-byte indicator field. Use one of the following bit symbols to indicate which type of information is passed to UCNTINC:

IW0IMSG
The number of input messages.

IW0OMSG
The number of output messages.

UCNTINC returns with the following conditions:

R15=0
The count is incremented.

```
        This UCNTINC call increments the number of received
        input messages
        Label CNTIND is defined as DC 'X'00' in the user
        monitor save area
        MVI   CNTIND,IW0IMSG      SET INPUT MESSAGE RECEIVED
        SPACE 1
        DXCSERV UCNTINC,PARM=(CNTIND)
        SPACE 1
```

Figure 60. Example: UCNTINC incrementing the system counter

Change a CRN in a communication table entry -- UCOMCHG

param_1
The address of the communication table entry for the resource to change.

param_2
The address of an 8-byte field containing either:

- The CRN to remove from the communication table entry
- Zero, if there is nothing to remove

param_3
The address of an 8-byte field containing either:

- The CRN to add to the communication table entry
- Zero, if there is nothing to add

UCOMCHG returns with the following conditions:

R15=0
The change is successful.

R15=4
The CRN cannot be added because it already exists.

R15=8
The CRN cannot be removed because it does not exist.

R15=12
The CRN cannot be added because an ALCS table is full. Or, the CRN cannot be removed because it does not match the resource that *param_1* specifies.

R15=16
The resource cannot be changed because it is active or logon is in progress.

R15=20
The CRN cannot be removed because it does not match any existing resource. Or, *param_1* is omitted.

R15=24
Unable to obtain lock for communication table entry.

```
        This UCOMCHG changes a CRN in the communication
        tables.
        The old CRN is taken from the communication table
        item while the new CRN is taken from an 8 bytes field
        which is addressed by register R02.
        The communication table address is in register R06.
        SPACE 1
        CO0RE REG=R06             COMMUNICATION TABLE ITEM DSECT
        DXCSERV UCOMCHG,PARM=((R06),RECOCRN,(R02))
        LTR   R15,R15             HAS THE CHANGE TAKEN PLACE
        BNZ   ERROR               NO - BRANCH
        SPACE 1
        DROP  R06                 DROP COMMUNICATION TABLE BASE
```

Figure 61. Example usage of UCOMCHG

Obtain a communication table entry address -- UCOMGET

param_1
Specify zero.

param_2
The address of a 1-byte indicator field. Use one of the following bit symbols to indicate which type of information is passed to UCOMGET:

IW0CRI
The CRI is supplied.

IW0CRN
The CRN is supplied.

IW0NEXT
Point to the communication table entry for the next resource (after the specified CRI).

IW0LEID
The LEID is supplied.

IW0UDATA
Return the address of the user part of the communication table entry.

param_3
The address of an input area for the function:

- 3-byte CRI for the CRI or next function
- 8-byte CRN for the CRN function
- 3-byte LEID for the LEID function

param_4
Specify zero if IW0UDATA is included in **param_2**.

```
*        This UCOMGET obtains the communication table address
*        of a device using the CRI as a search argument.
*        Label COMIND is defined as DC X'00' in the user monitor
*        save area.
*        ECB label EBROUT holds a 3 byte CRI.
*        Label PARM1 is equated to 0(R01).
         MVI   COMIND,IW0CRI       SET CRI AVAILABLE FOR SEARCH
         SPACE 1
         DXCSERV UCOMGET,PARM=(0,COMIND,EBROUT)
         SPACE 1
         LTR   R15,R15             IS THIS DEVICE KNOWN TO ALCS
         BNZ   ERROR               NO - BRANCH ERROR
         SPACE 1
         L     R14,PARM1           LOAD COMMS TABLE ADDRESS
```

Figure 62. Example: UCOMGET using a CRI

```
*        This UCOMGET obtains the communication table address
*        of a device using the CRN as a search argument.
*        It also obtains the address of the start of the user
*        part of the communication table for this device.
*        Label COMIND is defined as DC X'00' in the user monitor
*        save area.
*        Label LUNAME holds an 8 byte CRN padded to the right with
*        blanks.
*        Label PARM1 is equated to 0(R01).
         MVI   COMIND,IW0CRN+IW0UDATA  SET CRN AVAILABLE FOR SEARCH
         SPACE 1
         DXCSERV UCOMGET,PARM=(0,COMIND,LUNAME,0)
         SPACE 1
         LTR   R15,R15             IS THIS DEVICE KNOWN TO ALCS
         BNZ   ERROR               NO - BRANCH ERROR
         SPACE 1
         L     R14,PARM1           LOAD COMMS TABLE ADDRESS
         L     R15,PARM4           LOAD ADDRESS OF USER AREA
```

Figure 63. Example: UCOMGET using a CRN

```
*        This UCOMGET obtains the communication table address
*        of a device using the LEID as a search argument.
*        Label COMIND is defined as DC X'00' in the user
*        monitor save area.
*        Label CM5MLD holds a 3-byte LEID as part of an input message.
*        Label PARM1 is equated to 0(R01).
         MVI   COMIND,IW0LEID      SET LEID AVAILABLE FOR SEARCH
         SPACE 1
         DXCSERV UCOMGET,PARM=(0,COMIND,CM5MLD)
         SPACE 1
         LTR   R15,R15             IS THIS DEVICE KNOWN TO ALCS
         BNZ   ERROR               NO - BRANCH ERROR
         SPACE 1
         L     R14,PARM1           LOAD COMMS TABLE ADDRESS
```

Figure 64. Example: UCOMGET using an LEID

UCOMGET returns with the following conditions:

R15=0
The communication table entry exists.

param_1
The address of the communication table entry.

param_4
The address of the start of the user part of the communication table entry, if IW0UDATA is included in **param_2**.

R15=4
The communication table entry does not exist.

R15=8
Internal error detected.

SLC-ID option

The SLC ID option uses the following parameters (`param_3` through `param_7`):

param_1
Specify zero.

param_2
The address of a 1-byte indicator field. Use one of the following bit symbols to indicate which type of information is passed to UCOMGET:

IW0SLCID
The SLCID is supplied.

IW0UDATA
Return the address of the user part of the communication table entry.

param_3
The address of the 3-byte CRI of the SLC link.

param_4
The address of the 2-byte HEX of the remote terminal.

param_5
The address of the 1-byte TCID of the remote terminal.

param_6
The address of the 1-byte IA of the remote terminal.

param_7
The address of the 1-byte TA of the remote terminal.

param_8
Specify zero if IW0UDATA is included in **param_2**.

```
*        This UCOMGET obtains the communication table address
*        of an SLC terminal.
*        Label COMIND is defined as DC X'00' in the user monitor
*        save area.
*        Label LK4CRI holds a 3-byte CRI of the SLC link on which
*        this terminal is defined.
*        Label CM8HEN holds the 2-byte HEX of the terminal.
*        Label CM8TXH holds the 1-byte TCID of the terminal.
*        Label CM8TXH+1 holds the 1-byte IA of the terminal in line code.
*        Label CM8TXH+2 holds the 1-byte TA of the terminal in line code.
*        Label PARM1 is equated to 0(R01).
         MVI   COMIND,IW0SLCID     SET SLCID AVAILABLE FOR SEARCH
         SPACE 1
         DXCSERV UCOMGET,                                              -
               PARM=(0,COMIND,LK4CRI,CM8HEN,CM8TXH,CM8TXH+1,CM8TXH+2)
         SPACE 1
         LTR   R15,R15             IS THIS DEVICE KNOWN TO ALCS
         BNZ   ERROR               NO - BRANCH ERROR
         SPACE 1
         L     R14,PARM1           LOAD COMMS TABLE ADDRESS
```

Figure 65. Example: UCOMGET for an SLC terminal

UCOMGET with the SLC-ID option returns with the following conditions:

R15=0
The communication table entry exists.

param_1
The address of the communication table entry.

param_8
The address of the start of the user part of the communication table entry, if IW0UDATA is included in **param_2**.

R15=4
The communication table entry does not exist.

R15=8
Internal error detected.

Obtain a storage block and attach it to a DECB -- UDLEVGET

param_1
The address of the ECB.

param_2
The address of the DECB.

param_3
The address of a 2-byte field containing the block size code. Use the same conventions as the GETCC monitor-request macro.

See *ALCS Application Programming Reference - Assembler* for a description of the GETCC monitor-request macro.

UDLEVGET returns with the following conditions:

R15=0
A block is obtained.

R15=4
A block is not obtained.

```
*        This UDLEVGET obtains a storage block of size L2 attached to the DECB
*        R09 - has the ECB address
*        R14 - has the DECB address
*        Label SIZE is defined as DC AL2(L2) in the program
         DXCSERV UDLEVGET,PARM=((R09),(R14),SIZE)
         SPACE 1
         LTR   R15,R15             DID WE GET THE BLOCK
         BNZ   ERROR               NO - BRANCH
         SPACE 1
```

Figure 66. Example: UDLEVGET

Release a storage block attached to a DECB -- UDLEVREL

param_1
The address of the ECB.

param_2
The address of the DECB.

UDLEVREL returns with the following conditions:

R15=0
The block is released.

```
*        This UDLEVREL releases the storage block attached to the DECB
*        R09 - has the ECB address
*        R14 - has the DECB address
         DXCSERV UDLEVREL,PARM=((R09),(R14))
         SPACE 1
*                                       NO NEED TO CHECK RETURN CODE
```

Figure 67. Example: UDLEVREL

Validate a DECB storage level -- UDLEVVAL

param_1
The address of the ECB.

param_2
The address of the DECB.

UDLEVVAL returns with the following conditions:

R15=0
The storage level is valid and in use.

R15=4
The storage level is not in use.

R15=8
The storage level is corrupted.

```
*        This UDLEVVAL validates the block attached to the DECB
*        R09 - has the ECB address
*        R14 - has the DECB address
         DXCSERV UDLEVVAL,PARM=((R09),(R14))
         SPACE 1
         B     *+4(R15)            CHECK THE RETURN CODE
         B     USEIT               0 - LEVEL IN USE AND OK
         B     EMPTY               4 - LEVEL NOT IN USE
         B     ERROR               8 - LEVEL CORRUPTED
         SPACE 1
```

Figure 68. Example: UDLEVVAL

Branch to dispatcher -- UDISP

This service branches to the ALCS dispatcher (CPU loop) and does not return.

Note: You must use this service **only** in post-interrupt routines which are queued with UIOBQUE or UECBQUE.

```
         DXCSERV UDISP             BRANCH TO CPU LOOP
         SPACE 1
```

Figure 69. Example: UDISP

Obtain an ECB -- UECBGET

param_1
The address of the *creating* ECB, or zero if there is no creating ECB.

param_2
Specify zero.

param_3
The address of a 1-byte minimum-requirements field. Use one of the following labels to specify the minimum requirements:

IW0NONE
No minimum requirements.

IW0CREAT
The same requirements as ALCS before it creates a new ECB.

IW0RECEI
The same requirements as ALCS before it issues a new VTAM RECEIVE.

param_4
The address of a communication table entry for the originating resource for the entry, or zero if there is no originating resource.

UECBGET returns with the following conditions:

R15=0
An ECB is dispensed:

param_2
The address of the new ECB.

R15=4
No ECBs are available.

```
         R09 - has old creating ECB address
         Minimum required to obtain new ECB is the RECEIVE level
         Label PARM2 is equated to 4(R01)

         DXCSERV UECBGET,PARM=((R09),0,IW0RECEI,0)
         LTR   R15,R15             DO WE HAVE A NEW ECB
         BNZ   ERROR               NO - BRANCH
         SPACE 1
         L     R14,PARM2           LOAD NEW ECB ADDRESS
```

Figure 70. Example: UECBGET

Queue an ECB on a work list -- UECBQUE

param_1
The ECB address to queue.

param_2
The address of the post-interrupt routine or zero.

param_3
The address of an ALCS work list. Use one of the following labels to represent the address of the ALCS work list.

IW0READY
The address of the *ready* list.

IW0INPUT
The address of the *input* list.

IW0DELAY
The address of the *delay* list.

IW0DEFER
The address of the *defer* list.

param_4
The address of a 4-byte field which holds the name of a program to execute if the post-interrupt address (`param_2`) is zero.

param_5
The address of a communication table entry for the originating resource for the entry, or zero if there is no originating resource.

UECBQUE returns with the following conditions:

R15=0
The ECB is queued.

```
*        This UECBQUE will queue the ECB on the ready list
*        with ECBPOSTI as PI address
*        R14 - has the ECB address to be queued

         DXCSERV UECBQUE,PARM=((R14),ECBPOSTI,IW0READY,0,0)
         SPACE 1
                                NO NEED TO CHECK RETURN CODE
```

Figure 71. Example: UECBQUE to the ready list

```
*        This UECBQUE will queue the ECB on the input list
*        to be processed by program 'ABCD'
*        R14 - has the ECB address to be queued
*        R02 - has the address of the item (in the communication table)
*              for the input resource
*        Label PROGNAME is defined as DC C'ABCD' in the program

         DXCSERV UECBQUE,PARM=((R14),0,IW0INPUT,PROGNAME,(R02))
         SPACE 1
                                NO NEED TO CHECK RETURN CODE
```

Figure 72. Example: UECBQUE to the input list

Release an ECB -- UECBREL

param_1
The ECB address to release.

UECBREL returns with the following conditions:

R15=0
The ECB is released.

param_1
Set to zero, the address is no longer valid.

```
         R14 - has the ECB address to be released

         DXCSERV UECBREL,PARM=((R14))
         SPACE 1
                                NO NEED TO CHECK RETURN CODE
```

Figure 73. Example: UECBREL

Validate an ECB -- UECBVAL

param_1
The ECB address to validate.

UECBVAL returns with the following conditions:

R15=0
The address is a valid ECB address.

R15=4
The address is not a valid ECB address.

```
        R06 - has the ECB address to be validated

        DXCSERV UECBVAL,PARM=((R06))
        SPACE 1
        LTR   R15,R15             CAN WE USE THIS ECB ADDRESS
        BNZ   ERROR               NO - BRANCH
        SPACE 1
```

Figure 74. Example: UECBVAL

Free heap storage -- UFREE

param_1
The address of the ECB.

param_2
The address of the heap storage area to be freed.

UFREE returns with the following conditions:

R15=0
The heap storage was freed.

R15=8
The ECB address is not valid.

```
        DXCSERV UFREE,PARM=((R09),HEAPADR)
        SPACE 1
        LTR   R15,R15             CHECK THE RETURN CODE
        BNZ   ERROR               BRANCH IF STORAGE NOT RELEASED
```

Figure 75. Example: UFREE

Allocate heap storage -- UHEAP

param_1
The address of the ECB.

param_2
The address of a 4-byte length field.

param_3
Specify zero.

UHEAP returns with the following conditions:

R15=0
The heap storage was allocated. *param_3* is the address of the allocated heap storage area.

R15=4
The heap storage was not allocated.

R15=8
The ECB address is not valid.

```
DXCSERV UHEAP,PARM=((R09),HEAPLEN,0)
SPACE 1
B     *+4(R15)            CHECK THE RETURN CODE
B     VALID               0 - STORAGE OBTAINED
B     ERROR               4 - STORAGE NOT OBTAINED
B     ERROR               8 - ECB ADDRESS NOT VALID
```

Figure 76. Example: UHEAP

Obtain an IOCB -- UIOBGET

param_1
Specify zero.

param_2
The address of a 1-byte minimum-requirements field. Use one of the following labels to specify the minimum requirements:

IW0NONE
No minimum requirements.

IW0LOAD
The same requirements as a *controlled* dispense of an IOCB by ALCS.

This IOCB is flagged as a USER IOCB type in any ALCS dump.

UIOBGET returns with the following conditions:

R15=0
The IOCB is dispensed:

param_1
The address of the IOCB that is dispensed.

R15=4
An IOCB is not available.

```
Minimum required to obtain new IOCB is the controlled load
level as used within ALCS
PARM1 is equated to 0(R01)

DXCSERV UIOBGET,PARM=((0,IW0LOAD)
LTR   R15,R15             DO WE HAVE AN IOCB
BNZ   ERROR               NO - BRANCH
SPACE 1
L     R14,PARM1           LOAD IOCB ADDRESS
```

Figure 77. Example: UIOBGET using IW0LOAD

Queue an IOCB -- UIOBQUE

param_1
The address of the IOCB to queue.

You must post the MVS event control block (see note) to ensure that the IOCB is removed from the IOCB work list.

You can do this either:

- Before calling UIOBQUE (if the process does not depend on an external event)
- After the IOCB is queued (the external event posts the MVS event control block).

Note: The field IO0EVC contains either the MVS event control block, or the address of the MVS event control block. See "ALCS I/O control block - IO0CB" on page 323 for details of this field.

UIOBQUE returns with the following conditions:

R15=0
The IOCB is queued.

```
         UIOBQUE queues the IOCB on the IOB list.
         R02 - holds the IOCB address to be queued.
         The post-interrupt address must be stored in the
         IO0EXT field in IO0CB.

         OI    IO0EVC+(ECBCC-ECB),ECBPOST  SET THE EVCB AS POSTED
         DXCSERV UIOBQUE,PARM=((R02))
         SPACE 1
                                  NO NEED TO CHECK RETURN CODE
```

Figure 78. Example: UIOBQUE to IOB list

Release an IOCB -- UIOBREL

param_1
The address of the IOCB to release.

UIOBREL returns with the following conditions:

R15=0
The IOCB is released.

```
         R14 - has IOCB address to be released

         DXCSERV UIOBREL,PARM=((R14))
         SPACE 1
                                  NO NEED TO CHECK RETURN CODE
```

Figure 79. Example: UIOBREL

Obtain a storage block -- ULEVGET

This service obtains a storage block and attaches it to an ECB.

param_1
The address of the ECB

param_2
The address of a 2-byte level indicator. Use the same conventions as the GETCC monitor-request macro.

param_3
The address of a 2-byte block-type indicator. Use the same conventions as the GETCC monitor-request macro.

See *ALCS Application Programming Reference - Assembler* for a description of the GETCC monitor-request macro.

Note: ULEVGET does not initialize the storage block to binary zeros.

ULEVGET returns with the following conditions:

R15=0
The block was obtained

R15=4
The block was not obtained because:

- The maximum storage for this ECB is exceeded
- There are no more storage units available.

```
        This ULEVGET obtains a storage block of size L2
        on ECB level D6
        R09 - has the ECB address
        Label LEVEL is defined as DC AL2(D6) in the program
        Label SIZE is defined as DC AL2(L2) in the program

        DXCSERV ULEVGET,PARM=((R09),LEVEL,SIZE)
        SPACE 1
        LTR   R15,R15             DID WE GET THE BLOCK
        BNZ   ERROR               NO - BRANCH
        SPACE 1
```

Figure 80. Example: ULEVGET

Release a storage block -- ULEVREL

This service detaches a storage block from an ECB and releases it.

param_1
The address of the ECB with the block to release.

param_2
The address of a 2-byte level indicator. Use the same conventions as the GETCC monitor-request macro.

See *ALCS Application Programming Reference - Assembler* for a description of the GETCC monitor-request macro.

ULEVREL returns with the following conditions:

R15=0
The block is released.

```
        This ULEVREL releases storage block on level D6 from the ECB
        R09 - has the ECB address
        Label LEVEL is defined as DC AL2(D6) in the program
        SPACE 1
        DXCSERV ULEVREL,PARM=((R09),LEVEL)
        SPACE 1
                                  NO NEED TO CHECK RETURN CODE
```

Figure 81. Example: ULEVREL

Validate an ECB storage level -- ULEVVAL

param_1
The address of the ECB.

param_2
The address of a 2-byte level indicator. Use the same conventions as the GETCC monitor-request macro.

See *ALCS Application Programming Reference - Assembler* for a description of the GETCC monitor-request macro.

ULEVVAL returns with the following conditions:

R15=0
The storage level is valid and in use.

R15=4
The storage level is not in use.

R15=8
The storage level is overwritten.

```
        This ULEVVAL validates storage level D6 of the ECB
        R09 - has the ECB address
        Label LEVEL is defined as DC AL2(D6) in the program

        DXCSERV ULEVVAL,PARM=((R09),LEVEL)
        SPACE 1
        B     *+4(R15)            CHECK THE RETURN CODE
        B     USEIT               0 - LEVEL IN USE AND OK
        B     EMPTY               4 - LEVEL NOT USED
        B     ERROR               8 - LEVEL CORRUPTED
```

Figure 82. Example: ULEVVAL

Validate an ECB monitor storage level -- UMLEVVAL

The ECB contains 8 monitor storage levels which are reserved for ALCS use:

level D0
Automatic storage block.

level D1
Detached block table.

level D2
ZTEST work area.

level D3
High-level language work area.

level D4
Communication work area.

level D5
Communication work area.

level D6
Local save area stack.

level D7
CEP and OCTM work area.

The monitor storage levels can not be accessed by application programs, however you can use this service to test if they are valid.

param_1
The address of the ECB.

param_2
The address of a 2-byte level indicator. Use the same conventions as the GETCC monitor-request macro to specify the monitor storage level indicator, from D0 to D7.

See *ALCS Application Programming Reference - Assembler* for a description of the GETCC monitor-request macro.

UMLEVVAL returns with the following conditions:

R15=0
The monitor storage level is valid and in use.

R15=4
The monitor storage level is not in use.

R15=8
The monitor storage level is overwritten.

```
        This UMLEVVAL validates monitor storage level D6 of the ECB
        R09 - has the ECB address
        Label MLEVEL is defined as DC AL2(D6) in the program

        DXCSERV UMLEVVAL,PARM=((R09),MLEVEL)
        SPACE 1
        B     *+4(R15)            CHECK THE RETURN CODE
        B     VALID               0 - LEVEL IN USE AND OK
        B     VALID               4 - LEVEL NOT USED
        B     ERROR               8 - LEVEL CORRUPTED
```

Figure 83. Example: UMLEVVAL

Trace a message of a particular communications resource -- UMSGT1 and UMSGT2

This service allows an exit to trace a message of a particular communications resource. Use the UMSGT2 callable service in the Monitor installation-wide exit USRTCP2 or USRTCP4, and use the UMSGT1 callable service otherwise.

param_1
The address of the message to be traced.
This address must be always storage below the bar. If you use UMSGT2 and the message is in storage above the bar, you must copy (part of) the message to storage below the bar.

param_2
The address of a 4-byte field containing the size of the message to be traced.

param_3
The address of the communications table entry for this resource.

param_4
Address of a 4-byte field. This field is for use by the user. It must contain any non-zero binary information.

The top bit must be on if this an input message to ALCS. The top bit must be off if this is an output message from ALCS.

The installation-wide ECB controlled exit programs AMG1 and AMG2 are conditionally entered before a traced message is displayed. This allows the user (of AMG1) to modify the line of the message containing the TOD clock, CRN, size, and direction indicator, or (in the case of AMG2) to format a line containing data before the message is displayed or alternatively to request not to use the message.

param_5
The address of a work area for UMSGT1 or UMSGT2 used by the callable service.

UMSGT1
7-fullword work area

UMSGT2
25-fullword area

Note: After calling UMSGT2 registers R01, R02, R03, R07, R10, R12, R14, R15 are unpredictable.

Find program address callable service -- UPROGF

This service allows an exit to obtain the storage address of an application program.

This service can be used to verify that a program is loaded before an ECB is created and activated for it.

param_1
Specify zero.

param_2
The address of a 4-byte field which holds the name of a program. The service provides the storage address of this program.

param_3
This parameter contains:

- The address of a 3-byte field which holds the CRI of a resource. The system tries to find the program name for this resource.
- Zero if a system-wide loaded program address is requested.

UPROGF returns with the following conditions:

R15=0
The program was found.

param_1
The address of the program. If a CRI-specific loaded program was requested (**param_3** non-zero), but no test version of the program is found, the address of a system-wide version is returned (if a system-wide version of the program is found).

R15=4
The requested program is not loaded.

```
        This UPROGF checks that program TEMP is loaded system-wide.

        LA    R07,=C'TEMP'        LOAD ADDRESS OF PROGRAM NAME
        SPACE 1
        DXCSERV UPROGF,PARM=(0,(R07),0)
        SPACE 1
        LTR   R15,R15             IS THE PROGRAM LOADED
        BNZ   LAB10               NO - BYPASS ECB CREATION
        SPACE 1
```

Figure 84. Example: UPROGF

Callable service user routine - URTN1

URTN1 calls installation-wide exit USRRTN1 through ALCS to perform common user functions. This service allows multiple installation-wide monitor exits to use common code.

param_1,,*param_2*,,*param_3*,,*param_4*,,*param_5*,,*param_6*,,*param_7*,,*param_8*

Parameter list for USRRTN1.

URTN1 returns with the following conditions:

R15=*x*
Defined by USRRTN1. Can be any value except 20

R15=20
The installation-wide monitor exit USRRTN1 is not loaded (set by ALCS).

```
This URTN1 calls a common service in installation-wide
monitor exit USRRTN1.
It uses 3 parameters to pass to the exit.
Parameter 1 (FUNCTION) is used to indicate which routine
within USRRTN1 needs to be invoked.
This allows many services within 1 exit to be defined.
FUNCTION is defined as 4 in the exit.
R02 holds the address relevant to the second parameter.
R06 holds the address relevant to the third parameter.

SPACE 1
DXCSERV URTN1,PARM=(FUNCTION,(R02),(R06))
CH    R15,=H'20'          IS THE EXIT LOADED
BE    NOTTHERE            NO - BRANCH
```

Figure 85. Example usage of URTN1

Callable service user routine - URTN2

URTN2 calls installation-wide monitor exit USRRTN2 through ALCS to perform common user functions. This service allows multiple installation-wide monitor exits to use common code.

param_1,,*param_2*,,*param_3*,,*param_4*,,*param_5*,,*param_6*,,*param_7*,,*param_8*

Parameter list for USRRTN2.

URTN2 returns with the following conditions:

R15=*x*
Defined by USRRTN2. Can be any value except 20

R15=20
The installation-wide monitor exit USRRTN2 is not loaded (set by ALCS).

```
This URTN2 calls a common service in installation-wide
monitor exit USRRTN2.
It uses 3 parameters to pass to the exit.
Parameter 1 (FUNCTION) is used to indicate which routine
within USRRTN2 needs to be invoked.
This allows many services within 1 exit to be defined.
FUNCTION is defined as 4 in the exit.
R02 holds the address relevant to the second parameter.
R06 holds the address relevant to the third parameter.

SPACE 1
DXCSERV URTN2,PARM=(FUNCTION,(R02),(R06))
CH    R15,=H'20'          IS THE EXIT LOADED
BE    NOTTHERE            NO - BRANCH
```

Figure 86. Example usage of URTN2

Validate storage belonging to an ECB - USTRECB

This service allows an exit to check if an area of storage belongs to an ECB. The storage is considered to belong to an ECB if it lies within any of the type 1, type 2, or type 3 storage units which are currently assigned to the ECB.

param_1
The address of the ECB.

param_2
The address of the storage area.

param_3
The address of a fullword field containing the length of the storage area.

USTRECB returns with the following conditions:

R15=0
The storage belongs to the ECB.

R15=4
The storage does not belong to the ECB.

R15=8
The ECB address is not valid.

```
         This USTRECB checks if a storage area belongs to an ECB
         R09 - has the ECB address
         Label SADDR is the address of the storage area
         Fullword field SLEN contains the storage area length

         DXCSERV USTRECB,PARM=((R09),SADDR,SLEN)
         SPACE 1
         B     *+4(R15)            CHECK THE RETURN CODE
         B     VALID               0 - STORAGE BELONGS TO ECB
         B     ERROR               4 - STORAGE DOES NOT BELONG
         B     ERROR               8 - ECB ADDRESS NOT VALID
```

Figure 87. Example: USTRECB

Obtain MVS virtual storage (GETMAIN) -- USTRGET

param_1
Specify zero.

param_2
The address of a 4-byte length field. Number of bytes if request for storage to be allocated below the bar, or number of 1MB segments if storage to be allocated above the bar.

param_3
The address of a 1-byte indicator field. Use one (or a combination) of the following bit symbols to indicate which type of storage is required:

IW0PFIX
The storage must be page fixed

IW0PROT
The storage must be protected (tables storage key).

IW0RM24
Request virtual storage below 16MB.

IW0RM64
Request virtual storage above 2GB.

You must set **param_3** to zero if it is not used. This gives non-paged-fixed storage in entry key above 16MB in virtual storage.

USTRGET returns with the following conditions:

R15=0
The storage was obtained.

param_1
The address of the storage area for request of storage below the bar or the address of an 8 byte pointer field that addresses the storage area for request of storage above the bar.

R15<>0
The MVS GETMAIN return code.

```
        This USTRGET obtains 1600 bytes of storage in
        tables key and page fixed
        Label STORSIZE is defined as DC F'0' in the user monitor
        save area
        Label STORIND is defined as DC X'00' in the user monitor
        save area
        Label IW1ADR is defined as DC F'0' in the user extension of the
        IW0IT macro
        Label PARM1 is equated to 0(R01)

        LA    R04,1600            NUMBER OF BYTES NEEDED
        ST    R04,STORSIZE        SAVE FOR THE STORAGE GET
        MVI   STORIND,IW0PROT+IW0PFIX TABLES KEY, PAGE FIXED
        SPACE 1
        DXCSERV USTRGET,PARM=(0,STORSIZE,STORIND)
        SPACE 1
        LTR   R15,R15             IS THE STORAGE OBTAINED
        BNZ   ERROR1              NO - BRANCH ERROR
        SPACE 1
        L     R15,PARM1           LOAD STORAGE ADDRESS
        ST    R15,IW1ADR          SAVE THE ADDRESS
```

Figure 88. Example: USTRGET using IW0PROT and IW0PFIX

Release MVS virtual storage (FREEMAIN) -- USTRREL

param_1
The address of the storage area to release for request of storage below the bar or the address of a 8 byte pointer field that addresses the storage area to release for request of storage above the bar.

param_2
The address of a 4-byte length field. The number of bytes if request for storage to be released below the bar, or number of 1MB segments if storage to be released above the bar.

param_3
The address of a 1-byte indicator field. Use one (or a combination) of the following bit symbols to indicate which type of storage to release:

IW0PFIX
The storage is page fixed.

IW0PROT
The storage is protected (tables storage key).

IW0RM24
Request virtual storage below 16MB.

IW0RM64
Request virtual storage above 2GB.

You must use the **same** bit symbol (IW0PFIX, IW0PROT, IW0RM24, or IW0RM64) for a USTRGET and its corresponding USTRREL.

USTRREL returns with the following conditions:

R15=0
The storage was released.

R15<>0
The MVS FREEMAIN return code.

```
        This USTRREL releases 1600 bytes of storage (previously
        obtained by calling the USTRGET callable service)
        The storage is in tables key and page fixed
        Label STORSIZE is defined as DC F'0' in the user monitor
        save area
        Label STORIND is defined as DC X'00' in the user monitor
        save area
        Label IW1ADR is defined as DC F'0' in the user extension of the
        IW0IT macro

        LA    R04,1600            NUMBER OF BYTES NEEDED
        ST    R04,STORSIZE        SAVE FOR THE STORAGE RELEASE
        MVI   STORIND,IW0PROT+IW0PFIX TABLES KEY, PAGE FIXED
        L     R14,IW1ADR          LOAD THE ADDRESS TO BE RELEASED
        SPACE 1
        DXCSERV USTRREL,PARM=((R14),STORSIZE,STORIND)
        SPACE 1
        LTR   R15,R15             IS THE STORAGE RELEASED
        BNZ   ERROR2              NO - BRANCH ERROR
        SPACE 1
```

Figure 89. Example: USTRREL

Validate a virtual storage area -- USTRVAL

param_1
The address of the storage area to validate.

param_2
The address of a 4-byte length field.

param_3
The address of a 1-byte storage key. Use one of the following labels to specify the storage key:

IW0ENTRY
Validate entry storage.

IW0TABLE
Validate table storage.

IW0GLOBL
Validate global-area storage.

USTRVAL returns with the following conditions:

R15=0
The storage is accessible read/write when the PSW key matches the storage key specified by **param_3**. Note that the storage may not be accessible for a different PSW key.

R15=4
The storage is accessible read only when the PSW key matches the storage key specified by **param_3**. Note that the storage may not be accessible for a different PSW key.

R15=8
The storage is not accessible when the PSW key matches the storage key specified by **param_3**. Note that the storage may be accessible for a different PSW key.

```
        This USTRVAL validates an area of storage to check if it can
        be accessed in ENTRY key
        Label STORSIZE is defined as DC F'0' in the user monitor
        save area

        L     R05,EBW000          LOAD ADDRESS TO BE VALIDATED
        LH    R04,EBW004          LOAD LENGTH TO BE VALIDATED
        ST    R04,STORSIZE        SAVE FOR THE STORAGE VALIDATE
        SPACE 1
        DXCSERV USTRVAL,PARM=((R05),STORSIZE,IW0ENTRY)
        SPACE 1
        B     *+4(R15)            CHECK THE RETURN CODE
        B     STOROK              0 - STORAGE AVAILABLE READ/WRITE
        B     STOROK              4 - STORAGE AVAILABLE READ ONLY
        B     STORNOK             8 - STORAGE NOT AVAILABLE
        SPACE 1
```

Figure 90. Example: USTRVAL

Find a translate table -- UTAB1 through UTAB6

param_1
Specify zero.

UTAB1 through UTAB6 return with the following conditions:

param_1
The address of the requested translate table.

R15=0
The table is the translate table supplied by the system.

R15=4
The table is the translate table supplied by the corresponding installation-wide monitor exit.

UTAB1 through UTAB6 correspond to installation-wide monitor exits USRTAB1 through USRTAB6. Refer to "User translate tables - USRTAB1 through USRTAB6" on page 292 for more information on translate tables.

```
        This UTAB2 will retrieve the address of the current
        CCITT#2 to EBCDIC translate table
        If UTAB2 returns with RC=04 in register R15, then the
        external table USRTAB2 is used. Otherwise RC=0 which
        indicates that USRTAB2 is not loaded.
        Label PARM1 is equated to 0(R01).

        SPACE 1
        DXCSERV UTAB2,PARM=(0)
        L     R14,PARM1           GET THE TRANSLATE TABLE ADDRESS
```

Figure 91. Example usage of UTAB1 through UTAB6

Find a translate table for ASCIC -- UTAB7 through UTAB10

param_1
Specify zero.

UTAB7 through UTAB10 return with the following conditions:

param_1
The address of the requested translate table.

R15=0
The table is the translate table supplied by the system.

R15=4
The table is the translate table supplied by the corresponding installation-wide monitor exit.

UTAB7 through UTAB10 correspond to installation-wide monitor exits USRTAB7 through USRTAB10. For more information on translate tables, refer to "User translate tables for ASCIC - USRTAB7 through USRTAB10" on page 293.

```
         This UTAB7 will retrieve the address of the current
         EBCDIC to ASCII translate table
         If UTAB7 returns with RC=04 in register R15, then the
         external table USRTAB7 is used. Otherwise RC=0 which
         indicates that USRTAB7 is not loaded.
         Label PARM1 is equated to 0(R01).

         SPACE 1
         DXCSERV UTAB7,PARM=(0)
         L     R14,PARM1           GET THE TRANSLATE TABLE ADDRESS
```

Figure 92. Example usage of UTAB7 through UTAB10

Write to a system sequential file -- UWSEQ

param_1
The address of the data to write to the system sequential file.

param_2
The address of a 4-byte length field.

param_3
The address of a 1-byte indicator field. Use one of the following bit symbols to specify the type of system sequential file.

IW0DIA
The ALCS diagnostic file.

IW0DCR
The ALCS data-collection file.

IW0USR
The ALCS user file.

UWSEQ returns with the following conditions:

R15=0
The write operation was successful

R15=4
The system sequential file is not currently available

R15=8
The system sequential file type is not correct.

```
This UWSEQ attempts to write data to the ALCS data
collection sequential file
Label STORSIZE is defined as DC F'0' in the user monitor
save area

L     R05,EBW000          LOAD ADDRESS TO BE WRITTEN
LH    R04,EBW004          LOAD LENGTH TO BE WRITTEN
ST    R04,STORSIZE        SAVE FOR THE STORAGE VALIDATE
MVI   STORIND,IW0DCR      SET DATA COLLECTION FILE
SPACE 1
DXCSERV UWSEQ,PARM=((R05),STORSIZE,STORIND)
SPACE 1
B     *+4(R15)            CHECK THE RETURN CODE
B     OK                  0 - DATA WRITTEN OUT OK
B     NOK                 4 - SEQUENTIAL FILE NOT AVAILABLE
B     ERROR               8 - ERROR TAKE DUMP
SPACE 1
```

Figure 93. Example: UWSEQ to the data-collection sequential file

Implementing installation-wide ECB-controlled exits

This section describes the ALCS installation-wide ECB-controlled exits. These exits allow you to customize ALCS processing by writing and loading ECB-controlled programs in the same way as any other ALCS application programs. See *ALCS Application Programming Guide* for more information about writing ECB-controlled programs.

Examples of many of these programs are provided with the ALCS product in the installation-wide exit library.

Note: You are not required to write all (or any) of the programs described in this section. In the descriptions of each exit, the words "ALCS conditionally enters" indicate that ALCS enters a program only if that program is loaded.

Communication user data display exit program - ACD1

The ALCS ZDCOM command processor conditionally enters this program before displaying the communication user data for a resource, in response to ZDCOM with the USERDATA parameter. This allows the user to do any formatting of the user data before it is displayed, or to suppress the display.

ALCS enters ACD1 with the following conditions:

EBX000-EBX084
ZDCOM work area

Level D0
Reserved

Level D2
Reserved

Level D3
Reserved

Level D4
Reserved

Level D5
Work area containing the communication user data.

Register
Contents

R05
The address of communication system data for the resource

R06
The address to communication user data for the resource

R07
Length of communication user data for the resource.

ACD1 is conditionally entered by ENTRC and control must eventually return by BACKC.

Program ACD1, and any program that ACD1 calls, **must not**:

- Modify EBX000 through EBX084.
- Use ECB storage levels 0 or 2 through 4.
- Use ECB storage level 5, unless it is to reformat the communication user data.
- Modify registers R01, R03, R04, or R05.

The ALCS ZDCOM command processor tests the following return conditions from ACD1:

Register
Contents

R15=0
Continue ZDCOM processing. ALCS displays the communication user data in hexadecimal and character representation.

R06
Must contain the address of the data to be displayed.

R07
Must contain the length of the data to be displayed.

R15=4
Terminate ZDCOM processing. ALCS sends a response to the originating terminal without displaying any communication user data.

R15=8
Continue ZDCOM processing. ALCS displays the communication user data in a character string that is appended on a new line directly after the ZDCOM response message header.

R06
Must contain the address of the data to be displayed.

R07
Must contain the length of the data to be displayed.

Communication exit program - ACE1

This program allows an application to take action on errors and other events on the communication network. If this program does not eventually return to the caller, notification of some errors or status changes can be lost.

You must decide whether you need ACE1:

- If you do, check the version of ACE1 shipped with IPARS - ALCS V2 and modify it if necessary.
- If you do not need it, do not load it.

ALCS conditionally enters ACE1 with the following conditions:

EBW000
VTAM return code.

EBW001
VTAM feedback 2 code.

EBW002
Communication code.

W
WTTY

T
Terminal, ALCI LU

PVC
X.25 PVC.

A
APPC/MVS connection

I
TCP/IP connection

Code
Reason

00
Negative response (W).

01
Negative response (T).

02
Printer status (T).

03
Error completion of receive (W+T).

04
Error completion of expedited receive (W+T).

05
Error response (W+T).

06
(not valid).

07
(not valid).

08
Send error (T).

09
(not valid).

0A
Send error (W).

0B
Printer connected or reactivated (T).

0C
WTTY or PVC LU connected (W+PVC).

0D
Printer disconnected or deactivated (T).

0E
WTTY or PVC LU disconnected (W+PVC).

0F
(not valid).

10
Reserved.

11
LU 6.1 link error.

12
LU 6.1 link started.

13
NetView interface active.

14
NetView interface inactive.

15
APPC/MVS resource active.

16
APPC/MVS resource inactive.

17
APPC/MVS error.

18
TCP/IP resource active

19
TCP/IP resource inactive

1A
TCP/IP error

24
BATAP lockout condition

25
MQ resource active

26
MQ resource inactive

27
WAS resource active

28
WAS resource inactive

EBW008
VTAM sense byte.

EBW009
VTAM sense modifier byte.

EBW010-EBW011
User sense bytes.

EBROUT
Resource CRI.

EBW000-EBW016
Do not overwrite.

No storage or data levels are in use. The registers must be saved on entry and restored before returning to the caller.

ACE1 is conditionally entered by ENTRC, and control must eventually return by BACKC.

Communication exit program - ACE2

This program allows an application to take action on status changes on the SLC communication network.

ALCS conditionally creates a new entry to ACE2 if:

- A positive acknowledgment is received for a single block Type B message, or an acknowledge message label is received for a multiblock Type B message. The application can then take the next message from the queue.
- An SLC link that was down, comes up. When ALCS resumes data transmission, the application can send messages on the link.

- An SLC link that was up, goes down. When ALCS suspends data transmission, the application can hold all messages for the link.

ALCS conditionally enters ACE2 with the following conditions:

EBW000
Type of created entry to ACE2:

X'00'
ALCS receives a positive acknowledgment for a single block. Type B message, or an acknowledge message label for a multiblock Type B message.

X'40'
ALCS resumes data transmission on an SLC link.

X'80'
ALCS suspends data transmission on an SLC link.

EBW003
The link number of the SLC link to which the condition applies.

Communication exit program - ACE3 through ACE9

These programs allow an application to perform AX.25 Type B or MATIP Type B specific message processing. The registers must be saved on entry and restored before returning to the caller.

```
ACE3  BATAP application exit 1: Start transmission
ACE4  BATAP application exit 2: Stop transmission
ACE5  BATAP application exit 3: Re-queue message
ACE6  BATAP application exit 4: RECEIVE unique processing
ACE7  BATAP application exit 5: Request next message
ACE8  BATAP interface to IPARS message switching application
ACE9  Reserved for future message switching applications.
```

ALCS enters these programs with the following conditions:

EBROUT
Resource CRI.

Level DF
BATAP information list, for ACE5 only.

Level D0
Message in XMSG format, for ACE6 only.

These programs are conditionally entered by ENTRC, and control must eventually return by BACKC.

User command exit program - ACM0

ALCS enters this program for all commands (including user commands). This allows you to specify your own options (including the name of the program which handles the command) while leaving ALCS responsible for processing those options.

For user commands, ALCS passes a dummy set of options with a dummy program name. ALCS checks these on return and if a valid set of options is substituted (by the user code), ALCS processes the command. Otherwise ALCS takes an error path and calls ACM1.

Note: ACM0 provides centralized facilities such as:

- Optional parsing
- Translation
- Recognition of help requests (Z*xxxx* ? or Z*xxxx* HELP).

ALCS enters every user command handler with the command string (in IMSG format) on level D0. ALCS removes excess spaces and optionally translates:

'-' to '='

'/' to '(' or ')'

ALCS optionally parses the parameters into a work area on level D2. Refer to the C0OPR description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area. For example, the command

```
ZUSER PARM1,PARM2=(FRED,BILL),PARM3
```

is parsed into the work area in the following format

Table 30. How user-command parameters are parsed by ALCS

3 Parameters		
Positional	5	PARM1
Keyword	5	PARM2
	11	(FRED,BILL)
Positional	5	PARM3

You should use ACM0 for all new applications (because of the centralized checking), but ACM1 is still supported for existing users.

If you implement your own commands (in ACM0), consider adding help information, see "Help exit programs - AHL1, AHL2, and so on" on page 368.

ALCS can call ACM0 more than once for a particular command. For example, when different aspects of a command require different checks (altering data, then displaying data). You can either:

- Change the options to what you require
- Leave the options unchanged and let ALCS make the appropriate checks.

The ALCS command processor conditionally enters ACM0 before checking the command authorization, system state, and command edit restrictions. Use this program to change these restrictions for any ALCS command, or to set restrictions for any user command.

ALCS conditionally enters ACM0 with the following conditions:

Level D0
The input message. The CM1CM macro describes the format of the message.

EBROUT
The CRI of the originating terminal.

EBW000-EBW003
Second and subsequent characters of the command name (the first character is always 'Z'), translated to upper case, and padded to the right with space (blank) characters.

EBW004-EBW007
The name of the program which handles this command. This is set to a dummy value if the command is not recognized by ALCS.

EBW008
Bits 0 through 3 indicate the command edit restrictions:

Bit 0 ON, Parse the message.
Bit 1 ON, Translate '-' to '='
Bit 2 ON, Translate '/' to '(' or ')'
Bit 3 ON, Inhibit space (blank) character compression. If off, ALCS replaces any two or more consecutive space characters in the first line of the command with a single space.

For example, bit 1 ON and bit 2 ON specifies the following translation:

```
ZUSER PARM-/A1,A2/
```

is translated to

```
ZUSER PARM=(A1,A2)
```

EBW009
Bits 0 through 3 indicate the ALCS system states in which the command is allowed:

Bit 0 ON, IDLE
Bit 1 ON, CRAS
Bit 2 ON, MESW
Bit 3 ON, NORM.

Bit 0 through bit 3 can be mixed (combined).

Bits 4 through 6 indicate which terminals can issue the command:

Bit 4 ON, Any CRAS
Bit 5 ON, Prime CRAS
Bit 6 ON, Prime CRAS or AT1-AT16.

Bit 4 through bit 6 can be mixed, but the **most restrictive** value takes priority.

EBW010
Minimum message length (zero for no restriction).

EBW011
Maximum message length (zero for no restriction).

You can examine the input command and then take one of the following options:

- Return to the calling program using BACKC without modifying the work area. In this case ACM0 will have no effect.
- Modify EBW004-EBW011 (do not alter EBW000-EBW003) and then return to the calling program using BACKC. In this case the calling program uses the modified restrictions.
- Do not return to the calling program. In this case you must process the ALCS command completely in ACM0.

All application registers can be used by ACM0. The exit must not update work areas EBX000 through EBX103.

User command exit program - ACM1

When the ALCS command processor does not recognize a command, it conditionally enters ACM1.

Note: Do not use ACM1 for user commands in new installations. ACM0 is the preferred method to implement user commands.

ALCS enters ACM1 with the following conditions:

Level D0
The input message. The CM1CM macro describes the format of the message.

You can examine the input command and take one of the following actions:

- If the command is a valid user command, process it completely in ACM1
- If the command is not a valid user command, return to the ALCS command processor using BACKC.

ALCS can also pass application messages to ACM1 where the first character of the message is Z. This is controlled by the ALCS generation COMDEF macro, where the parameter FMSG=NO (the default) is coded.

If your application is designed to accept messages beginning with Z, consider coding FMSG=YES on the COMDEF macro.

ZACOM parameter error exit program - ACME

The ALCS ZACOM command processor conditionally enters this program when the operator specifies an unrecognized keyword or positional parameter on ZACOM. This allows you to process your own ZACOM command parameters.

Before calling ACME, ALCS parses all the parameters into a work area on level D2. Refer to the C0OPR description in "Intended interfaces for installation-wide monitor exits" on page 308 for details of the layout of this area. For example, the command

```
ZACOM PARM1,PARM2=(FRED,BILL),PARM3
```

is parsed into the work area in the following format

Table 31. How user-command parameters are parsed by ALCS

3 Parameters		
Positional	5	PARM1
Keyword	5	PARM2
	11	(FRED,BILL)
Positional	5	PARM3

ALCS conditionally enters ACME with the following conditions:

Level D0
The input message. The CM1CM macro describes the format of the message.

Level D2
The parsed input message. The C0OPR macro describes the format of the parsed message.

EBROUT
The CRI of the originating terminal.

You can examine the input command and then take one of the following options:

- Return to the ALCS command processor using BACKC with R15=0. In this case ALCS will reject the ZACOM command with one of the messages:

  ```
  DXC8013E  Invalid positional parameter
  DXC8012E  Invalid keyword parameter
  ```

- Return to the ALCS command processor using BACKC with R15 not equal to 0. In this case ALCS will reject the ZACOM command with the message:

  ```
  DXC8070E  Command rejected by installation-wide exit ACME
  ```

- Do not return to the ALCS command processor. In this case you must process the ALCS command completely in ACME, and send an appropriate response to the originating terminal.

If returning to the calling program then registers R05 and R06 must not be altered.

Global area load exit program - AGL1

The application global area load routine conditionally enters AGL1 on completion of the global area load.

Level D3
This must not be in use when program AGL1 returns.

AGL1 is entered by ENTRC and control must eventually return using BACKC.

For an example, see the IPARS - ALCS V2 version of this program.

Global area tag exit programs - AGT0, AGT1, and so on

The ALCS ZAC0R/ZDC0R command conditionally reads these application programs to resolve application global area tags. Each program can either be a separate program or a transfer vector in another program. Program AGT0 contains the list of tags for records in global directory 0, AGT1 contains the list for global directory 1, and so on.

Each item in the tag list is either:

- A 4-byte character constant containing the tag name. The tag name can be 1 through 4 characters. It must be left justified and padded with space (X'40') characters if required.
- A 10-byte field containing:

  Bytes
  Contain

  0-1
  The length (binary) of the tag name, excluding trailing space (X'40') characters.

  2-9
  The tag name, 1 through 8 characters. It must be left justified and padded with space (X'40') characters if required.

For global directories with 4-byte slots, the first item in the tag list is the tag for the first record in the global directory, the second item is the tag for the second record, and so on.

For global directories with 8-byte slots, the first item in the tag list is the tag for the first record in the global directory, the second item is a "dummy" tag containing all space (X'40') characters, the third item is the tag for the second record in the global directory, the fourth is a "dummy" tag containing all space (X'40') characters, and so on.

The last tag item in all cases must be followed by:

```
        DC    CL4'<<>>'
```

Avoid defining tag names that contain only the characters A through F and 0 through 9. The ZAC0R and ZDC0R commands cannot distinguish such names from storage addresses.

You must not mix 4-byte tag fields and 10-byte tag fields in the same program.

If you use 4-byte tag names the appropriate GL*n*BA DSECT macro can provide a parameter for generation of the list. The IPARS GL0BA is an example of how to do this for records in a directory with 8-byte slots.

Figure 94 on page 367 shows a sample global area tag program using 10-byte tag fields.

```
         BEGIN NAME=AGT5,VERSION=00,AMODE=31
         SPACE 1
*=====================================================================*
*        APPLICATION GLOBAL AREA TAG LIST FOR DIRECTORY 5             *
*=====================================================================*
         SPACE 1
         DC    AL2(5),CL8'NAME1'
         DC    AL2(7),CL8'ANOTHER'
⋮
         DC    CL4'<<>>'           MARKER FOR END OF TAG LIST
         SPACE 1
         LTORG ,
         SPACE 1
         FINIS ,
         SPACE 1
         END ,
```

Figure 94. Example of a global tag program using 10-byte tag fields

Figure 95 on page 368 shows a sample global area tag program using the GL5BA DSECT macro to generate a list of 4-byte tag fields.

```
         BEGIN NAME=AGT5,VERSION=00,AMODE=31
         SPACE 1
*=================================================================*
*        APPLICATION GLOBAL AREA TAG LIST FOR DIRECTORY 5         *
*=================================================================*
         SPACE 1
         PRINT GEN
         GL5BA ACTION=TABLE
         DC    CL4'<<>>'           MARKER FOR END OF TAG LIST
         SPACE 1
         PRINT NOGEN
         SPACE 1
         LTORG ,
         SPACE 1
         FINIS ,
         SPACE 1
         END   ,
```

Figure 95. Example of a global tag program using 4-byte tag fields

Help exit programs - AHL1, AHL2, and so on

These programs define user help text. The ALCS ZHELP command processor program conditionally reads each help exit program for the required help text. Provide the user help text in these programs by coding HELPC macros. There are four formats for the HELPC macro and each is described below. ALCS identifies the help text which is to be displayed from the primary help topic that is coded with the HELPC macro. For example if the primary help topic is GAME, the help text coded in the HELPC macro (which contains a primary help topic of GAME) would be displayed when the command ZHELP GAME was entered.

ALCS searches AHL1 first, then AHL2, and so on, with AHL0 last. It displays the help text from the first HELPC macro that has been coded with the given primary help topic. If ALCS cannot find a HELPC macro with the given primary help topic in the help exit programs, then it searches the ALCS help items. To modify an ALCS help item, code HELPC with the same primary help topic in a help exit program.

ALCS does not raise any error condition when the ALCS help facility cannot access one or more of the user help programs. Each user help program can be either a separate program or a transfer vector in another program.

Product-sensitive Programming Interface

HELPC macro - providing help text

ALCS supports four formats of the HELPC macro for use in user help programs. Two of these formats specify the help text for a particular help topic (or topic and subtopic), one format specifies a help topic (or topic and subtopic) synonym, and one format indicates the end of the help information in a particular user help program.

You can use the following HELPC format to specify the help text for a particular help topic:

```
topic HELPC 'text_line',...
```

Where:

topic
Primary help topic, 1 to 4 characters.

'text_line'
One line of help text, characters enclosed in quotes.

You can code help text in mixed case. This parameter must obey the normal rules for quoted strings in assembler macroinstructions. In particular, you must double-up quote (') and ampersand (&) characters.

Code each line of your help text as a separate quoted string. Separate each quoted string from the next with a comma. If you code two consecutive commas, ALCS includes a blank line in your help text at that point.

The first line of help text should be 1-200 characters long. ALCS uses it as the heading line for your help information.

ALCS supports this HELPC format primarily for compatibility with older ALCS versions and releases. If you are developing new help information, we recommend that you use the following format instead.

Using the following format, you do not code the help text itself as conventional macroinstruction operands. Instead, you code the help text in records that follow the HELPC macroinstruction.

The format is:

```
HELPC TOPIC={primary|(primary,secondary)}
            [,HEADING='head_text']
            [,TRUNC={71|length}]
            [,CONT={'+'|'c'}]
            [,DELIM={'*END'|'string'}]
```

Where:

TOPIC={*primary*|(*primary*,*secondary*)}
Help topic or topics, where:

primary
Primary help topic, 1 to 8 characters.

secondary
Secondary help topic, 1 to 8 characters.

These help topics must be in upper case only, and they cannot include spaces or special characters other than:

Star or asterisk (*)
Commercial at (@)
Hyphen or minus (-)
Hash or pound (#)

You cannot use 'INDEX' as either the primary or the secondary help topic.

Note: If you specify an ALCS-provided help topic then your help information will **replace** the help information provided with ALCS.

HEADING='*head_text*'
Heading line for your help information, 1-200 characters enclosed in quotes.

You can code heading text in mixed case. This parameter must obey the normal rules for quoted strings in assembler macroinstructions. In particular, you must double-up quote (') and ampersand (&) characters.

The remaining parameters control the way that the HELPC macro interprets the contents of records that follow it in the program source.

TRUNC={71|*length*}
The right-hand source margin for the help text in the following records, a decimal number in the range 10 through 71.

By default (TRUNC=71), HELPC ignores any characters in columns 72 through 80 (which includes the columns that normally contain sequence numbers).

By specifying a smaller number, you can use columns to the right of your help text for update markers without having these update markers appear in the help display.

CONT={'+'|'*c*'}
Text continuation indicator, a single non-blank character enclosed in quotes. HELPC normally treats the last non-blank character of help text on a record as the end of a display line - the text from the next record appears on the next display line. But if the last non-blank character is the text continuation indicator, HELPC replaces the continuation indicator with a blank (space) character and treats the text on the next record as a continuation of the same display line.

Note that HELPC does **not** treat *c* as a continuation indicator unless it is the **last** non-blank character.

DELIM={'*END'|'*string*'}
Delimiter for indicating the end of the help text records for this HELPC macroinstruction, any characters enclosed in quotes. You must include a record containing this string (starting in column 1) following the record that contains the last line of your help text. HELPC does not include this delimiter record as part of your help text.

If you do not include this delimiter record, then HELPC will include the following records (which contain normal assembler instructions and macroinstructions) as part of your help text.

You can use the following HELPC format to define help topic synonyms:

```
HELPC TOPIC={primary|(primary,secondary)}
            ,USE={primary|(primary,secondary)}
```

Where:

TOPIC={*primary*|(*primary*,*secondary*)}
Synonym help topic or topics, where:

primary
Primary help topic, 1 to 8 characters.

secondary
Secondary help topic, 1 to 8 characters.

The rules for these synonym help topics are the same as described above.

USE={*primary*|(*primary*,*secondary*)}
Help topic or topics to which the synonym refers, where:

primary
Primary help topic, 1 to 8 characters.

secondary
Secondary help topic, 1 to 8 characters.

The topic or topics that you specify here must correspond exactly to another HELPC that provides help text. They must **not** refer to another synonym definition. Alternatively, you can specify 'INDEX' (for either *primary* or *secondary*) so that the synonym topic or topics display an index.

You use the following HELPC format to indicate the end of the help information in a particular user help program:

```
HELPC LAST=YES
```

HELPC macro - example of a user help program

Figure 96 on page 371 shows a sample user help program.

The first two HELPC macroinstructions specify the help text that ALCS displays as responses to the commands ZHELP DOG and ZHELP CAT.

The next two HELPC macroinstructions define synonyms. They allow the end user to enter ZHELP ANIMALS DOG as a synonym for ZHELP DOG and ZHELP ANIMALS CAT as a synonym for ZHELP CAT.

The next two HELPC macroinstructions specify the help text that ALCS displays as responses to the commands ZHELP ANIMALS MOUSE and ZHELP ANIMALS MOSQUITO. Notice that the help information for mosquitos uses the default continuation indicator (+) to join two source records into a single display line.

The next two HELPC macroinstructions define synonyms that allow the end user to access the same help information with the commands ZHELP MOUSE and ZHELP MOSQUITO.

The last HELPC macroinstruction indicates the end of the help information in this user help program.

```
         BEGIN NAME=AHL1,VERSION=00,AMODE=31,TYPE='USER HELP'
         SPACE 1
*=====================================================================*
*        USER APPLICATION HELP MESSAGE TEXT                           *
*=====================================================================*
         SPACE 1
DOG      HELPC 'Information about dogs',                               -
               'A dog is an animal with four legs and a tail',         -
               'When a dog wags its tail it is happy'
         SPACE 1
CAT      HELPC 'Information about cats',                               -
               'A cat is an animal with four legs and a tail',         -
               'When a cat wags its tail it is not happy'
         SPACE 1
         HELPC TOPIC=(ANIMALS,DOG),USE=DOG
         HELPC TOPIC=(ANIMALS,CAT),USE=CAT
         SPACE 1
         HELPC TOPIC=(ANIMALS,MOUSE),                                  -
               HEADING='Information about mice'
A mouse is an animal with four legs and a tail
When a mouse sees a cat it is not happy
*END
         SPACE 1
         HELPC TOPIC=(ANIMALS,MOSQUITO),                               -
               HEADING='Information about mosquitos'
A mosquito is an animal with six legs and wings
It is difficult to be completely sure if a mosquito+
is happy or not happy
*END
         SPACE 1
         HELPC TOPIC=(MOUSE),USE=(ANIMALS,MOUSE)
         HELPC TOPIC=(MOSQUITO),USE=(ANIMALS,MOSQUITO)
         SPACE 1
         HELPC LAST=YES
         EJECT ,
         LTORG ,
         SPACE 1
         FINIS ,
         SPACE 1
         END   ,
```

Figure 96. Example of a user help program

HELPC macro - setting and clearing help context

ALCS supports two formats of the HELPC macro for use in your application programs. These HELPC formats modify the help context. The help context is the default help topic (or topic and subtopic) for the ZHELP command for the originating end user.

You can use the following HELPC format to set the help context for the ZHELP command for the originating end user:

```
[label] HELPC SET_CONTEXT
                        ,TOPIC=(primary[,secondary])
```

Where:

label
Any valid assembler label.

SET_CONTEXT,
Set the default primary and secondary help topic for the ZHELP command for this end user. This default remains in effect until either:

- A subsequent HELPC macro changes the default, or
- The end user enters another input message or command.

TOPIC=(*primary*[,*secondary*]):
The primary and optional secondary help topic. Help topics are strings of up to 8 characters. They must be in upper case, and they cannot include spaces or special characters other than:

Star or asterisk (*)
Commercial at (@)
Hyphen or minus (-)
Hash or pound (#)

Specify primary and optionally secondary as one of:

'topic'
Help topic enclosed in quotes.

field
Name of an 8-byte field that contains the topic left-aligned and padded as required with space (X'40') characters.

(reg)
Register enclosed in parentheses that contains the address of an 8-byte field. The field contains the topic left-aligned and padded as required with space (X'40') characters.

If you omit the secondary help topic, HELPC clears the default secondary help topic.

You can use the following HELPC format to clear the help context:

```
[label] HELPC CLEAR_CONTEXT
```

Where:

label
Any valid assembler label.

CLEAR_CONTEXT
Clears the default primary and secondary help topics for the ZHELP command for this end user.

With no default topics, ZHELP displays the help menu.

End of Product-sensitive Programming Interface

User-programmable PF key exit program - AKY1

The ALCS default PF key settings are:

Program function key	Action key	Meaning assumed by IPARS
PF1	E	End transaction
PF2	I	Ignore transaction
PF3	0A	Arrival unknown
PF4	R	Repeat message
PF5	U	Request unsolicited message
PF6	XI	Cancel itinerary
PF7	*	Display functions
PF8	A*	More availability
PF12	X'FF'	Echo input to top of screen

The values are repeated for PF keys 13 through 24.

This program allows you to override these default setting. The individual end users can also set up their own PF key settings using the ZAKEY command. This is described in the *ALCS Operation and Maintenance*.

ALCS conditionally enters AKY1 with the following conditions:

EBROUT
The CRI of the originating terminal.

CE1URA-CE1UR7
Available to user

Register
Contents

R00
Key number for which the setting is required.

R07
The address of the target area where AKY1 is to store the required setting.

ALCS tests the following return conditions from AKY1:

R14=0
Use the AKY1 setting, unless the end user provides one.

R14=4
Use the ALCS default setting, unless the end user provides one.

R14=8
Use the AKY1 setting, even if the end user provides one.

R14=12
Use the ALCS default setting even if the end user provides one.

If R14=0 or 8, AKY1 must also place the required PF key setting in the target area pointed to by R07, and place the length of the PF key setting in R06.

Registers R03, R04, and R07 must not be changed by AKY1.

A sample AKY1 is provided in the installation-wide exit library. It contains a table in which each PF key and the corresponding return condition is defined. You can alter the PF key settings by changing values in the table.

If no AKY1 program is loaded, any PF key settings provided by the individual end users take precedence over the ALCS default settings.

LU 6.1 link queue swing exit program - ALK1

A queue of LU 6.1 link messages can be moved from one LU 6.1 link to another using the ALCS ZACOM command (with the TOCRN parameter). This action is called queue swing.

At the start of queue-swing processing, ALCS conditionally enters ALK1 with the following conditions:

Level D2
Unused or held resource control record for CRI in register 14.

Level D3
Unused or held resource control record for CRI in register 15.

Register
Contents

R14
CRI of resource from which messages are moved.

R15
CRI of resource to which messages are moved.

R07
The address of an automatic storage block. The format is described in program CLQQ.

Program ALK1 and any program that ALK1 calls must not modify R07.

Program ALK1 must issue BACKC to return to the calling program. Otherwise the LU 6.1 link queues will not be processed.

WTTY exit program - ALS1

ALCS conditionally enters ALS1 when a WTTY line starts.

ALCS enters conditionally ALS1 with the following conditions:

EBW000
Line number.

EBW001
Line type code.

EBW002
Type of line start:

X'00'
Normal line start.

X'01'
Line start during state change.

EBW003
Type of enter to ALS1:

X'00'
Call from ALCS.

Program ALS1 must issue a BACKC to return to the calling program.

ALS1 must not alter EBX000 through EBX003.

Online message trace exit program - AMG1

The ALCS ZTRAC message display command processor conditionally enters user exit program AMG1 before displaying a traced message. This allows the user to format a line containing the TOD clock, CRN, size, and direction indicator.

ALCS enters AMG1 under the following conditions:

Register
Contents

R02
The 4-byte indicator set by either callable services UMSGT1, UMSGT2, or by ALCS itself (values X'00000000' and X'80000000').

R06
The address of the display line which contains the TOD clock, CRN, size, and direction indicator.

R15
The size of the display line pointed to by register R06.

AMG1 may change the display line before returning to the calling program. In that case register R06 contains the address of the new line (terminated by a #CAR) and R15 contains the size of that new line.

AMG1 must not alter any ECB work areas except EBW080-EBW099. All registers (except R06 and R15) must be the same as at entry.

If you require a data level, ATTAC and DETAC macros (with parameter type=STACK) must be used to save and restore the contents of any data levels being used.

AMG1 must return using a BACKC.

Online message trace exit program - AMG2

The ALCS ZTRAC message display command processor conditionally enters user exit program AMG2 before displaying a traced message. This allows the user to format a line containing data before the message is displayed or alternatively to request not to use the message.

ALCS enters AMG2 under the following conditions:

Data Level
Contents

D6
Traced message data

Register
Contents

R02
The 4-byte indicator set by either callable services UMSGT1, UMSGT2, or by ALCS itself.

R06
Actual size of the traced message..

R14
Size of the traced message data which is a maximum of 3900 bytes.

R15
Address of user information which is a maximum of 8 characters padded with blanks.

This user information is passed by the ZTRAC command processor. See the Installation and Operations for more details on the `ZTRAC msg display` command.

AMG2 must not alter any ECB work areas except EBW001 (bit 4) and EBW080-EBW099.

All registers (except R14 and R15) must be the same as at entry.

If you require a data level, ATTAC and DETAC macros (with parameter type=STACK) must be used to save and restore the contents of any data levels being used.

AMG2 must return using a BACKC. You can set before you return bit 4 (X'08') in EBW001 to indicate that the traced data must be interpreted as ASCII characters.

ALCS tests the following return conditions from AMG2:

R15=0
The ZTRAC command processor displays the message upon return.

R15=4
The ZTRAC command processor discards the message upon return.

MQSeries connect exit program - AMQR

ALCS can be connected to an MQSeries queue manager by the operator command ZCMQI with the CONNECT parameter. ALCS conditionally creates a new entry to AMQR whenever a connection to MQSeries is established using this command.

There are no entry conditions to AMQR.

When program AMQR has performed the required actions, it must issue EXITC or BACKC to exit.

Incoming SITA NCB processing exit program - ANCB

ALCS conditionally enters ANCB when a SITA NCB is received on an Type 3 SLC link, or on an X.25 Type 4 link which uses the input message editor CNCB.

ALCS conditionally enters ANCB with the following conditions:

Level D0
Network Control Block (NCB). The CM1CM macro (input format) describes this.

EBW000-EBW001
High Level Designator (HLD) of regional control center (RCS) (if P1124 link).

EBW002-EBW003
Logical Channel Number (LCN) of agent set control unit (ASCU) (if AX.25 link).

EBW004
NCB type indicator, where:

X'00'
Report NCB

X'04'
Non-report NCB.

Register
Contents

R01
Points to the NCB on level D0.

R04
Points to one component report (OCR) within the NCB.

ANCB is invoked by ALCS for each component report. If the NCB contains multiple component reports, ANCB is invoked for each component report.

All registers can be used by ANCB. ECB work areas EBW000 through EBW103 and EBX000 through EBX103 are reserved.

ALCS tests the following return conditions from ANCB:

R15=0
Process report NCB.

R15=4
Discard report NCB.

You can modify fields EBX000 through EBX007 to replace the SITA HLD addresses. For more details see the SITA document *Status Control Service Step 2: Automatic Protected Report to ACS*, 085-2/LP-SD-001.

OCTM policing exit program - AOCM

ALCS conditionally enters this exit program when the Online Communication Table Maintenance (OCTM) policing function is scanning the OCTM database. The OCTM policing function scans the base and update

records on the OCTM database to check the status of each communication resource which is managed by OCTM. This exit is activated when ALCS detects a period of inactivity (since the last COMTC macro was issued) for a communication resource or communications group. The exit allows the application to determine if a warning message should be output on RO CRAS and to initiate the next COMTC action that is outstanding for the communication resource or group.

ALCS conditionally enters AOCM with the following conditions:

EBW000-EBW007
Communications Resource Name (CRN) or OCTM communications group name.

EBW008-EBW009
The elapsed time in hours since the last COMTC macro was issued for this communication resource or group. This is a halfword binary value.

EBW010
Communications group indicator. If this field contains the character G, the ECB field EBW000 will contain the name of a communications group, otherwise EBW000 will contain a CRN.

EBW011-EBW020
The COMTC action that is outstanding for this communication resource or group (this field contains the name of that action). It can be one of the following four COMTC actions: LOAD, CONFIRM, COMMIT or UNALLOCATE.

Program AOCM, and any program that AOCM calls, must not issue any SERRC macros.

AOCM is conditionally entered by ENTRC and control must eventually be returned to ALCS by a BACKC.

ALCS tests the following return codes from AOCM:

R15=0
A warning message about the outstanding COMTC action is to be sent to RO CRAS.

R15=4
A warning message about the outstanding COMTC action is not to be sent to RO CRAS.

In program AOCM, and any program that AOCM calls, a COMTC macro can be issued to perform the outstanding action. If the COMTC action remains outstanding for a further 8 hours, this exit program will be activated again for the communication resource or group.

Stripe exit program - APA1

Fixed file and short-term pool records can be redistributed across the ALCS data base using the ALCS restripe function (ZDASD STRIPE). During a restripe additional long-term pool records are required to accommodate the redistributed records. ALCS must ensure, prior to commencement of the restripe, that there is enough long-term pool of the correct size to complete the restripe process and to continue normal operation (both during and after the restripe). After the restripe is complete, those data base records which had previously contained fixed file and/or short-term pool records are not returned to the available long-term pool until after a Recoup has been run.

ALCS conditionally enters APA1 before commencing the restripe to allow the application to change the minimum number of long-term pool addresses that must remain after the restripe function has completed. It also allows the application to change the predefined minimum depletion time for the long-term pool (set at 24 hours).

If no APA1 program is loaded, ALCS will allow the restripe to start if there is enough long-term pool for:

- The number of records that will be redistributed by restripe
- Plus enough available long-term pool records to ensure that the pool is not depleted within 24 hours from the commencement of the restripe (or 500 long-term pool records, if this is a higher value).

ALCS conditionally enters APA1 with the following conditions:

EBW000-EBW003
Minimum number of long-term pool records that must remain after those required for the restripe function have been deducted from the currently available pool.

EBW004-EBW007
Minimum depletion time (in hours) for the long-term pool, based on the current pool depletion rate (for the count of addresses in the pool after deduction of those required for the restripe function).

Register
Contents

R02
Address of an information area for the file type being restriped. The format of this information area is described in the *ALCS Application Programming Reference - Assembler* (under the description of the GETRON parameter on the RONIC macro). Use the RONBTY(R02) equate to access the size code (for example, L1, L2, ...).

Program APA1 can change either of the two minimum values defined in EBW000 and EBW004, and ALCS will use those values instead of the default ones.

Program APA1, and any program that APA1 calls, must not use EBX000 through EBX039 (in the ECB work area) and must not issue any SERRC macros.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

Program APA1 must issue BACKC to return to the calling program to commence the restripe.

Performance monitor history exit program - APF1

ALCS conditionally enters APF1 when updating the performance monitor history records. APF1 must return with the number of years (beyond the current year) for which history records will be kept online. The range is zero to five years. The default is five years.

There are no special entry conditions for APF1.

Program APF1, and any program that APF1 calls, **must not**:

- Modify EBW000 through EBW103 or EBX000 through EBX103.
- Use ECB storage levels.
- Modify registers R00 through R07.

APF1 must return using BACKC.

On return from APF1, register R15 must contain the number of years (0-5).

Figure 97 on page 378 shows an example where history records will be kept online for 3 years.

```
          SPACE 1
APF1      EQU   *
          LA    R15,3           SET NUMBER OF YEARS (THREE)
          BACKC ,               RETURN
```

Figure 97. Example of retaining history records online

Pool dispensing array for restore program exit - APDR

ALCS conditionally enters APDR when creating the PDAR structure.

APDR is invoked once for each long term pool interval and returns either the numbers of records to be reserved or the new factors that rule the calculation for that interval.

Program APDR, and any program that APDR calls, must not:

- Modify ECB storage levels.
- Modify registers R00 through R07.

ALCS enters APDR with the EBW000-EBW103 ECB fullword field equates defined as:

CPDRVMAX
Default for the maximum number of records reserved

CPDRVMIN
Default for the minimum number of records reserved

CPDRVPCT
Default for ratio from the total number of records

CPDRITVL
Long-term pool interval

LL L1
F'01'

LT L2
F'03'

LT L3
F'05'

LT L4
F'07'

LT L5
F'09'

LT L6
F'11'

LT L7
F'13

LT L8
F'15'

ALCS returns from APDR with the EBW000-EBW103 ECB fullword field equates updated:

CPDRVMAX
New value for the maximum number of records reserved

CPDRVMIN
New value for the minimum number of records reserved

CPDRVPCT
New value for the ratio from the total number of records

CPDRNREC
Number of records to be reserved

and these registers updated:

R15=0
Use the new calculation factors set on CDPRVMAX, CDPRVMIN, and CDPRVPCT.

R15=4
Bypass calculation. Use the number of records from CPDRNREC.

Performance monitor history exit program - APF1

ALCS conditionally enters APF1 when updating the performance monitor history records. APF1 must return with the number of years (beyond the current year) for which history records will be kept online. The range is zero to five years. The default is five years.

There are no special entry conditions for APF1.

Program APF1, and any program that APF1 calls, **must not**:

- Modify EBW000 through EBW103 or EBX000 through EBX103.
- Use ECB storage levels.

- Modify registers R00 through R07.

APF1 must return using BACKC.

On return from APF1, register R15 must contain the number of years (0-5).

Figure 98 on page 380 shows an example where history records will be kept online for 3 years.

```
            SPACE 1
APF1        EQU   *
            LA    R15,3           SET NUMBER OF YEARS (THREE)
            BACKC ,               RETURN
```

Figure 98. Example of retaining history records online

Performance monitor average exit program - APF2

ALCS conditionally enters APF2 during performance monitor average processing. APF2 must return with the addresses of two tables, one containing the hours to be used when creating the day average file record, and the other containing the days to be used when creating the month average file record.

The default is to use all hours of a day and all days of a month.

There are no special entry conditions for APF2.

Program APF2, and any program that APF2 calls, **must not**:

- Modify EBW000 through EBW103 or EBX000 through EBX103.
- Use ECB storage levels.
- Modify registers R00 through R07.

APF2 must return using BACKC.

On return from APF2, register R14 must contain the address of the hour table used by the day average process and register R15 must contain the address of the day table used by the month average process.

Table 32. Format of the hour table

Offset	Length	Contents
0	2	Number of entries in the table, a halfword value from 1 to 24.
2	2	First hour value. Two numeric characters from 00 to 23.
4	46	Up to 23 further hour values.

Table 33. Format of the day table

Offset	Length	Contents
0	2	Number of entries in the table, a halfword value from 1 to 31.
2	2	First day value. Two numeric characters from 01 to 31.
4	60	Up to 30 further day values.

Figure 99 on page 381 shows an example where average processing will use all hours of a day and all days of a month.

```
         SPACE 1
APF2     EQU   *
         LA    R14,APF2HNUM    ADDRESS HOUR TABLE
         LA    R15,APF2DNUM    ADDRESS DAY TABLE
         SPACE 1
         BACKC ,               RETURN
         SPACE 1
******** HOUR TABLE
         SPACE 1
APF2HNUM DC   Y(APF2HLEN/2)
APF2HTBL DC   C'000102030405060708091011121314151617181 9'
         DC   C'20212223'
APF2HLEN EQU  *-APF2HTBL
         SPACE 1
******** DAY TABLE
         SPACE 1
APF2DNUM DC   Y(APF2DLEN/2)
APF2DTBL DC   C'0102030405060708091011121314151617181920'
         DC   C'21222324252627282930 31'
APF2DLEN EQU  *-APF2DTBL
         SPACE 1
```

Figure 99. Example of the hour and day tables

Application start-up exit program - APL1

ALCS application start/stop conditionally enters this program to perform any custom initialization required during start-up of an ALCS application. This custom initialization can include starting specialized communication links dedicated for use by the application.

ALCS conditionally enters APL1 with the following conditions:

EBW000-EBW007
CRN of application being started.

EBROUT
CRI of originating terminal, or zeros if none.

Program APL1 must issue BACKC to return to the calling program to complete the application start-up.

ALCS tests the following return conditions from APL1:

R15=0
Custom application initialization complete OK

R15=4
Application CRN not known to APL1

R15=8
Custom application initialization failed.

If R15=0, APL1 must set the following fields:

EBW008-EBW011
Four-character name of the receiver program for the application, or binary zeros if none.[11]

EBW012-EBW016
Four-character name of the notify program for the application, or binary zeros if none.[12]

If APL1, or any program that it calls, modifies any other ECB fields, levels, or registers, then APL1 must restore them before it returns.

[11] The current release of ALCS does not use the receiver program name.
[12] The current release of ALCS does not use the notify program name.

Application shut-down exit program - APL2

ALCS application start/stop conditionally enters this program to perform any custom clean-up required during shut-down of an ALCS application. This custom clean-up can include stopping specialized communication links dedicated for use by the application.

ALCS conditionally enters APL2® with the following conditions:

EBW000-EBW007
CRN of application being started.

EBROUT
CRI of originating terminal, or zeros if none.

Program APL2 must issue BACKC to return to the calling program to complete application shut-down.

ALCS tests the following return conditions from APL2:

R15=0
Custom application clean-up complete OK

R15=4
Application CRN not known to APL2

R15=8
Custom application clean-up failed.

If APL2, or any program that it calls, modifies any ECB fields, levels, or registers, then APL2 must restore them before it returns.

Long-term pool activity monitor exit programs - APM1 and APM2

Minimum pool threshold - APM1

The long-term pool activity monitor issues a warning message whenever the count of available pool records drops below the minimum threshold you have set using the ZP00L command.

ALCS conditionally enters APM1 before issuing this warning message, with the following conditions:

EBW000-EBW003
The pool size code (1 = size L1, and so on).

EBW016-EBW019
The count of available records.

EBW020-EBW023
The interval length in minutes.

EBW024-EBW027
The number of minutes into the current interval.

Program APM1, and any program that APM1 calls, must not use EBW000 through EBW063 (in the ECB work area) and must not issue any SERRC macros.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

Program APM1 must issue BACKC to return to the calling program.

ALCS tests the following return conditions from APM1:

R15=0
Do issue warning message DXC2772W or DXC2773W

R15=4
Do not issue warning message DXC2772W or DXC2773W

Pool dispense rate threshold - APM2

The long-term pool activity monitor issues a warning message whenever the dispense rate or depletion time passes any of the thresholds you have set using the ZP00L comand.

ALCS conditionally enters APM2 before issuing this warning message, with the following conditions:

EBW000-EBW003
The pool size code (1 = size L1, and so on).

EBW004-EBW007
The period (in minutes) over which the threshold was exceeded.

EBW008-EBW011
The dispense rate (per second) over this period.

EBW012-EBW015
The time to depletion (hours) at this rate.

EBW016-EBW019
The count of available records.

EBW020-EBW023
The interval length in minutes.

EBW024-EBW027
The number of minutes into the current intrerval.

Program APM2, and any program that APM2 calls, must not use EBW000 through EBW063 (in the ECB work area) and must not issue any SERRC macros.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

Program APM2 must issue BACKC to return to the calling program.

ALCS tests the following return conditions from APM2:

R15=0
Do issue warning message DXC2774W

R15=4
Do not issue warning message DXC2774W

Printer queue swing exit program - APR1

A queue of printer messages can be moved from one printer to another by the operator command ZACOM (with the TOCRN= parameter). This is called printer queue swing.

ALCS conditionally enters APR1 at the start of queue swing processing. This allows printer queues that are under the control of an application to be moved from one printer to another.

ALCS conditionally enters APR1 with the following conditions:

Register
Contents

R14
CRI of the resource that the queues are being moved from.

R15
CRI of the resource that the queues are being moved to.

Program APR1, and any program that APR1 calls, must not modify general register R07.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

Program APR1 must issue BACKC to return to the calling program to swing the printer queue.

Printer shadowing exit programs - APR2 through APR4

Printer shadowing allows up to 16 printers to receive a copy of each message sent to a specific printer. It is activated by the ZACOM command.

Suppressing the header - APR2

ALCS conditionally enters APR2 when a printer message is ready to be shadowed. This allows the application to indicate whether shadow messages must be preceded by a header or not.

ALCS enters APR2 with the following conditions:

EBX000-EBX002
CRI of the printer that is being shadowed.

Level D0
First or only block of message to be shadowed in CM1CM format.

ALCS tests the following return conditions from APR2:

R15=0
Add header to the message.

R15=4
Do not add header to the message.

for all shadow printers that will receive this message.

Program APR2 must issue BACKC to return to the calling program to shadow the printer message.

Program APR2, and any program that APR2 calls, **must not**:

- Modify EBW000 through EBW103
- Modify EBX000 through EBX003
- Issue any SERRC macros.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

Modifying the header - APR3

ALCS conditionally enters APR3 when a shadow message is ready to be sent. This allows the application to perform additional processing such as modifying the text of the message header before the message is sent to a shadow printer.

ALCS enters APR3 with the following conditions:

EBX000-EBX002
CRI of the printer that is being shadowed.

Level D3
First or only block of shadow message in CM1CM format. If you have requested shadow message headers (through APR2), this contains the header.

Program APR3 must issue BACKC to return to the calling program to send the shadow message.

Program APR3, and any program that APR3 calls, **must not**:

- Modify EBW000 through EBW103
- Modify EBX000 through EBX003
- Issue any SERRC macros.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

End of printer shadowing processing - APR4

ALCS conditionally enters application interface program APR4 after sending all the shadow messages. This allows the application to perform any processing required before shadow printer processing is finished.

ALCS enters APR4 with the following conditions:

EBX000-EBX002
CRI of the printer that is being shadowed.

Level D3
First or only block of shadow message in CM1CM format.

Program APR4 must issue BACKC to return to the calling program to send the shadow message.

Program APR4, and any program that APR4 calls, **must not**:

- Modify EBW000 through EBW103
- Modify EBX000 through EBX003
- Issue any SERRC macros.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

Printer exit programs - APR5 and APR6

Modifying the text - APR5

ALCS conditionally enters APR5 immediately prior to transmission of a message to a printer. This allows the application to change the text of the message according to the type of transmission or message content.

ALCS enters APR5 with the following conditions:

EBX000
Indicates the type of transmission:

X'00'
Direct transmission through SLMTC

X'01'
Direct transmission to RO CRAS (on low pool condition)

X'02'
Normal transmission

X'03'
Retry transmission

X'04'
Timeout transmission (test message)

X'05'
Acquisition message to a shared 3270 printer. ALCS sends a message to the printer to indicate that ALCS is using it.

EBX001
Indicates message block being transmitted:

X'80'
First block of message. This bit is on (1) if the block is the first or only block of the message.

X'40'
Last block of message. The bit is on (1) if the block is the last or only block of the message.

EBX002-EBX003
Contains one of:

- The retry count limit for a retry transmission (when EBX000 is X'03'). If the limit is exceeded, a test message is sent to the printer. The test message is repeated at regular intervals.
- The interval between test-message transmissions (when EBX000 is X'04'). The test message is sent when the retry count limit is exceeded.

EBX004-EBX007
The priority number of the queue from which the message is being transmitted.

Level D3
The message to be transmitted in CM1CM format.

Program APR5 and any program that APR5 calls, **must not**:

- Modify EBW000 through EBW103
- Issue any SERRC macros.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

Program APR5 must issue BACKC to return to the calling program to send the message.

Set thresholds for long-term pool dispensing - APR6

ALCS conditionally enters APR6 immediately before dispensing the long-term pool records in which the printer message will be filed.

This allows the application to change the predefined minimum number of long-term pool addresses that must be available before ALCS dispenses an address for the printer message. It also allows the application to change the predefined minimum pool depletion time after which ALCS does not dispense an address for the printer message.

ALCS sets the entry conditions for APR6, and the user can change these thresholds.

ALCS conditionally enters APR6 with the thresholds in the following locations:

Table 34. APR6 - Location of thresholds for dispensing long term pool record

Type	Minimum Number of Records	Minimum Depletion Time
L1LT	EBW000-EBW003	EBW004-EBW007
L2LT	EBW008-EBW011	EBW012-EBW015
L3LT	EBW016-EBW019	EBW020-EBW023
L4LT	EBW024-EBW027	EBW028-EBW031
L5LT	EBW032-EBW035	EBW036-EBW039
L6LT	EBW040-EBW043	EBW044-EBW047
L7LT	EBW048-EBW051	EBW052-EBW055
L8LT	EBW056-EBW059	EBW060-EBW063

Program APR6 and any program that APR6 calls, must use only EBW000 through EBW064 and must not issue any SERRC macro calls.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

Program APR6 can change any of the minimum values defined in EBW000 to EBW063 for any of the long-term pools. When the new minimum values are defined, APR6 must issue BACKC to return to the

calling program to select the long term pool from which a pool address is dispensed for the printer message.

Printer redirection exit programs - APR7 through APR9

Printer redirection allows messages addressed to one printer to be redirected to an alternate printer (called the redirection printer). It is activated by the ZACOM command.

Suppress redirection - APR7

ALCS conditionally enters APR7 when a printer message is ready to be redirected. This allows the application to indicate whether this message is to be redirected to the redirection printer.

ALCS conditionally enters APR7 with the following conditions:

EBX000-EBX002
CRI of the printer whose messages are being redirected.

EBX003
Message transmission indicator:

X'80'
Printer message was transmitted by an application which requires notification of the answerback, (positive acknowledgement).

EBX004-EBX011
CRN of the application which is to receive the answerback.

EBX012-EBX015
The priority number of the queue from which the message is being transmitted.

Program APR7 and any program that APR7 calls can use the ECB work areas but must not issue any SERRC macro calls.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

ALCS tests the following return conditions from APR7:

R15=0
Process the redirection.

R15=4
Do not process the redirection.

Program APR7 must issue BACKC to return to the calling program to begin redirection of the message (according to the return condition in register R15).

Suppressing the header - APR8

ALCS conditionally enters APR8 immediately prior to appending a header to the redirected message. This allows the application to indicate whether the redirected message must be preceded by a header.

ALCS conditionally enters APR8 with the following conditions:

EBX000-EBX002
CRI of the printer whose message is being redirected.

EBX003-EBX005
CRI of the redirection printer (the printer to which this message is being sent).

Program APR8 and any program that APR8 calls can use the ECB work areas but must not issue any SERRC macro calls.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

ALCS tests the following return conditions from APR8:

R15=0
Add header to the message.

R15=4
Do not add header to the message.

Program APR8 must issue BACKC to return to the calling program to send the message to the redirection printer.

End of redirection processing - APR9

ALCS conditionally enters APR9 after sending the redirected message. This allows the application to perform any processing required before ALCS redirection processing is finished.

ALCS conditionally enters APR9 with the following conditions:

EBX000-EBX002
CRI of the printer whose message is being redirected.

EBX003-EBX005
CRI of the redirection printer (the printer to which this message was sent).

Program APR9 and any program that APR9 calls, can use the ECB work area but must not issue any SERRC macro calls.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

Program APR9 must issue BACKC to return to the calling program.

ALCS printer message purge exit program - APRA

ALCS conditionally enters APRA during RCR police. This allows the application to indicate whether ALCS must purge messages on printer queues.

ALCS conditionally enters APRA with the following conditions:

Level D0
RCR record (not held).

Register
Contents

R00
CRI of the printer (right adjusted).

R14
Total number of messages on all priority queues for the printer. Or zero if R15 is X'FFFFFFFF'

R15
Age (in days) of the oldest message on any priority queue for the printer (rounded down to the nearest whole number of days). Or X'FFFFFFFF' if the first message on a priority queue is a short-term pool record.

Program APRA, and any program that APRA calls, must not:

- Modify the ECB work area.
- Issue any SERRC macros.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

Program APRA must issue BACKC to return to the calling program to continue RCR police.

APRA must return with the RCR record on level D0 (not held).

ALCS tests the following return conditions from APRA:

R15=0
Do not purge or discard printer messages.

R15=*number_of_days*
Purge all printer messages queued more than *number_of_days* ago. Note that short-term pool records are not purged, but are discarded.

Printer Queue threshold exit programs - APRB and APRC

These two exit programs allow specific actions to be taken when the number of messages on a printer queue reaches or exceeds the threshold limit. The threshold limit is defined in the QWARNING parameter on the ALCS communication generation COMGEN macro. The purpose of the first exit program that is activated (program APRB) is to decide if any action should be taken for the specific printer that has reached or exceeded the threshold limit. That action could be, for example, to issue a warning message. For some printers you may wish to take no action, but for others you may have specific actions you wish to take. The purpose of the second exit program that is activated (program APRC) is to take the actions that are required. In summary, exit program APRB just identifies the printers for which actions are required, and exit program APRC must perform the required actions. ALCS activates exit program APRC only if exit program APRB has requested it.

You can implement the second exit program (APRC) without implementing the first exit program (APRB). Do this when you require actions to be taken for every printer.

Printer Selection Exit - APRB

ALCS conditionally enters APRB, each time the printer queue threshold is reached or is exceeded as defined in the QWARNING parameter on the ALCS communication generation COMGEN macro.

ALCS enters APRB with the following conditions:

EBX001-EBX003
CRI of the printer

EBX004-EBX011
CRN of the printer

EBX012-EBX015
Total number of messages queued for the printer

EBX016-EBX019
Printer queue threshold

EBX020-EBX023
Printer queue frequency

Program APRB and any program that APRB calls, **must not**:

- Modify EBW000 through EBW103
- Issue any SERRC, WTOPC, CRASC, SLMTC, or SENDC L macros.

ALCS tests the following return conditions from APRB:

R15=0
Invoke the APRC exit program for this printer.

R15<>0
Do not invoke the APRC exit program for this printer.

Program APRB must issue BACKC to return to the calling program.

Printer Selection Exit - APRC

ALCS conditionally enters APRC asynchronously, (that is, a new ECB is created) each time the printer queue threshold is reached or is exceeded as defined in the QWARNING parameter on the ALCS communication generation COMGEN macro.

The APRC exit program can take any actions that are required; for example, it can issue a warning message.

ALCS enters APRC with the following conditions:

EBX001-EBX003
CRI of the printer

EBX004-EBX011
CRN of the printer

EBX012-EBX015
Total number of messages queued for the printer

EBX016-EBX019
Printer queue threshold

EBX020-EBX023
Printer queue frequency

When program APRC has performed the required actions, it must issue EXITC or BACKC to exit.

Retrieve output display exit program - ARE1

ARE1 allows you to modify the output which will be sent to the terminal by the ALCS retrieve facility as the result of a ZRETR Prev or ZRETR Next or ZRETR Display command.

ALCS calls ARE1 just before the message is sent to the terminal. This could be used, for example, to suppress a password which is contained in the message to be written to the screen.

If no ARE1 is provided, the message is sent unmodified.

ALCS enters ARE1 with the following conditions:

R14
Address of the message.

R15
Length of the message.

The exit must not change the contents of the ECB levels. EBW013-EBW104 in ECB Work Area 1 is available. ECB Work Area 2 (EBX000-EBX104) is not available.

ARE1 must return using BACKC.

R14
Must be unchanged.

R15
Must contain the length of the message.

The contents of all other registers are irrelevant.

Following the return from ARE1, the message is sent to the terminal.

ZROUT/ZACOM routing exit program - ARO1

ARO1 allows you to control the applications that a particular user is allowed to route to. By checking the supplied CRI and application CRN you can permit or reject the routing request.

ARO1 is entered by ZROUT or ZACOM command processing for a request to establish, change or remove the routing for a resource.

ALCS enters ARO1 with the following conditions:

EBROUT
CRI of originating terminal.

EBW000
Length of application name or zero if routing is to be removed.

EBW001-EBW008
Application name padded with blanks.

EBW012-EBW014
CRI of resource to be updated.

Level D0
Input message (CM1CM format).

ALCS tests the following return conditions from ARO1:

R15=0
Continue ZROUT or ZACOM processing.

R15¬=0
Terminate ZROUT or ZACOM processing.

ARO1 must not use EBX000-EBX103.

ARO1 is conditionally entered by ENTRC and control must eventually return by BACKC.

State change exit programs - ASC1 through ASC4

Start of state change - ASC1

The ZASYS command processor conditionally enters ASC1 when the system state changes.

ALCS enters ASC1 with the following conditions:

EBSW01
The code for the "from" system state, one of:

Code	Symbol	From state
X'00'	CCHALT	Halt
X'10'	CCNORM	Normal
X'20'	CCMESW	Message switching
X'40'	CCCRAS	CRAS
X'80'	CCIDLE	Idle

EBSW02
The code for the "to" system state. These are the same as the codes for the "from" system states (see above).

No ECB storage or data levels are in use.

Program ASC1, and any program that ASC1 calls, can modify EBW000 through EBW103 and EBX000 through EBX103.

Program ASC1 must issue BACKC to return to the calling program. Otherwise the ALCS state change fails to complete.

End of state change - ASC2

The ZASYS command processor conditionally enters ASC2 when the ALCS state change completes.

ALCS enters ASC2 with the following conditions:

EBSW01
The code for the old ALCS system state, one of:

Code	Symbol	System state
X'00'	CCHALT	Halt
X'10'	CCNORM	Normal
X'20'	CCMESW	Message switching
X'40'	CCCRAS	CRAS
X'80'	CCIDLE	Idle

EBSW02
The code for the new system state. These are the same as the codes for the old system states (see above).

No ECB storage or data levels are in use.

Program ASC2, and any program that ASC2 calls, can modify EBW000 through EBW103 and EBX000 through EBX103.

Program ASC2 must issue BACKC to return to the calling program. Otherwise, the ALCS state change fails to complete.

State change from IDLE to higher state - ASC3

The ZASYS command processor conditionally enters ASC3 when the ALCS state changes from IDLE state to CRAS, MESW, or NORM state.

No ECB storage or data levels are in use.

ASC3 and any program that ASC3 calls, can modify EBW000 through EBW103 and EBX000 through EBX103.

Program ASC3 must issue BACKC to return to the calling program. Otherwise, the ALCS state change fails to complete.

State change to IDLE from higher state - ASC4

The ZASYS command processor conditionally enters ASC4 when the ALCS state changes to IDLE state from CRAS, MESW, or NORM state.

No ECB storage or data levels are in use.

Program ASC4, and any program that ASC4 calls, can modify EBW000 through EBW103 and EBX000 through EBX103.

Program ASC4 must issue BACKC to return to the calling program. Otherwise, the ALCS state change fails to complete.

External security manager display exit program - ASD1

The ALCS ZDCOM command processor conditionally enters this program before displaying the installation-defined data from the external security manager connected GROUP profile for the end user who is currently logged on to the resource, in response to ZDCOM with the SAF parameter. This allows the user to do any formatting of the SAF data before it is displayed, or to suppress the display.

ALCS enters ASD1 with the following conditions:

EBX000-EBX084
ZDCOM work area

Level D0
Reserved

Level D2
Reserved

Level D3
Reserved

Level D4
Reserved

Level D5
Work area containing the external security manager data.

Register Contents

R05
The address of communication system data for the resource.

R06
The address of external security manager data for the resource.

R07
The length of external security manager data for the resource.

ASD1 is conditionally entered by ENTRC and control must eventually return by BACKC.

Program ASD1, and any program that ASD1 calls must not:

- Modify EBX000 through EBX084.
- Use ECB storage levels 0 or 2 through 4.
- Use ECB storage level 5, unless it is to reformat external security manager data.
- Modify registers R01, R03, R04, or R05.

The ALCS ZDCOM command processor tests the following return conditions from ASD1:

Register
Contents

R15=0
Continue ZDCOM processing. ALCS displays external security manager data in hexadecimal and character representation.

R06
Must contain the address of the data to be displayed.

R07
Must contain the length of the data to be displayed.

R15=4
Terminate ZDCOM processing. ALCS sends a response to the originating terminal without displaying any of the external security manager data.

R15=8
Continue ZDCOM processing. ALCS displays the external security manager data in a character string that is appended on a new line directly after the ZDCOM response message header.

R06
Must contain the address of the data to be displayed.

R07
Must contain the length of the data to be displayed.

ALCS e-mail exit programs - ASM0 through ASM3, ASM5, ASM6, and ASMA

These programs allow customization of the ALCS e-mail facility. You should also read the description of the ZMAIL command in *ALCS Operation and Maintenance*.

Outgoing e-mail customization program - ASM0

ALCS conditionally enters ASM0 before transmission of an outgoing Simple Message Transfer Protocol (SMTP) message. This allows the application to add extra lines to outgoing messages. For e-mail

messages constructed using the ZMAIL command, ALCS calls ASM1 (see “ZMAIL message customization program - ASM1” on page 394) and then calls ASM0.

ALCS conditionally enters ASM0 with the following conditions:

ECB Level D0
The output e-mail message. The DXCSMTM macro describes the format of the message. On entry to ASM0, the output message text starts at label SMTPTEXT, and the fullword field SMTPLENG contains the length (number of bytes) of the message in binary. There is no pool file address assigned to the prime or to a single block. When an email consists of more than one block, the blocks are chained using the standard forward chain field. The DXCSMTM macro describes the format of the message.

The prime block is not assigned a pool file address. Pool file addresses are also not assigned to single blocks used when only one block is needed.

When email consists of more than one block, the blocks are chained using the standard forward chain field (SMTPHFCH). The BID of each block must be X'AB10'

ASM0 can modify the output message as required. On return from ASM0, the output message text (modified or not) must start at label SMTPTEXT, and SMTPLENG must contain the length.

ASM0 does not need to save and restore registers, but it must save and restore any ECB work areas and ECB levels which it modifies.

ASM0 must return using BACKC.

ZMAIL message customization program - ASM1

ALCS conditionally enters ASM1 before transmission of an outgoing Simple Message Transfer Protocol (SMTP) message that has been built using the ZMAIL command. This allows the application to add extra lines to outgoing messages. For e-mail messages constructed using the ZMAIL command, ALCS calls ASM1 and then calls ASM0 (see “Outgoing e-mail customization program - ASM0” on page 393).

ALCS conditionally enters ASM1 with the following conditions:

ECB Level D0
The output e-mail message. The DXCSMTM macro describes the format of the message. On entry to ASM1, the output message text starts at label SMTPTEXT, and the fullword field SMTPLENG contains the length (number of bytes) of the message in binary.

If the message was entered by an ALC terminal, bit SMTPZALC is on. (Note that for ALC input, ALCS converts the input to lower case, substitutes tokens, and so on, before it calls ASM1.)

ASM1 can modify the output message as required. On return from ASM1, the output message text (modified or not) must start at label SMTPTEXT, and SMTPLENG must contain the length.

ASM1 does not need to save and restore registers, but it must save and restore any ECB work areas and levels which it modifies.

ASM1 must return using BACKC.

Incoming e-mail header selection program - ASM2

ALCS conditionally enters ASM2 when it receives an incoming Simple Message Transfer Protocol (SMTP) message. ASM2 can select which message header lines in the e-mail message ALCS will pass to the recipient.

When ASM2 is not used, and the recipient is Postmaster or an ALCS application, ALCS passes every header line.

When ASM2 is not used, and the recipient is a terminal, ALCS passes the following header lines:

Date
Subject
From
To

Cc
Bcc
Sender
Reply-to

For each recipient (if any), and for each header line, ALCS conditionally enters ASM2 with the following conditions:

EBW000
8-byte CRN of the recipient.

EBW008
3-byte CRI of the recipient.

EBW011
1-byte recipient type code, one of the following binary values:

Code	Recipient type
0	3270 terminal
1	ALC terminal.
3	Postmaster.
4	ALCS application.

EBW012
1-byte binary length of the header name.

EBW013
Header name in upper-case characters.

ALCS tests the following return conditions from ASM2:

R15=0
ALCS retains the header in the message.

R15=8
ALCS removes the header from the message.

If the incoming e-mail message is a Multipurpose Internet Mail Extensions (MIME) multipart message, ALCS also conditionally enters ASM2 with the following special header names:

--
ALCS conditionally enters ASM2 with this header name for each MIME multipart boundary line. ASM2 can return with 0 in register 15 to include the boundary line in the message or 8 in register 15 to remove it.

X-ALCS-ACTION
ALCS conditionally enters ASM2 with this header name for the special header lines which ALCS itself inserts into some incoming e-mail messages. ALCS only adds this type of header line to indicate where it has removed a part from a multipart input message. ASM2 can return with 0 in register 15 to include the header line in the message or 8 in register 15 to remove it.

ASM2 must return using BACKC.

Incoming e-mail MIME-part selection program - ASM3

ALCS conditionally enters ASM3 when it receives an incoming Simple Message Transfer Protocol (SMTP) message which contains Multipurpose Internet Mail Extensions (MIME). ASM3 can select which Multipurpose Internet Mail Extensions (MIME) message parts in the e-mail message that ALCS will pass to a recipient.

For each recipient, and for each MIME `Content-Type` header which specifies a media type other than multipart, message, or text, ALCS conditionally enters ASM3 with the following conditions:

EBW000
8-byte CRN of the recipient.

EBW008
3-byte CRI of the recipient.

EBW011
1-byte recipient type code, one of the following binary values:

Code	Recipient type
0	3270 terminal
1	ALC terminal
3	Postmaster
4	ALCS application

EBW012
1-byte binary length of the MIME media type and subtype.

EBW013
MIME media type and subtype, upper-case characters in the format *type*/*subtype*.

ALCS tests the following return conditions from ASM3:

R15=0
ALCS retains the message part in the message.

R15=8
ALCS removes the message part from the message.

ASM3 must return using `BACKC`.

ALC e-mail custom conversion program - ASM5

This program can customize how the `ZMAIL` command converts messages entered at an ALC terminal for transmission using the Simple Message Transfer Protocol (SMTP).

The following types of customization are possible:

- ASM5 can perform the entire conversion itself and bypass the conversion normally done by ALCS.
- ASM5 can provide one or more custom conversion tables which ALCS then uses instead of its default conversion tables.

Note: If you implement custom conversion, you must provide documentation for your end users which explains how your custom conversion affects the messages they enter using `ZMAIL`. (Note that end users can bypass your ASM5 customization by using the BASIC parameter on the `ZMAIL` command. In this case, ALCS does not call ASM5.)

If you have ASM0 or ASM1 installation-wide exits, ALCS calls ASM5 before it calls ASM0 or ASM1.

If the `ZMAIL` command does not specify BASIC, ALCS conditionally enters ASM5 with the following conditions:

EBW000
Fullword, contains the address of the message text.

EBW004
Fullword, contains the length (number of bytes) in binary of the message text.

EBW008
Fullword address of the target area for the text conversion. If your ASM5 program converts the text itself, then it must store the converted text in this target area.

EBW012
Fullword length of the converted text (not initialized on entry to ASM5). If your ASM5 program converts the text itself, then it must store the length of the converted text in EBW012.

EBW016
Reserved fullword, do not use this field.

EBW020
One byte length (number of bytes) in binary of the convert table selector token. The convert table selector token is optionally included as a parameter of the ZMAIL command. If the convert table selector token is omitted from the ZMAIL command then EBW020 contains binary zeros.

EBW021
The convert table selector token. If your ASM5 replaces this token, it must save the length of the replacement token in EBW020.

When ASM5 returns to ALCS, the contents of general register 15 (RDB) must be set as follows:

- If ASM5 does the message conversion, it must set register 15 to binary zeros. In this case, ASM5 must store the converted text at the address passed in EBW008 and the length of the converted text in EBW012.
- If ASM5 does not provide custom conversion tables for this particular message, it must return with general register 15 unmodified (the same as at entry to ASM5). In this case, ASM5 can set EBW020 to binary zeros (indicating that ALCS must use the default convert tables).
- If ASM5 provides one or more custom convert tables, it must set general register 15 to the address of a table of one or more fullword addresses. Each fullword contains the address of a custom convert table. A fullword of binary zeros must follow the last convert table address.

ASM5 does not need to save and restore registers, but it must save and restore any ECB work areas and levels (other than the work area 1 fields listed above) which it modifies.

ASM5 is conditionally entered by ENTRC, and control must eventually return by BACKC.

Custom ALC convert tables

If your ASM5 program does the ALC conversion itself then it does not need to include custom convert tables. Otherwise use the DXCAMT macro (see “Using the DXCAMT macro” on page 399) to build custom convert tables in ASM5.

Your ASM5 can include more than one custom table. For example, it might contain separate tables for converting messages in the English and Spanish languages, selected by the tokens ENGLISH and ESPANOL (with SPANISH as a synonym), like this:

```
         CLC   ESPSYN,EBW020       'SPANISH' SPECIFIED
         BNE   NOTSYN              NO - PROCEED

         MVC   EBW020(L'ESPNAME),ESPNAME CORRECT TO 'ESPANOL'

NOTSYN   DC    0H'0'
         LA    R15,MYTABLE         ADDRESS CUSTOM TABLE LIST
         BACKC ,                   RETURN TO ALCS

MYTABLE  DC    A(ENGLISH)          ADDRESS OF ENGLISH TABLE
         DC    A(ESPANOL)          ADDRESS OF SPANISH TABLE
         DC    A(0)                END OF LIST

ESPSYN   DC    0CL8' ',AL1(7),C'SPANISH' SYNONYM FOR SPANISH
ESPNAME  DC    0CL8' ',AL1(7),C'ESPANOL' SELECTOR FOR SPANISH

ENGLISH  DXCAMT START              ENGLISH TABLE
         ...                       (subtables for English)
         DXCAMT END

ESPANOL  DXCAMT START              SPANISH TABLE
```

```
...                              (subtables for Spanish)
DXCAMT END
```

Each custom convert table contains three subtables. These are:

SHIFTS
This subtable specifies how ALCS processes shifted characters. Shifted characters allow an end user to enter a punctuation mark with only two key-strokes instead of the three or four characters they need for a token.

ZMAIL uses the star (*) as the shift character. It interprets a star (shift character) and the immediately following character (shifted character) as a pair, and replaces the pair with a single character, as follows:

1. ALCS looks-up the shifted character in the shifts subtable. If the subtable includes the shifted character then ALCS replaces the pair with the replacement character.
2. If the subtable does not include the shifted character then ALCS checks if the shifted character is alphabetic. If it is alphabetic, ALCS replaces the pair with the corresponding upper case character (capital letter).
3. If the shifted character is not alphabetic, ALCS replaces the pair with the unchanged shifted character.

After replacing the pair, ALCS checks if the resulting character is a full stop; that is a period (.), question mark (?), or exclamation mark (!) followed by a space or new line character. If it is a full stop, ALCS capitalizes the first character of the following text (if any).

TOKENS
This subtable specifies how ALCS processes tokens. Tokens allow an end user to enter a punctuation mark which they do not have on their ALC keyboard, or which their terminal transmits with a nonstandard binary encoding.

ZMAIL uses the slash (/) as the token delimiter character. ALCS interprets the character string which immediately follows the slash as a token. ALCS identifies the end of a token by a delimiter which can be either a slash or any other character which is not alphabetic or numeric. It processes each token as follows:

1. ALCS looks-up the token in the tokens subtable. If the subtable includes the token then ALCS replaces the initial slash, the token itself, and the trailing slash (if any) with the replacement character or string.

   If the subtable identifies the token as a full stop, ALCS capitalizes the first character of the following text (if any).
2. If the subtable does not include the token then ALCS leaves the initial slash unmodified, translates the token itself to lower case, and leaves the trailing slash (if any) unmodified.

Note that ALCS does not capitalize the first letter of a token, or its replacement (if any), even if the token follows a full stop.

WORDS
This subtable specifies how ALCS processes words. You can use this subtable to recognize any occurrence of a particular word entered at an ALC terminal and replace it with a specified character string.

For example, you may have particular rules for capitalizing the name of your organization, or other words that your end users often need in messages.

In this context, a "word" is any string of letters or numbers (or a mixture of letters and numbers) which does not contain spaces, new lines, or punctuation marks.

Note: ALCS does *not* recognize words which immediately follow a slash (/) or a star (*). For example, ALCS does not recognize 'IBM' as a word in '/IBM' or '*IBM'.

ALCS processes words as follows:

1. It looks-up the word in the words subtable. If the subtable includes the word then ALCS replaces the word with the replacement character or string.

   ALCS does not capitalize the first letter of the replacement word even if the word follows a full stop.

   If the subtable identifies the word as a full stop, ALCS capitalizes the first character of the following text (if any).
2. If the subtable does not include the word then ALCS translates the word to lower case. If the word follows a full stop, ALCS capitalizes the first letter.

Using the DXCAMT macro

Use the DXCAMT macro to build a table of formatting rules for Simple Message Transfer Protocol (SMTP) e-mail messages for ALC terminal equipment. You can include one or more of these tables in your ASM5 installation-wide exit program (see "ALC e-mail custom conversion program - ASM5" on page 396). The format of the DXCAMT macro is:

To start a table:

```
label DXCAMT START
```

To start a subtable:

```
DXCAMT {SHIFTS|TOKENS|WORDS}
            [,USE={DEFAULT|name]
```

To build an entry in a subtable:

```
DXCAMT ALC=('alc'[,minlen]),EBCDIC='ebcdic'
            [,TYPE=({FULLSTOP][,KEYIN|DISPLAY])]
```

To end a table:

```
DXCAMT END
```

Where:

label
Name of this custom ALC e-mail conversion table, up to 16 characters. You can refer to this name in an assembler language A-type address constant. It is also the convert selector token that your ALC end users specify on the ZMAIL command to request conversion of their e-mail using this table.

Do not use any of the following reserved names: ALC, BATCH, BASIC, DEBUG, DEFAULT, or TRACE.

START
Start the custom ALC e-mail conversion table.

SHIFTS
Start the shifts subtable. This subtable specifies how the ALC e-mail shift character (*) affects the following character in an e-mail message entered at an ALC terminal.

If you omit the shifts subtable, or include a shifts subtable with no entries then the shift character affects the following character as follows:

- An alphabetic character changes to upper case (capital letter)
- Any other character does not change.

TOKENS
Starts the tokens subtable. This subtable lists the tokens for ALC e-mail messages and specifies the corresponding EBCDIC character or character string.

Tokens are character strings delimited by a starting slash (/) and an ending slash or an ending character which can be anything other than an alphabetic or numeric character. The starting slash, the token, and the ending slash (if any) in an ALC e-mail message represents the corresponding EBCDIC character or character string.

If you omit the tokens subtable, or include a tokens subtable with no entries the ALCS does not replace tokens (or the delimiting slash characters).

WORDS
Starts the words subtable. This subtable lists words which can be used in ALC e-mail messages which you want to replace with a specific character string. For example, you may want to replace the word 'IBM' with 'IBM'. (If you do not do this, 'IBM' will convert to 'ibm' or 'Ibm', depending on whether or not it follows a full stop.)

In this context, a "word" is any string of letters or numbers (or a mixture of letters and numbers) which does not contain spaces, new lines, or punctuation marks.

Note: ALCS does *not* recognize words which immediately follow a slash (/) or a star (*). For example, ALCS does not recognize 'IBM' as a word in '/IBM' or '*IBM'.

If you omit the words subtable, or include a words subtable with no entries the ALCS does not replace words.

USE={DEFAULT|*name*}
Use the default subtable (DEFAULT), or the corresponding subtable from the table *name* - that is, from another custom ALC convert table in your ASM5 program which uses *name* as the label on its DXCAMT START macro.

Note:

In the following two parameters, you must follow the normal assembler language rules for quoted strings. In particular, you must double-up the single quote (') and ampersand (&) characters.

Also, IBM 3270 and compatible terminals and emulators support a variety of different code pages. ALCS interprets the following parameters 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 these parameters. For example, if you are using a Brazilian EBCDIC code page (00273 or 00275) you must use the Ã character in place of @.

ALC=('*alc*'[,*minlen*])
ALC representation, where:

alc
For a shifts subtable entry, a single character which is not alphabetic or numeric. For a tokens or words subtable entry, a string of upper case alphabetic or numeric characters (or a mixture of upper case alphabetic and numeric characters).

minlen
For a shifts subtable entry, this parameter is ignored. For a tokens or words subtable entry, a decimal number which is the length of the shortest abbreviation of the character string allowed in ALC ZMAIL commands. If omitted, this parameter defaults to the actual length of the string (meaning that no abbreviation is allowed).

EBCDIC='*ebcdic*'
EBCDIC representation.

For a shifts subtable entry, *ebcdic* is a single character. For a tokens or words subtable entry, *ebcdic* is a string of one or more characters, mixed case.

TYPE=([FULLSTOP][,KEYIN|DISPLAY])
For a shifts subtable entry, this parameter is ignored.

For a tokens or words subtable entry:

FULLSTOP
Token or word indicates end of sentence. When converting ALC ZMAIL input, ALCS capitalizes the first letter of the following word (if any).

For a tokens subtable entry:

KEYIN
Token is used only when converting ALC ZMAIL input.

DISPLAY
Token is used only when displaying e-mail at an ALC terminal.

END
End the custom ALC e-mail conversion table.

Incoming e-mail application authorization program - ASM6

ALCS conditionally enters ASM6 when it receives an incoming Simple Message Transfer Protocol (SMTP) message which specifies an application CRN as the destination. Use ASM6 to authorize particular applications to receive inbound e-mail messages. If ASM6 is not used, ALCS rejects any inbound e-mail message which specifies an application CRN as the destination.

For every inbound e-mail message which specifies an application CRN as the destination, ALCS conditionally enters ASM6 with the following conditions:

EBW000
8-byte CRN of the destination application, left justified and padded with space (blank) characters.

ALCS tests the following return conditions from ASM6:

R15=0
The application is not authorized to receive inbound e-mail.

R15=4
The application is authorized to receive inbound e-mail.

ASM6 must return using BACKC.

Outgoing e-mail sender program - ASMA

ALCS conditionally enters ASMA after transmission of an outgoing Simple Message Transfer Protocol (SMTP) message. This allows (for example) to update user statistical information.

At entry, Register R15 contains the hexadecimal ALCS SMTP sender return code:

00
Message sent.

04
Transient error. ALCS will retry the transmission.

08
Rejected by MTA. The ALCS outbound e-mail queue handler was not able to send an e-mail SMTP message to the local message transfer agent (MTA). Accompanying error messages describe the error condition. ALCS discards the rejected message and tries to send the next message from the queue.

0C
Rejected by ALCS. The ALCS outbound e-mail queue handler was not able to send an e-mail SMTP message to the local message transfer agent (MTA). Accompanying error messages describe the error condition. ALCS discards the rejected message and tries to send the next message from the queue.

10
MTA connection problem. The ALCS outbound e-mail queue handler was not able to send an e-mail SMTP message to the local message transfer agent (MTA). Accompanying error messages describe

the error condition. ALCS leaves the messages on queue. ALCS will try to contact the MTA again at regular intervals while there are messages on queue.

14
TCP/IP connection problem. ALCS is not currently connected to TCP/IP. ALCS leaves the messages on queue. ALCS will try to contact the MTA again at regular intervals while there are messages on queue.

Programming considerations when using ASMA:

- ASMA does not need to save and restore any registers
- ASMA must save and restore any ECB work areas
- ASMA may use data levels D7 to DF. However ASMA must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels used by ASMA
- ASM6 must return using BACKC.

TCP/IP concurrent server exit program - ATCP

When the ALCS TCP/IP concurrent server (Listener) is started, it waits for connections on a specified TCP/IP port. When a client connects to that port, the ALCS concurrent server creates a new ECB and passes control to the installation-wide exit program ATCP if it exists.

ALCS enters ATCP with the following conditions:

CE1SOCK
The 4-byte TCP/IP socket descriptor for the connection.

CE1PORT
The 4-byte port number for the connection.

CE1LIST
The 4-byte concurrent server index number (1 to 8). Up to eight concurrent servers can be started at the same time, each waiting for inbound connections on a different port number.

No storage or data levels are in use.

Program ATCP (and programs that ATCP calls) should use TCP/IP socket calls to communicate with the TCP/IP client. The TCP/IP connection that was established by the client is identified by the socket descriptor (the descriptor number is in the ECB field CE1SOCK). The socket descriptor is preserved across enter/back macros but can not be passed between entries, therefore the entry that activated ATCP is responsible for responding to the client via TCP/IP socket calls. When ATCP (and programs that ATCP calls) has completed its processing, it should:

- Ensure that the TCP/IP connection has been closed
- Issue the EXITC macro to terminate processing (it should not return to ALCS using BACKC)

The TCP/IP port numbers which the concurrent server (Listener) will use are defined by the TCPPORT parameter on the SCTGEN generation macro, or they are defined on the `ZCTCP LISTEN,START` command.

Attention:

If you plan to use the ALCS TCP/IP concurrent server, please contact IBM for advice on the implementation of the ATCP installation-wide exit.

ALCS throttle table exit - ATH1

ALCS conditionally enters ATH1 when updating the throttle table. ATH1 must return with a value in R15 (range is 0 through 99). If this value exceeds the non-zero USR threshold of a throttle application, then the restricted (and not the normal) throttle values defined for this throttle application will be used by the throttle. See "ALCS Throttle" in *ALCS Operation and Maintenance* for a description of the ALCS throttle.

There are no special entry conditions for ATH1.

Program ATH1, and any program that ATH1 calls, must not:

- Modify EBW000 through EBW103 or EBX000 through EBX103.
- Use ECB storage levels.
- Modify registers R00 through R07.

ATH1 must return using BACKC.

On return from ATH1, register R15 must contain a binary value in the range 0 through 99.

ZATIM command exit program - ATM1

ALCS conditionally enters ATM1 before processing a ZATIM command that changes the local or GMT time. Use this exit to allow the command (with or without command confirmation) or to disallow it.

ALCS enters ATM1 with the following conditions:

EBROUT
CRI of originating terminal.

EBX008-EBX011
Local minute-in-day - current.

EBX012-EBX015
GMT minute-in-day - current.

EBX016-EBX019
Local minute-in-day - change.

EBX020-EBX023
GMT minute-in-day - change.

EBX024-EBX027
Indicator for local date change:

1
for positive date change.

0
for no date change.

-1
for negative date change.

EBX028-EBX031
Indicator for GMT date change:

1
for positive date change.

0
for no date change.

-1
for negative date change.

Level D0
Input message (CM1CM IMSG format).

Level D2
Parsed input message.

ALCS tests the following return conditions from ATM1:

R15=0
Use default time change criteria:

Negative date change is not allowed
Positive date change is allowed with confirmation.

R15=4
Allow time change with no confirmation.

R15=8
Allow time change with confirmation, the text depends on the contents of R01 as follows:

R01=0
Use default confirmation text:

```
DXC8515I CMD M hh.mm.ss CONF Are you sure?
Reenter within 30 seconds to confirm
```

R01<>0
User-supplied confirmation text. R01 contains the address of user confirmation text in the form:

```
AL1(length),C'confirmation_text...'
```

R15=12
Do not allow time change, the text depends on the contents of R01 as follows:

R01=0
Use default rejection text:

```
DXC8646E CMD M hh.mm.ss. ATIM Not allowed
Reject by installation exit
```

R01<>0
User-supplied rejection text. R01 contains the address of user rejection text in the form:

```
AL1(length),C'rejection_text...'
```

ATM1 must issue BACKC to return to the calling program. ATM1 can modify registers R01, R14, and R15. It must not modify any other registers or ECB fields.

Third party exit - ATRD

ALCS calls this exit during the conversational trace display routine.

This exit is intended for use with third party software. For more information contact the ALCS Support Group (email alcs@uk.ibm.com).

Third party exit - ATRE

ALCS calls this exit when processing conversational trace messages.

This exit is intended for use with third party software. For more information contact the ALCS Support Group (email alcs@uk.ibm.com).

Trace exit programs - ATR1, ATR2, and ATR9

Implement user trace commands - ATR1

ALCS conditionally enters ATR1 if the user attempts to use an unsupported trace command during a conversational trace. Use ATR1 to implement user trace commands.

Note that ALCS reserves all commands that start with M or O for user trace commands.

ALCS enters ATR1 with the following conditions:

EBX000
The 4-byte address of the ECB of the entry that is being traced.

Level D0
The input message (trace command); the CM1CM macro describes the format of the message.

ATR1 must either process the command, or issue BACKC to return to the calling program (if the command is not valid).

Display user ECB fields - ATR2

User exit ATR2 is a table of user labels in the ECB. The table must conform to the same format as CGTMECBT in ECB-controlled program CGTM.

For each user ECB label, there is one 12 byte entry, and the table must be delimited by a null entry of 12 bytes.

A sample exit is provided with the product in the installation-wide exit library.

Remote terminal trace response program - ATR9

When an agent uses conversational trace to trace input messages from another (remote) terminal, the agent can cancel processing for a particular input message before the application sends a response.

In this case, ALCS normally sends message DXC8464I to the remote terminal. This message tells the user at the remote terminal that their input message has been cancelled and unlocks their terminal so that they can enter another input message.

Some ALCS applications handle input messages and responses for terminals that connect to ALCS through a communication link, for example an LU6.1 link. That is, the application implements a function similar to the ALCS message router facility.

These applications typically include:

- A receiver component which handles input messages from the link
- A sender component which transmits responses across the link.

The receiver component determines the CRI of the originating (OSYS) terminal, saves this originating CRI in the EBROUT field in the ECB, and passes the input message (usually in IMSG format) to the rest of the application.

For this type of input, you do not want ALCS to send message DXC8464I to the "originating CRI". Instead, you can use ATR9 to build an appropriate response and transmit the response across the communication link.

ALCS conditionally enters ATR9 before it sends message DXC8464I. The entry conditions to ATR9 are:

EBROUT
CRI of the originating terminal

If ATR9 issues BACKC, ALCS sends the default message DXC8464I to the CRI in EBROUT.

Alternatively, ATR9 can (for example) build and transmit a response message across a communication link. In this case, ATR9 must issue EXITC.

Unsolicited message exit programs - AUM1 through AUM4

These programs allow customization of the ALCS unsolicited message package. You should also read the description of the ZSNDU command in *ALCS Operation and Maintenance*.

Validating operands in INCLUDE and EXCLUDE lists - AUM1

ALCS conditionally enters AUM1 when a broadcast message is requested and the INCLUDE= or EXCLUDE= parameters have been supplied with the ZSNDU command.

Each of the INCLUDE= and EXCLUDE= parameters specified in the ZSNDU command must correspond to one of a predefined list of parameters in AUM1 and AUM2. ALCS enters AUM1 to validate the

INCLUDE= and EXCLUDE= parameters. If AUM1 is not loaded, INCLUDE= and EXCLUDE= parameters are not allowed.

ALCS conditionally enters AUM1 with the following conditions:

EBX066
The INCLUDE= and EXCLUDE= parameter list (maximum length 20 bytes).

EBX064
The length of the parameter list (2 bytes).

ALCS tests the following return conditions from AUM1:

R01=0
One or more parameters are not valid.

R01=1
All parameters are valid.

AUM1 must not alter the ECB work areas. All registers (except R15) must be the same as at entry.

AUM1 must return using BACKC.

Checking a terminal against INCLUDE and EXCLUDE lists - AUM2

ALCS conditionally enters AUM2 to determine whether a given broadcast message should be sent to a particular terminal.

If INCLUDE= or EXCLUDE= parameters are used for the message, AUM2 checks if the selection criteria for the message are satisfied by a given terminal.

If INCLUDE= or EXCLUDE= parameters have not been used for this message, all terminals are considered to meet the selection criteria.

ALCS enters AUM2 with the following conditions:

EBX066
The INCLUDE= and EXCLUDE= parameter list (maximum length 20 bytes).

EBX064
The length of the parameter list (length 2 bytes).

EBX088
The address of a work area, where COMIC data for the requesting terminal is held.

ALCS tests the following return conditions from AUM2:

R01=0
The message should not be sent to this terminal.

R01=1
The message should be sent to this terminal.

AUM2 must not alter the ECB work areas. All registers must be the same as at entry.

You can supply AUM1 without AUM2; however AUM1 serves no useful purpose without AUM2. Because AUM1 and AUM2 both need to have the same table of valid INCLUDE= and EXCLUDE= parameters, they can be transfer vectors in the same program.

A sample AUM1 program with transfer vectors AUM1 and AUM2 is supplied in the installation-wide exit library. You can modify this sample program to include your own selection of INCLUDE= and EXCLUDE= parameters.

AUM2 must return using BACKC.

Unsolicited message timeout - AUM3

About this task

When an unsolicited message is received by ALCS, it is queued until requested by the destination terminal. When a certain time (specified by the timeout value) has elapsed, the message is purged from the queue, even if the destination terminal has not requested it.

This timeout value can be specified in three ways:

Procedure

1. By the operator who originates the message using the ZSNDU command. This is explained in the *ALCS Operation and Maintenance*. This overrides the value from AUM3 or the ALCS default.
2. By the installation-wide exit program AUM3. If program AUM3 is loaded and the ZSNDU command is not used, the value from AUM3 overrides the ALCS default.
3. By the ALCS default (3 hours). If program AUM3 is not loaded and the ZSNDU command is not used, the default is used.

Results

AUM3 sets up the required timeout value in a 3-byte work area supplied by the calling program.

ALCS enters AUM3 with the following conditions:

Register
Contents

R07
The address of the 3-byte work area.

ALCS tests the following return conditions from AUM3:

R07
The address of the 3-byte work area.

The required timeout value is set up in the 3-byte work area in one of the following ways:

C'*mm*M'
Timeout value in minutes *mm* (decimal 1 through 99).

C'*hh*H'
Timeout value in hours *hh* (decimal 1 through 99).

C'*dd*D'
Timeout value in days *dd* (decimal 1 through 28).

C'*ww*W'
Timeout value in weeks *ww* (decimal 1 through 4).

R06
Contains the number of bytes used in the work area. This is usually three, but is two if a leading zero is suppressed.

A sample AUM3 is supplied in the installation-wide exit library.

AUM3 must return using BACKC.

Unsolicited messages to one destination - AUM4

AUM4 is entered when an unsolicited message addressed to one destination only is requested by a ZSNDU command. AUM4 can be used to prevent unauthorized traffic between terminals.

ALCS enters AUM4 with the following conditions:

EBX026-EBX028
CRI of destination terminal.

EBROUT
CRI of originating terminal.

ALCS tests the following return conditions from AUM4:

R01=0
The message is not authorized.

R01=1
The message is authorized.

AUM4 can use any registers available to application programs. AUM4 must not alter the ECB work areas.

AUM4 must return using BACKC.

Scrolling package and retrieve function exit program - AUS1

Some application programs allow several users to share the same terminal. For example, IPARS-based application programs can allow agents A, B, C, D, and E to share one terminal. The ALCS scrolling package and retrieve facility allows as many as 16 users to share the same terminal. This is done by allowing the application program to provide a number (0 through 15) specifying which user's messages are being scrolled or retrieved. ALCS conditionally enters AUS1 to do this.

AUS1 can add an identifying code for the specified user-area number. If no AUS1 is provided, or if the system is not in normal state, ALCS assumes area 0.

ALCS conditionally enters AUS1 with the following conditions:

EBROUT
The CRI of the terminal representing the user whose messages are being scrolled or retrieved.

ALCS tests the following return conditions from AUS1:

R05 (RGD)
Contains the area number and (optionally) scroll-area code, as follows:

Byte
Contents

0
Scroll area code, alphanumeric, or binary zeros.

1
Reserved, must be binary zeros.

2
Reserved, must be binary zeros.

3
Scroll area or retrieve area number, binary 1-16.

If AUS1 provides an alphanumeric area code in byte 0 of R05 then ALCS includes this character in the header of scrolled output messages.

Registers R03 and R04 must not be changed by AUS1.

ECB work areas and storage levels must not be altered by AUS1.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

If you need to use part of the ECB work area, save that part before using it, and restore it before returning to the calling program.

A sample AUS1 is provided in the installation-wide exit library.

AUS1 must return using BACKC.

Scroll log and retrieve inhibit exit program - AUS2

AUS2 allows you to control which terminals can use the ALCS scroll log mode and which terminals can start the retrieve function.

ALCS calls AUS2 when a terminal issues a ZSCRL LOG START or a ZRETR START command. AUS2 decides if scroll log mode, or the retrieve function, is allowed for this terminal.

If no AUS2 is provided, all terminals are allowed to use scroll log mode and the retrieve function.

ALCS enters AUS2 with the following conditions:

EBROUT
The CRI of the terminal which enters the ZSCRL LOG or ZRETR START command.

R14=0
ZSCRL LOG START issued.

R14=4
ZRETR START issued.

ALCS tests the following return conditions from AUS2:

R14=0
Scroll log mode or the retrieve function is allowed.

R14=4
Scroll log mode or the retrieve function is not allowed.

AUS2 must return using BACKC. AUS2 may modify the registers but must not change the contents of the ECB levels. Before ALCS calls AUS2, it retrieves (with record hold) the resource control record (RCR) for the terminal.

Inhibit scrolling package exit program - AUS3

ALCS conditionally enters this program to perform any custom formatting, when scrolling is inhibited and the response is sent to a communication resource connected to ALCS over a MATIP TYPE A TCP/IP connection.

ALCS enters AUS3 with the following conditions:

Register
Contents

R05
Points to the message (block), which is in CM1CM format.

AUS3 (or any program that AUS3 calls) must not alter the ECB areas one and two.

AUS3 (or any program that AUS3 calls) must not alter registers (except registers R14 and R15).

If you require a data level, then you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

AUS3 is conditionally entered by ENTRC, and control must eventually return by a BACKC.

An example of AUS3 is provided with the ALCS product in the installation-wide exit library.

Application utility control exit programs - AUT1, AUT2, and AVCT

Many applications written for use with ALCS or TPF include utility functions. For example, the IPARS - ALCS V2 reservations and message-switching applications include utility functions such as:

File maintenance
Queue maintenance
Availability recap
Authorization percentage

Message switching purge
Message switching retrieval.

Usually these utilities must run one at a time. For example, file maintenance cannot run at the same time as queue maintenance, and so on. Applications enforce this by using switches in the application global area. Typically they use 32 switch bits in a fullword field called @SWITCH.

One @SWITCH bit is associated with each utility. When an operator tries to start one of the utilities, the utility checks the corresponding bit in @SWITCH, and either:

- Rejects the request, because another conflicting utility is already running
- Sets the bit on, to indicate that the utility is started.

Additionally, some of these utilities cannot run at the same time as ALCS functions such as:

- Recoup
- State change (ZASYS command).

The application can allocate other bits in @SWITCH to allow the same checking for conflicting ALCS functions.

ALCS provides three installation-wide exit programs to help implement this type of control mechanism. They are:

AUT1
ALCS conditionally calls this program when starting any of the following functions:

State change (ZASYS command)
Recoup (ZRECP command)
Data dump (ZDATA command)
Data load (ZDATA command)
Restore (ZRSTR command).

AUT1 can test for and, if required, set bits in @SWITCH (or other indicators). If there is any conflicting application utility running, AUT1 can inhibit the ALCS function by setting a non-zero return code.

AUT2
ALCS conditionally calls this program when any of the functions listed above completes.

AUT2 can reset bits in @SWITCH (or other indicators) as required.

AVCT
ALCS conditionally uses this program as a table to associate a utility name with each bit in any non-zero return code set by AUT1. This allows ALCS to construct an error message which lists the utilities which conflict with the ALCS function.

ALCS conditionally enters AUT1 and AUT2 with the following conditions:

Register
Contents

R14
Contains one of the following values that indicate the ALCS function which is starting (on entry to AUT1), or ending (on entry to AUT2):

Value
ALCS function

0
State change (ZASYS command).

8
Recoup (ZRECP command).

16
Data dump (ZDATA command).

24
Data load (ZDATA command).

32
Restore (ZRSTR command).

AUT1 and AUT2 can modify registers R00, R01, R14 and R15. They must not modify any other registers or ECB fields.

AUT1 and AUT2 must return using BACKC.

ALCS tests the following return conditions from AUT1:

R15=0
ALCS can proceed with the function.

R15<>0
ALCS must not proceed with the function.

If program AVCT is loaded, then ALCS interprets the return code from AUT1 as a 32-bit string. It associates each bit with a utility program name. If a bit is on (1), then the corresponding utility is active.

Program AVCT must be coded as follows:

```
         BEGIN NAME=AVCT,...
*        Comments, SPACE, and EJECT instructions can be included
*        as required
         DC    AL1(L'label0)
label0   DC    C'first_utility_name'
         DC    AL1(L'label1)
label1   DC    C'second_utility_name'
         ...
         DC    AL1(L'label31)
label31  DC    C'last_utility_name'
         DC    AL1(0)             DELIMITER MUST FOLLOW LAST NAME
         FINIS ,
         END   ,
```

Where *first utility* is the utility corresponding to bit 0 of the AUT1 return code, *second utility* corresponds to bit 1, and so on up to *last utility*, which corresponds to bit 31. Specify a utility name consisting of a single blank (space) character as the name corresponding to any unused bit.

WAS Application protocol type 2 start-up initialization program - AWA1

ALCS conditionally enters this program to perform any custom initialization required before a protocol type 2 WAS Bridge becomes active.

ALCS conditionally enters AWA1 with the following conditions:

EBW004-EBW007
Address of the WAS Bridge CO0RE communications table entry. See “Communication resource data DSECT - CO0RE” on page 312 for details.

EBROUT
CRI of the WAS Bridge.

Programming considerations when using AWA1:

- AWA1 (or any program AWA1 calls) must not alter either EBW000-EBW016 or EBROUT.
- AWA1 (or any program AWA1 calls) may use all other work areas and registers. If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.
- AWA1 is conditionally entered by ENTRC, and control must eventually return by a BACKC.

An example of AWA1 including sample code is provided with the ALCS product in the installation-wide exit library.

WAS Application protocol type 2 shutdown initialization program - AWA2

ALCS conditionally enters this program to perform any custom initialization required after a protocol type 2 WAS Bridge becomes inactive.

ALCS conditionally enters AWA2 with the following conditions:

EBW004-EBW007
Address of the WAS Bridge CO0RE communications table entry. See "Communication resource data DSECT - CO0RE" on page 312 for details.

EBROUT
CRI of the WAS Bridge.

Programming considerations when using AWA2:

- AWA2 (or any program AWA2 calls) must not alter either EBW000-EBW016 or EBROUT.
- AWA2 (or any program AWA2 calls) may use all other work areas and registers. If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.
- AWA2 is conditionally entered by ENTRC, and control must eventually return by a BACKC.

An example of AWA2 including sample code is provided with the ALCS product in the installation-wide exit library.

Web Server custom content-type method exit program - AWM1

The *ALCS World Wide Web Server User Guide* describes this exit.

Web Server custom content-type table exit program - AWM2

The *ALCS World Wide Web Server User Guide* describes this exit.

Output message routing exit programs - AXA*n*

ALCS outputs centralized messages which are either responses to commands or internally generated messages (they can be error, attention, or information messages).

You can also define your own centralized messages using these installation-wide exit programs. Use the WTOPC macro to send centralized messages from application programs. For compatibility with previous releases of ALCS, message prefix APP is reserved as a subcomponent identifier for user-defined messages in the range DXC3000 to DXC3999.

Each message, together with its destination routing information, is passed to these exit programs.

User message routine - AXA1

ALCS conditionally enters AXA1 prior to sending the message to each predefined destination. This allows the application to indicate whether the message should be sent to this destination.

ALCS conditionally enters AXA1 with the following conditions:

Level D1
Output message block in CM1CM OMSG format.

Register
Contents

R04
The address of the message prefix data which is inserted at the start of the output message. The prefix data for an ALCS centralized message. is in the format:

Product Code
3 characters, 'DXC'

Message Number
4 characters

Severity Code
1 character

Blank
1 byte

Subcomponent Identifier
3 characters

Blank.
1 byte

ALCS System Identifier
1 character

The prefix data for a user centralized message is in the format:

Prefix
4 characters

Message Number
4 characters

Severity Code
1 character

Blank
1 byte

R06
The address of the message destination. This destination can be a CRN or one of the following:

EBROUT
Destination is originating terminal.

ASSDEV
Destination is the associated device for the originating terminal.

ROC
Destination is the RO CRAS.

AT*nnn*
Destination is alternate CRAS AT*nnn*.

AP*nnn*
Destination is alternate CRAS printer AP*nnn*.

R15
The address of an area containing:

Prefix
4 characters (or a 3-character subcomponent identifier followed by one blank character).

Message Number
4-byte binary number.

Program AXA1, and any program that AXA1 calls, must not modify the ECB work area.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

On return from AXA1, all registers (except R14 and R15) must be the same as at entry.

ALCS tests the following return conditions from AXA1:

R15=0
Process the message.

R15=4
Discard the message.

Program AXA1 must issue BACKC to return to the calling program.

User message routine - AXA2

ALCS conditionally enters AXA2 before sending the message to any of the predefined destinations. This allows the application to review the current destination list for the message and send the message to additional destinations.

ALCS enters AXA2 with the following conditions:

Level D1
Output message block in CM1CM OMSG format.

Register
Contents

R04
The address of the message prefix data which has been inserted at the start of the output message. The prefix data for an ALCS centralized message is in the format:

Product Code
3 characters, 'DXC'

Message Number
4 characters

Severity Code
1 character

Blank
1 byte

Subcomponent Identifier
3 characters

Blank
1 byte

ALCS System Identifier
1 character

The prefix data for a user centralized message is in the format:

Prefix
4 characters

Message Number
4 characters

Severity Code
1 character

Blank
1 byte

R06
The address of a message destination routing list. Each item in the list is 8 bytes. The list is terminated by hexadecimal FF in the first byte of the last item. The destination can be a CRN or a character string which describes the destination as:

EBROUT
The originating terminal.

ASSDEV
The associated device for the originating terminal.

ROC
RO CRAS.

AT*nnn*
Alternate CRAS AT*nnn*.

AP*nnn*
Alternate CRAS printer AP*nnn*.

R15
The address of an area containing:

Prefix
4 characters (or a 3-character subcomponent identifier followed by one blank character).

Message Number
4-byte binary number.

Program AXA2, and any program that AXA2 calls, must not modify the ECB work area.

If you require a data level, you must use DETAC and ATTAC macros (with TYPE=STACK) to save and restore the contents of any data levels that you use.

On return from AXA2, all registers (except R14 and R15) must be the same as at entry.

Program AXA2 must issue BACKC to return to the calling program to start transmission of the message to each destination in the routing list.

User message routine - AXA3

ALCS conditionally enters AXA3 (during the WTOPC macro service or from CXA0) to determine if user prefixes are defined for centralized-message handling.

AXA3 contains a message-prefix table where the user defines the message-routing program to process messages for a particular prefix and range of message numbers.

Code an entry in this table for each prefix which uses centralized messages and routings.

"User message routine - AXA4 through AXAn" on page 416 is an example of a program to provide messages and routings. If you need to create additional programs, use the names AXA5, AXA6, AXA7, and so on.

ALCS enters AXA3 with the following conditions:

Register
Contents

R14
Contains the message prefix (or X'00' followed by the subcomponent identifier)

R15
Contains the message number.

ALCS tests the following return conditions from AXA3:

R03
Points to the prefix entry if R06 is not zero

R06<>0
A prefix entry exists.

R06=0
No entry exists for the specified message prefix and message number.

R07
Unchanged

Other registers may be changed. The ECB work area must not be altered.

Figure 100 on page 416 shows an example which defines AXA4 and AXA5 as user-message routines for subcomponent identifier APP (message numbers 000 through 099) and prefix UGEN (message numbers 000 through 199).

```
*--------------------------------------------------------------------*
*        ENTRIES IN TABLE AXA3TAB HAVE THE FOLLOWING FORMAT:         *
*                                                                    *
*        BYTES  USAGE                                                *
*        -----  ---------------------------------------------        *
*        0 - 3  4 CHARACTER PREFIX (OR X'00' FOLLOWED BY             *
*               SUBCOMPONENT IDENTIFIER)                             *
*        4 - 7  RESERVED - SET TO ZERO                               *
*        8 -11  NAME OF USER MESSAGE PROGRAM                         *
*        12-15  MAXIMUM MESSAGE NUMBER HANDLED BY THIS               *
*               USER MESSAGE PROGRAM                                 *
*                                                                    *
*        ENTRIES WITH THE THE SAME PREFIX MUST                       *
*        BE IN ASCENDING ORDER OF MAXIMUM MESSAGE NUMBER             *
*--------------------------------------------------------------------*
         SPACE 1
AXA3TAB  EQU   *
         DC    A(C'APP',0,C'AXA4',99)  GENERAL APPLICATION MESSAGES
AXA3TABL EQU   *-AXA3TAB
         DC    A(C'UGEN',0,C'AXA5',199) GENERAL APPLICATION MESSAGES
         SPACE 1
AXA3END  EQU   *
```

Figure 100. Sample AXA3 program: Defines AXA4 and AXA5 as user-message routines

User message routine - AXA4 through AXA*n*

AXA4 through AXA*n* contain application messages and destination routing information for centralized message handling.

ALCS enters AXA4 with the following conditions:

R02
The address of the parameter list. The parameter list is in the following format:

Bytes 0 through 3
Prefix (or X'00' followed by subcomponent identifier).

Bytes 4 through 7
Fullword containing the message number.

Bytes 8 through 11
Fullword containing the value -1 (minus 1).

Bytes 12
Contiguous fullwords containing the addresses of any substitution text.

ALCS tests the following return conditions from AXA4:

R02
As at entry.

R03
The address of message or zeros.

R06
The address of destination routing list.

AXA4 uses the message number to index into a table containing addresses of application messages and destination routing lists. The following example shows a table which contains application messages and associated destination routing lists for the subcomponent APP and messages 0 through 5. These appear as message numbers DXC3000 through DXC3005 (DXC3000 through DXC3999 are reserved for user-defined messages).

```
*-------------------------------------------------------------------*
*        MESSAGE ADDRESS TABLE                                      *
*-------------------------------------------------------------------*
*        EACH ENTRY CONSISTS OF TWO FULLWORDS                       *
*                                                                   *
*        FULLWORD 1 -- THE ADDRESS OF A MESSAGE DEFINED BY DXCEMG   *
*        FULLWORD 2 -- THE ADDRESS OF A ROUTING DEFINED BY DXCEMR   *
*                                                                   *
*-------------------------------------------------------------------*
         SPACE 1
APPMSGB  DC    F'0'               BASE MESSAGE NUMBER
APPMSGS  DC    F'0'               MAXIMUM MESSAGE NUMBER
         SPACE 1
APPMS0   DC    A(APP000,APPRRO)   ENTRY FOR APPLICATION MESSAGE 0
APPMS1   DC    A(APP001,APPR01)   ENTRY FOR APPLICATION MESSAGE 1
APPMS2   DC    A(APP002,APPRT4)   ENTRY FOR APPLICATION MESSAGE 2
APPMS3   DC    A(APP003,APPROR)   ENTRY FOR APPLICATION MESSAGE 3
APPMS4   DC    A(APP004,APPRAS)   ENTRY FOR APPLICATION MESSAGE 4
APPMS5   DC    A(APP005,APPR99)   ENTRY FOR APPLICATION MESSAGE 5
APPMSGE  EQU   *-12
         SPACE 1
         ORG   APPMSGS            ORG BACK TO MAX MESSAGE NUMBER
         DC    A((APPMSGE-*)/8)   SET MAX MESSAGE NUMBER
         ORG   ,                  AND RESET LOCATION COUNTER
         EJECT ,
```

Figure 101. Sample AXA4 program: Defines the message-address table and routing

Using the DXCEMG macro

Use the ALCS DXCEMG macro to build each application message that is pointed to by this table. The format of the macro is:

```
label DXCEMG number,severity
            {,'text'|,(substitution_text)|,(#CAR),...}
```

---

label
A valid assembler label, maximum of 6 characters.

number
Message number, 3 or 4 decimal digits.

severity
Severity code, one of:

R
Reply. The operator must reply.

I
Information. Processing continues normally.

W
Attention. Processing continues but results may differ from those required.

E
Error. Processing continues but results will differ from those required.

S
Severe error. Processing cannot continue.

T
Termination. Processing is terminated.

text
Constant text, a quoted string.

(*substitution_text*)
Substitution text, either:

(*type*,*length*)

or

(*type*,*length*,{TRUNC|JUSTIFY}),

where:

type
One of the following:

CHAR
Character string.

DEC
Decimal number.

HEX
Hexadecimal number.

length
A self-defining term. For CHAR it is the length of the passed string. For DEC and HEX the passed data is always a fullword and *length* is the length of its character representation. For example (DEC,3) converts F'3' to C'003' and converts F'1234' to C'234'.

TRUNC
Truncate the result, remove:

- Trailing blanks, when used with *type* CHAR
- Leading zeros, when used with *type* DEC or *type* HEX. For example, (DEC,3,TRUNC) converts F'3' to '3'.

JUSTIFY
Replace the high-order zeros in the result with the immediately preceding character: JUSTIFY applies only to *type* DEC

(#CAR)
Use the (#CAR) parameter to specify a new line.

```
*-------------------------------------------------------------------*
*        MESSAGES DEFINITIONS                                       *
*-------------------------------------------------------------------*
*-------------------------------------------------------------------*
*        MESSAGES IN RANGE  0 - 5   SAMPLE MESSAGES                 *
*-------------------------------------------------------------------*
         SPACE 1
APP000   DXCEMG 3000,E
APP001   DXCEMG 3001,I,'This is an example'
APP002   DXCEMG 3002,S,'Program ',(CHAR,4),' has ended with errors'
APP003   DXCEMG 3003,W,'There are ',(DEC,3,TRUNC),' shopping days left'
APP004   DXCEMG 3004,I,'The price is: *',(DEC,8,JUSTIFY),' US dollars'
APP005   DXCEMG 3005,T,'Application terminated - cannot read database'
         EJECT ,
```

Figure 102. Sample AXA4 program: Defines the contents for each message

```
          WTOPC PREFIX=APP,NUM=0,TIME=NO
          WTOPC PREFIX=APP,NUM=1,TIME=NO
          WTOPC PREFIX=APP,NUM=2,SUB=(CHARA,NAME),TIME=NO
          WTOPC PREFIX=APP,NUM=3,SUB=(DEC,123),TIME=NO
          WTOPC PREFIX=APP,NUM=4,SUB=(DECA,DEC45),TIME=NO
          WTOPC PREFIX=APP,NUM=5,TIME=NO

 DEC45    DC    F'45'
 NAME     DC    C'ABCD'
```

Figure 103. Sample input: WTOPC statements

```
DXC3000E APP M

DXC3001I APP M  This is an example

DXC3002S APP M  Program ABCD has ended with errors

DXC3003W APP M  There are 123 shopping days left

DXC3004I APP M  The price is: *******45 US dollars

DXC3005T APP M  Application terminated - cannot read database
```

Figure 104. Sample output: Formatted output screens (DXC3000 - DXC3005)

Using the DXCEMR macro

Use the ALCS DXCEMR macro to build a destination routing list for each application message. The format of the DXCEMR macro is:

```
label DXCEMR routing[,routing, ...]
```

Where:

label
A valid assembler label, maximum of 6 characters.

routing
Destination routing of the message, send the message to:

ROC
The RO CRAS.

AT*nnn*
Alternate CRAS AT*nnn*.

AP*nnn*
Alternate CRAS printer AP*nnn*.

crn
The resource with CRN *crn*.

EBROUT
The resource whose CRI is in EBROUT.

ASSDEV
The associated resource.

The following example shows how to specify message routing.

```
*-------------------------------------------------------------------*
*        ROUTING LIST DEFINITIONS                                    *
*-------------------------------------------------------------------*
         SPACE 1
APPRRO   DXCEMR ROC               ROUTE TO RO CRAS
APPR01   DXCEMR ROC,EBROUT        ROUTE TO RO CRAS AND ORIGINATOR
APPRT4   DXCEMR AT4               ROUTE TO AT4 CRAS
APPROR   DXCEMR EBROUT            ROUTE TO ORIGINATOR
APPRAS   DXCEMR ASSDEV            ROUTE TO ASSOCIATED DEVICE
APPR99   DXCEMR ROC,AP1,NET001    ROUTE TO ROC, AP1 AND CRN NET001
```

Figure 105. Sample DXCEMR macro: message routing

Global load control programs - GOA*n*

The ALCS application global load routine (CGL1, the ECB-controlled monitor program) conditionally reads these programs; they define the application global loading. “Updating global load control programs” on page 450 explains how to code them.

Time available supervisor program - TIA1

ALCS creates an entry to execute the application program TIA1. This entry is known as the time available supervisor (TAS). When program TIA1 completes processing it must issue the TASBC macro. See the description of the TASBC monitor-request macro in the *ALCS Application Programming Reference - Assembler*. All ECB fields and registers that application programs can use have unpredictable contents upon entry to program TIA1.

Application date/time update program - UGU1

ALCS conditionally creates an entry to execute the application program UGU1 whenever the ALCS clock reaches a minute boundary. By convention, the application program UGU1 maintains the application clocks and calendar. Note that, unlike ALCS/VSE and TPF, ALCS calls program UGU1 in idle state as well as CRAS, message switching, and normal states. Conventionally, applications maintain their own clocks to synchronize with the ALCS clocks. This is not a requirement of ALCS. However, if the application clocks are independent, program UGU1 must decide when the application clocks cross a date boundary.

ALCS enters UGU1 with the following conditions:

EBW000
One byte, set as follows:

X'00'
Minute boundary with no date change.

X'01'
Minute boundary with Greenwich Mean Time (GMT) date change.

X'10'
Minute boundary with local mean time date change.

X'11'
Minute boundary with GMT date change and local mean time date change.

All other ECB fields and registers that application programs can use have unpredictable contents. When called from the ALCS monitor, program UGU1 must issue an EXITC monitor-request macro to terminate processing.

AAA address compute/find utility program - WGR1

If your application uses the Agent Assembly Area (AAA) record, you may have allocated a AAA record on your ALCS database for each communication resource. Your application may use program WGR1 to calculate the file address of the AAA fixed file record. The WGR1 program can obtain the fixed file address using the ordinal number allocated to the communication resource that the AAA record is associated with.

Each AAA fixed file record may contain references to data records that are held in long-term pool, therefore the ALCS Recoup utility must access the fixed AAA records for the purpose of chain chasing the pool records that they contain references to. When the ALCS Recoup utility calculates the file address of each AAA fixed file record, it unconditionally activates the application program WGR1. Recoup activates WGR1 if you have coded a Recoup Descriptor program that contains an INDEX macro with the parameters:

```
INDEX REF=(AAA,ref-address)
```

The *ref-address* must contain the displacement to a field containing the communication CRI address for the AAA record. Chapter 9, “Long-term pool space recovery - Recoup,” on page 480 describes the INDEX macro.

The Recoup utility enters WGR1 with the following conditions:

Register R01
Contains the address of the field holding the three-byte CRI.

EBCM01
An indicator byte with bit 0 set on and the remaining bits set to zero.

WGR1 can use any of the general registers, but must issue a BACKC macro to return to Recoup with the following fields initialized:

CE1FM1
Contains the fixed file address of the AAA record.

EBW040
Contains return conditions (bit 0 must be on if WGR1 is unable to calculate the file address).

Message switching header analysis program - XHP1

Application program XHP1 is the IPARS - ALCS V2 message switching output message header analysis program. ALCS creates an entry to XHP1 as part of the SENDC T monitor-request macro service routine.

ALCS enters XHP1 with the following conditions:

Level D0
A size L1 block containing the output message (with a possible standard forward chain). The XM5XM macro (part of IPARS - ALCS V2) describes the format of this message. The 2-character record ID of this message is TM.

ALCS programs that applications can call

Some ALCS ECB-controlled monitor programs perform functions that application programs require. Application programs can use these ALCS programs.

Scan data base program - CAP1

Application programs can issue ENTRC to CAP1 in order to retrieve (at high speed) all the records that satisfy certain user-supplied selection criteria.

Each time a record is found that satisfies the user's criteria, the scan function enters a user-supplied program with a copy of the record in a block attached to the ECB.

The scan function enters a user-supplied program before starting the scan and again after the end of the scan. You provide the names of these programs as part of the CAP1 entry conditions.

You can use this function in any system state, but not when the following facilities are in use:

- ZDATA LOAD
- ZDATA DUMP
- ZRSTR

You can cancel a scan with a ZDATA CANCEL command.

Notes:

1. Entries that cause a call to CAP1 must originate at Prime CRAS.
2. ALCS is responsible for global-area serialization when user programs update the work area.
3. The user programs are responsible for record hold if the user programs update any records that the data base scan function identifies.

Applications enter CAP1 with the following conditions:

Level D0
A size L2 block containing the entry parameters in 20-byte or 16-byte format. You should should use the 20-byte format in preference. The earlier 16-byte format continues to be supported.

- Layout of 20-byte header:

  Byte 0
  Entry code (see below).

  Byte 1
  Return code (see below).

  Bytes 2-3
  Number of items (between 1 and 40).

  Bytes 4-7
  Name of user program to call before the start of the scan and after the end of the scan.

  Byte 8
  Format of entry parameters. The only value allowed is 1.

  Byte 9
  Bit 0 when set, presents the allocatable pool file address of any record selected. All other bits should be zero.

  Bytes 10-19
  Reserved - should be zero.

- Layout of 20-byte items which immediately follow the 20-byte header:

  Bytes 0-1
  Selection mask that optionally restricts which records are presented to the user program (byte 0 is used only when record class = 10):

  CAP1 Selection mask

  Byte 0 bit 0
  When set, selects only those records which are flagged in-use.

  Byte 0 bit 1
  When set, selects system fixed file records.

  Byte 0 bit 2
  When set, selects application fixed file records.

  Byte 0 bit 3
  When set, selects short-term pool records.

  Byte 0 bit 4
  When set, selects Type 1 long-term pool records.

  Byte 0 bit 5
  When set, selects application allocatable pool records.

  Byte 0 bit 6
  When set, selects system allocatable pool records.

  Byte 0 bit 7
  Reserved.

Byte 1 bit 0
Reserved.

Byte 1 bit 1
When set, selects all fixed file and ignores record class/type.

Byte 1 bit 2
When set, selects only those records with the required record ID.

Byte 1 bits 3-7
Reserved.

Bytes 2-3
Record ID (used only when selection mask byte 1 bit 2 is set).

Bytes 4-5
Record class:

1
Scan system fixed file records, where bytes 6-7 identify the record size.

2
Scan application fixed file records, where bytes 6-7 identify the fixed file type.

3
Scan short-term pool records, where bytes 6-7 identify the pool interval.

4
Scan Type 1 long-term pool records, where bytes 6-7 identify the pool interval.

10
Scan records using selection mask in byte 0, where bytes 6-7 identify the record size.

11
Scan records on a general file, where bytes 6-7 identify the general file.

Bytes 6-7
Record type, one of:

Record size code (L1, L2, and so on).
Application fixed file record type value.
Pool interval number (L1STPOOL, L1LTPOOL, and so on).
Pool interval number (L1, L2, and so on).
General file number.

Bytes 8-11
Start ordinal number (use 00000000 when selecting all ordinal numbers).

Bytes 12-15
End ordinal number (use FFFFFFFF when selecting all ordinal numbers).

Bytes 16-19
Name of user program to call when the scan finds a record that satisfies the selection criteria.

- Layout of 16-byte header:

Byte 0
Entry code (see below).

Byte 1
Return code (see below).

Bytes 2-3
Number of items (between 1 and 40).

Bytes 4-7
Name of user program to call before the start of the scan and after the end of the scan.

Byte 8
Selection mask that optionally restricts which records are presented to the user program:

CAP1 Selection mask

Byte 8 bit 0
When set, selects only those records that are flagged in-use.

Byte 8 bits 1-7
Reserved.

Bytes 9-15
Reserved.

- Layout of 16-byte items, which immediately follow the 16-byte header:

Bytes 0-1
Record size code (L1, L2, and so on).

Bytes 2-3
Record ID.

Bytes 4-7
Name of user program to call when the scan finds a record which satisfies the selection criteria.

Bytes 8-15
Reserved.

- CAP1 Entry code:

0
Start request - with automatic restart (ALCS will restart the scan automatically after an outage).

4
Start request - with no automatic restart (ALCS will not restart the scan automatically after an outage but will await a restart request).

8
Restart request.

Applications can test the following return conditions from CAP1:

- CAP1 Return code after start request:

0
Started successfully.

4
Another utility is already in use.

8
Entry parameters are invalid.

12
One or more user programs not loaded.

16
Error return from user start-of-scan program.

20
ECB not authorized (Prime CRAS authority required).

- CAP1 Return code after restart request:

0
Restarted successfully.

4
Restart request invalid.

12
One or more user programs not loaded.

16
Error return from user start-of-scan program.

20
ECB not authorized (Prime CRAS authority required).

User program entered at start of scan and end of scan

This user supplied program can:

- Set up or clear the work area in the application global area
- Open or close a general sequential file for output
- and so on.

CAP1 enters (by ENTRC) this program with the following conditions:

EBW096
Entry code

EBW096=0
Start of scan (normal).

EBW096=4
Start of scan (restart).

EBW096=8
End of scan.

EBW096=12
Scan cancelled by operator.

EBW096=16
Scan cancelled by system error.

EBW100-103
Address of 480 byte work area in the application global area. ALCS keypoints this work area if the return code indicates that the work area is updated.

The program can use general purpose registers R00 through R07 and R14 and R15 without restoring them. The ECB work areas and data levels (except for the return code in EBW097 and EBW098) must be unchanged when this program returns to the calling program

This program must return to CAP1 with the following conditions:

- Return code 1 (EBW097) and 2 (EBW098) after entry at start of scan:

  EBW097=0
  Proceed with scan.

  EBW097=4
  Do not proceed with scan.

  EBW098=0
  The application did not update the work area

  EBW098=1
  The application did update the work area.
- Return code 1 (EBW097) and 2 (EBW098) after entry at end of scan:

  EBW097
  Ignored

  EBW098=0
  The application did not update the work area

  EBW098=1
  The application did update the work area.

  The program must return to the calling program to enable ALCS to complete end-of-scan housekeeping.

User program entered when record found which satisfies user criteria

This user supplied program can be used for such purposes as:

Writing the record on a sequential file

Producing a report
Updating the record and writing it back to the database
and so on.

CAP1 enters (by ENTRC) this program with the following conditions:

Level D2
A block containing a copy of the record found.

EBW100-103
Address of 480 byte work area in the application global area. ALCS keypoints this work area if the return code indicates that the work area is updated. This is the same area to which the start-of-scan/end-of-scan program refers.

The program can use general purpose registers R00 through R07 and R14 and R15 without restoring them. The ECB work areas and data levels (except for the return code in EBW097 and EBW098) must be unchanged when this program returns to the calling program.

This program must return with the following conditions:

EBW097=0
Retrieve another record which satisfies the criteria.

EBW097=4
Terminate the scan of records of this size.

EBW097=8
Terminate the scan of records (all sizes).

EBW098=0
The application did not update the work area.

EBW098=1
The application did update the work area.

Start BSC lines - CBSC

An application program can issue ENTRC to CBSC to open and start BSC communication lines.

There are no special entry conditions for CBSC.

On return from CBSC, general register contents are unpredictable, and the 4-byte field starting at EBX044 is corrupted.

Stop BSC lines - CBSD

An application program can issue ENTRC to CBSD to stop and close BSC communication lines.

There are no special entry conditions for CBSD.

On return from CBSD, general register contents are unpredictable.

Character conversion routine - CCNV

Refer to the *ALCS Application Programming Reference - Assembler* for entry requirements and usage notes.

ALCS command processor - CFMS

For applications that process messages with the first character Z, ALCS routes all messages (except those with first 5 characters ZROUT or ZLOGF, followed by a space) to the application. See the FMSG parameter description in “COMDEF parameters for a local application” on page 123. These applications must issue ENTDC to CFMS to process ALCS commands.

Applications enter CFMS with the following conditions:

Level D0
A size L1 block containing the message. The CM1CM macro describes the format of the message.

All other levels must be available. The ECB work areas (EBW*nnn* and EBX*nnn*) must contain zeros.

Open and start SLC links - CML8 and CMLA

An application program can issue ENTRC to CML8 to open SLC communication links, and ENTRC to CMLA to start them.

There are no special entry conditions for CML8 or CMLA.

On return from CML8 and CMLA, general register contents are unpredictable, and the ECB work areas are corrupted.

Stop and close SLC links - CMLD and CMLG

An application program can issue ENTRC to CMLD to stop SLC communication links, and ENTRC to CMLC to close them.

There are no special entry conditions for CMLD or CMLG.

On return from CMLD and CMLG, general register contents are unpredictable, and the ECB work areas are corrupted.

SLC AML handler - CMRA

For SLC Type B input messages, the application must make the message secure and inform ALCS, so that ALCS can send a clear message label (CML) according to SLC procedure. To do this, the application must create an entry to CMRA.

Applications enter CMRA with the following conditions:

EBW000
Must contain the SLC link number.

EBW001
Contains:

Bit 0
Possible duplicate message (PDM) indicator.

Bits 1 through 3
Message label.

Bits 4 through 6
SLC link channel number.

Bit 7
Type B indicator.

Command confirmation program - CONF

Sometimes application input messages or commands have effects that are difficult or impossible to reverse. Command confirmation allows the end user to reconsider before proceeding. It prompts the end user to reenter the command (or input message) within 30 seconds. If the end user does not do this, the command (or input message) is ignored.

You can either use the default prompt (`Are you sure`) or you can provide your own text that describes possible unintended or unexpected effects of the command (or input message).

You request command confirmation by calling (ENTRC) program CONF.

Entry conditions for CONF are as follows:

Register
Contents

Level D0
A size L1 block containing the input user command in IMSG format. The CM1CM macro describes the format of the message.

R01
Must contain zero or address of replacement message text for the output response.

When a user command requires confirmation, CONF outputs a two line response:

```
Are you sure?
Reenter within 30 seconds to confirm
```

If this is a suitable response, CONF should be activated with a value of zero in Register R01. If replacement text is used for the first line of the response the address of the replacement text should be in register R01 when CONF is activated. The format of the replacement text is a one byte field containing the length of the text (in binary) immediately followed by the message text itself (in EBCDIC).

R14
Must contain a four-character token which identifies the action that the user command is requesting. This token could be the four characters that follow the primary action code in the user command.

Do not use the tokens PURG, CSEQ, INIT, ATIM, STAT, OCTM, PDAR, RECP, CTHR, or PERF because they are reserved for ALCS use.

R15
Identifies the action to be taken by CONF as follows:

0
User command requires confirmation

4
User command does not require confirmation

Before activating CONF the calling program should:

Validate the parameters on the user command or input message (otherwise syntax errors will not be reported until after confirmation has completed).
Not issue any macros that request terminal hold (AAA hold), resource hold, record hold, or sequential file assign.
Use the same CONF entry conditions for the *initial* input and the *confirmation* input.

The return conditions are that the registers (except R15), ECB work area, and data levels are unchanged. On some conditions CONF will not return control to the calling program.

R15=0
Only one of the user commands is returned to the calling program. CONF returns the entry containing the *initial* command after it has successfully processed the *confirmation* command. The entry containing the *confirmation* command is not returned.

Neither of the user commands will be returned if the *confirmation* command is:

Not entered within 30 seconds
Not identical to the *initial* command
Not entered from the same display as the *initial* command

R15=4
CONF returns the entry to the calling program.

Application start-up program - CPL3

An application program can issue ENTRC to CPL3 to start an ALCS application.

Applications enter CPL3 with the following conditions:

EBW000-EBW007
CRN of application to start.

EBROUT
CRI of originating terminal, or zeros if none.

Storage levels D0, D2, and DE must be available.

On return from CPL3, general register contents are unpredictable, ECB work areas are as at entry.

Application shut-down program - CPL4

An application program can issue ENTRC to CPL4 to stop an ALCS application.

Applications enter CPL4 with the following conditions:

EBW000-EBW007
CRN of application to stop.

EBROUT
CRI of originating terminal, or zeros if none.

Storage levels D0, D2, and DE must be available.

On return from CPL4, general register contents are unpredictable, ECB work areas are as at entry.

Resource control record queue checking - CQS7

CQS7 reads a resource control record (RCR). From this record, it takes information about message queues for a specified CRI, and passes the information back to the calling program.

Applications enter CQS7 with the following conditions:

Register
Contents

R14
The address of storage available to receive return information.

R15
CRI of resource to be checked in lower 3 bytes.

R15
Type of information to be supplied in top byte:

01
Printer queues (R14 must address 64 bytes). There are 16 priority queues. For each queue, CQS7 returns the count of the number of messages queued for a particular priority. Each count is 4 bytes long.

02
Printer queues extended (R14 must address 192 bytes). There are 16 priority queues. For each queue, CQS7 returns the following:

- 4-byte count of the number of messages queued for a particular priority.
- 4-byte TOD stamp of the first message queued
- 4-byte TOD stamp of the last message queued.

03
Printer queues extended (R14 must address 320 bytes). There are 16 priority queues. For each queue, CQS7 returns the following:

- 4-byte count of the number of messages queued for a particular priority
- 4-byte TOD stamp of the first message queued
- 4-byte file address of the first message queued
- 4-byte TOD stamp of the last message queued

- 4-byte file address of the last message queued.

Return conditions for CQS7 are as follows:

R00 - R14
Same as upon entry.

R15
CQS7 completion code, a non-zero completion code indicates an error.

0
Normal completion.

4
The CRI is not valid.

8
The storage address provided by the calling program is not valid.

12
The information-type code provided by the calling program is not valid.

16
Error occurred (RONIC/FACE/FIND type). CQS7 also supplies a dump.

EBW*nnn* area
Same as upon entry.

EBX*nnn* area
Same as upon entry.

Send a message using SMTP - CSMS

An application program can issue ENTRC to CSMS to send a message using the Simple Message Transfer Protocol (SMTP).

The 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.

Applications enter CSMS with the following conditions:

Level D0
A storage block containing the message which CSMS sends. There is no pool file address assigned to the prime or only a single block. When an email consists of more than one block, the blocks are chained using the standard forward chain field. The format of this storage block is defined by the DXCSMTM macro. The maximum size of this block is LX (the largest size defined in your system).

Your program must clear the storage block to binary zeros (if you obtain a new storage block, it is already cleared to binary zeros) before initializing the following fields:

SMTPTEXT
The text of the message. The message text must include a message header and a message body (more information is provided below on the format of the message text). The message header must be in EBCDIC. Your program can prepare the message body using any character set (the default is EBCDIC). If the body of your message is not in EBCDIC then your program must also initialize the fields SMTPCSET and SMTPCNAM.

Your program can prepare the message body using plain text or HTML (the default is plain text). HTML (Hypertext Markup Language) is a formatting language used to create documents for use on the World Wide Web. Many e-mail reader programs can interpret HTML codes which make URLs into clickable links and change the font, size, or colour of text. If the message body is to be treated as an HTML document then your program must also initialize the field SMTPHTML.

SMTPCSET
Bit indicating that the message body is not in EBCDIC. Set this bit on using:

```
OI SMTPCSET,L'SMTPCSET
```

Note: When SMTPCSET is set, then SMTPLENG must be less then 75% of the largest block size defined in your system.

SMTPCNAM
40-byte field containing the name of the character set used for the message body.

Note: Official character set names are registered with the Internet Assigned Numbers Authority (IANA) at URL <http://www.iana.org>.

SMTPHTML
Bit indicating that the message body is an HTML document. Set this bit on using:

```
OI SMTPHTML,L'SMTPHTML
```

SMTPORIG
128-byte field containing the e-mail address of the sender. By default, ALCS constructs an e-mail address for the sender using the terminal address in the ECB field EBROUT. Problems in the transport or delivery of the original message are reported to this e-mail address. If the recipient used `REPLY TO SENDER` when sending their reply, the destination of the reply depends on other fields included in the original e-mail message:

- If there is a "Reply-To:" field, the reply will be sent to the address(es) indicated in that field.
- If there is a "From:" field but no "Reply-To" field, the reply will be sent to the address(es) indicated in the "From" field.
- If there are no "From:" or "Reply-To:" fields, the reply will be sent to the address of the sender.

SMTPHFCH
Forward chain (a file address) to the next part of the message or zeroes if none. The record id of the records that contain the message must be X'AB10'.

SMTPLENG
Fullword field containing the length (number of bytes in binary) of the message text in this block.

The prime block is not assigned a pool file address. Pool file addresses are also not assigned to single blocks used when only one block is needed.

When email consists of more than one block, the blocks are chained using the standard forward chain field (SMTPHFCH).

On return from CSMS, the storage block on level D0 is released. Other ECB levels, and the ECB work areas, are unmodified. General register 15 (RDB) contains one of the following binary return code values:

Code
Meaning

0
Message transmitted OK

12
Message not transmitted - message format incorrect

All other registers are unmodified.

Although the format of the message text that you pass to CSMS is described in the Internet publication "Standard for the Format of ARPA Internet Text Messages", RFC 822 (and other Internet memos) there are some differences. The following describes those differences:

- The header of the message you pass to CSMS must be in EBCDIC, not ASCII. (But the characters you use must be in the US ASCII character set.)
- If the body of the message you pass to CSMS is not in EBCDIC, you must specify the name of the actual character set in the entry conditions for CSMS.
- Each line of EBCDIC text must end with #CAR, not with #CR, #LF.
- If the body of the message you pass to CSMS is to be treated as an HTML document instead of plain text, you must specify this in the entry conditions for CSMS.
- You must include at least one recipient e-mail address in your message headers.

- Do not include these headers in your message:

```
Sender:
Date:
Message-ID:
```

(CSMS automatically adds these headers to your message.)

ALCS WAS utility - CWAD

An application program can issue an ENTRC to CWAD to obtain status information about the WAS connection.

On return from CWAD

R15
Return code

0
WAS connection enabled

4
WAS connection not enabled

EBW000
Status of the WAS connection (when R15 =0)

X'80'
Connected

X'40'
Disconnected

X'20'
Disconnecting

EBW004-EBW007
Unpredictable

EBW008-EBW011
Total number of ALCS WAS subtasks used by OLA callable services (when R15 = 0)

EBW012-EBW015
Active number of ALCS WAS subtasks used by OLA callable services (when R15 = 0)

EBW016-EBW103
Same as upon entry.

EBX000-EBX113
Same as upon entry.

Registers R00-R14
Same as upon entry.

Levels D0-DF
Same as upon entry.

Check availability of pool - CVEP and CVEQ

An application program can issue ENTRC to CVEP or CVEQ to check the availability of long-term or short-term pool.

Applications enter CVEP with the following conditions:

Register
Contents

R14
Pool interval number (L1STPOOL, L1LTPOOL, ...)

Applications enter CVEQ with the following conditions:

Register
Contents

R14
Record ID in low-order 2 bytes
Record ID qualifier (0 to 9) in high-order byte.

Return conditions for CVEP and CVEQ are as follows:

Register
Contents

R14
Number of available records in the pool, or zero if an error condition has occurred.

R15
Current pool dispense rate per second, or completion code if register R14 is zero.

The completion code is one of:

4
GFS is not active.

8
The pool does not exist.

12
The pool is not active.

16
The pool is depleted.

20
The pool interval number is not valid.

24
The record ID qualifier is not valid.

28
The record ID is not valid for LT or ST pool.

32
The record ID is ambiguous.

R00
Unpredictable

Other registers are the same as on entry.
ECB work areas are the same as on entry.

Numbered output message processor - CXA0 and CXA1

Application programs can issue ENTRC to CXA0 to generate an output message that conforms to the ALCS numbered message format. See *ALCS Messages and Codes* for a description of this format. Installation-wide exits can create an entry to CXA1 to do the same.

CXA0 and CXA1 call exits AXA4, AXA1, and AXA2 (see "Output message routing exit programs - AXAn" on page 412). Use these exits to define output messages and destination routing information according to your installation's requirements.

Applications enter CXA0 with the following conditions:

Register
Contents

R01
The address of parameter list. The parameter list is in the following format:

Bytes 0-3
Characters APP followed by a blank character.

Bytes 4-7
Fullword containing the message number (this must correspond to a message defined in the installation-wide ECB-controlled exit program AXA4).

Bytes 8-11
Fullword containing the value -1 (minus 1).

Bytes 12-...
Contiguous fullwords containing the addresses of any substitution text.

Applications enter CXA1 with the following conditions:

EBW000
Start of parameter list. The parameter list is in the same format as described above.

CXA0 restores all registers, ECB error indicators, ECB work areas, condition code, and program mask before returning to the calling program.

Set WTTY line status to started - CXLE

An application program can issue ENTRC to CXLE to set the status of WTTY communication links to started.

There are no special entry conditions for CXLE.

On return from CXLE, general register contents are unpredictable, and the ECB work areas are corrupted.

Set WTTY line status to stopped - CXLF

An application program can issue ENTRC to CXLF to set the status of WTTY communication links to stopped.

There are no special entry conditions for CXLF.

On return from CXLF, general register contents are unpredictable, and the ECB work areas are corrupted.

Fixed file record file address compute - FACE and FACS

Refer to the *ALCS Application Programming Reference - Assembler* for entry requirements and usage notes.

Customization for Recoup

This section describes the customization that you can perform at your installation to prepare for using Recoup. Recoup is described fully in Chapter 9, “Long-term pool space recovery - Recoup,” on page 480.

Recoup descriptor program directory - BZ00

You must have a Recoup descriptor program directory BZ00. A sample is provided with IPARS - ALCS V2.

BZ00 must contain a list of all the Recoup descriptor programs for the application. Each item in the list is a 4-byte program name. Recoup descriptor programs must be separate programs, they cannot be transfer vectors. ALCS issues an unconditional FINPC macro for BZ00. Chapter 9, “Long-term pool space recovery - Recoup,” on page 480 for more details of Recoup. Figure 106 on page 435 shows an example of how to code BZ00.

```
         BEGIN NAME=BZ00,VERSION=00,AMODE=31
         SPACE 1
*=====================================================================*
*        TABLE OF DESCRIPTOR PROGRAMS FOR USE WITH ALCS RECOUP        *
*=====================================================================*
*        THIS PROGRAM CONTAINS A LIST OF ALL DESCRIPTOR PROGRAMS      *
*                                                                     *
*        THE FIRST  SLOT NUMBER MUST HAVE LABEL BZ00TAB1              *
*        THE LAST   SLOT NUMBER MUST HAVE LABEL BZ00TEND              *
*                                                                     *
*        THERE ARE NO SPECIAL RESTRICTIONS ON PROGRAM NAMES           *
*        BUT IT IS RECOMMENDED THAT DESCRIPTOR PROGRAM NAMES ARE      *
*        CHOSEN IN A WAY THAT DISTINGUISHES THEM FROM OTHER PROGRAMS  *
*---------------------------------------------------------------------*
         SPACE 1
         DC    A((BZ00TEND-BZ00TAB1)/4) MAX SLOT NUMBER
BZ00TAB1 DC    CL4'BZ01'           SLOT 0  TTYH/TTYF
         DC    CL4'BZ02'           SLOT 1  UAT(AAA)
         DC    CL4'BZ03'           SLOT 2  PNID/PNIG
         DC    CL4'BZ04'           SLOT 3  XLGI/XLGO/XTCB/XLMA/XTIQ
         DC    CL4'BZ05'           SLOT 4  CGI
         DC    CL4'BZ06'           SLOT 5  PNIOC/PNIOG
         DC    CL4'BZ07'           SLOT 6  BING/BIND/BINM
         DC    CL4'BZ08'           SLOT 7  XCPC/XRPT
BZ00TEND DC    CL4'BZ09'           SLOT 8  UAF
         SPACE 1
         FINIS ,
         SPACE 1
         END   ,
```

Figure 106. Example of a Recoup descriptor program directory

The first fullword in program BZ00, after the BEGIN statement is the number of descriptor programs minus 1. Figure 106 on page 435 assumes that there are two or more descriptor programs. If there are no descriptor programs then the first fullword in BZ00 must be F'-1'. This can be followed immediately by the FINIS macro. If there is one descriptor program then the first fullword in BZ00 must be F'0'. This must be followed immediately by the descriptor program name. This can be followed immediately by the FINIS macro.

Error handling in Recoup

In addition to the six installation-wide exits to user-written subroutines situated in descriptor programs, there are six exit programs which are (conditionally) entered when certain error conditions are detected by Recoup in descriptor programs or in the database. These installation-wide exit programs are described in the following sections.

For each error condition, attention messages are sent to the RO CRAS and the ALCS diagnostic file. The fields in these messages have the following meanings:

aaaa
Descriptor program name.

bbbbbbbb
Prime group name.

cc
Record ID.

dddddddd
File address.

eeeeeeee
Record ordinal (in hexadecimal).

ffffffff
Displacement from start of record to the reference (in hexadecimal).

gggggg
CRI.

Missing Recoup descriptor - ARC1

If Recoup detects a missing descriptor program (that is, a program that is listed in BZ00 but is not loaded), it normally issues the information message:

```
hh.mm.ss RECP  Missing descriptor program -- Recoup continues --
Missing descriptor program aaaa
```

If the user has loaded program ARC1, then Recoup issues an ENTRC to ARC1 instead of issuing the attention message.

ALCS enters ARC1 with the following conditions:

Register
Contents

R02
The address of the name of the missing program.

ARC1 can return control to Recoup, in which case Recoup continues at the next descriptor program. If ARC1 does not return control, Recoup cannot continue, and you must cancel it by ALCS command.

FACE error in prime group - ARC2

If Recoup detects a FACE error in a prime group when METHOD=SEQ, it normally issues the attention message:

```
hh.mm.ss RECP  FACE error in prime group -- Recoup continues --
Prime group bbbbbbbb first ordinal not valid
```

or:

```
hh.mm.ss RECP  FACE error in prime group -- Recoup continues --
Prime group bbbbbbbb last ordinal not valid
```

If the user has loaded program ARC2, then Recoup issues an ENTRC to ARC2 instead of issuing the attention message.

ALCS enters ARC2 with the following conditions:

EBX028-EBX031
The address of the prime group.

ARC2 can return control to Recoup, in which case Recoup continues at the next descriptor program. If ARC2 does not return control, Recoup cannot continue, and you must cancel it by ALCS command.

Undefined record ID - ARC3

If Recoup fails to find the required record ID when METHOD=STOR, it normally issues the attention message:

```
hh.mm.ss RECP  RECID not found in prime group -- Recoup continues --
Prime group bbbbbbbb RECID cc
```

If the user has loaded program ARC3, then Recoup issues an ENTRC to ARC3 instead of issuing the attention message.

ALCS enters ARC3 with the following conditions:

EBX028-EBX031
The address of the prime group.

ARC3 can return control to Recoup, in which case Recoup continues at the next descriptor program. If ARC3 does not return control, Recoup cannot continue, and you must cancel it by ALCS command.

File address error in prime group - ARC4

If Recoup finds an invalid file address in the global area when METHOD=GL, it normally issues the attention message:

```
hh.mm.ss RECP  Invalid file address in prime group -- Recoup continues --
 Prime group bbbbbbbb File addr dddddddd
```

If the user has loaded program ARC4 then Recoup issues an ENTRC to ARC4 instead of issuing the attention message.

ALCS enters ARC4 with the following conditions:

EBX028-EBX031
The address of the prime group.

ARC4 can return control to Recoup, in which case Recoup continues at the next descriptor program. If ARC4 does not return control, Recoup cannot continue, and you must cancel it by ALCS command.

Record ordinal number not valid - ARC5

If Recoup detects a FACE error when REF=FACE, it normally issues the attention message:

```
hh.mm.ss RECP  Invalid FACE ordinal in record -- Recoup continues --
Refer-from record dddddddd Ordinal eeeeeeee
Disp ffffffff
```

If the user has loaded program ARC5, then Recoup issues an ENTRC to ARC5 instead of issuing the attention message.

ALCS enters ARC5 with the following conditions:

Level D0
The refer-from record

ARC5 must return control to Recoup, and Recoup continues with the next reference.

CRI not valid - ARC6

If Recoup detects an invalid CRI when REF=AAA, it normally issues the attention message:

```
hh.mm.ss RECP  Invalid CRI in record -- Recoup continues --
Refer-from record dddddddd CRI gggggg
Disp ffffffff
```

If the user has loaded program ARC6, Recoup issues an ENTRC to ARC6 instead of issuing the attention message.

ALCS enters ARC6 with the following conditions:

Level D0
The refer-from record

ARC6 must return control to Recoup, and Recoup continues with the next reference.

Using the database analysis file in Recoup

In addition to the installation-wide exit programs described above, there is an installation-wide exit program that enables the user to select the information that is put on the database analysis file:

Adding information to the database analysis record - ARC7

If the operator has asked for database analysis information to be collected by Recoup, then Recoup places an item in the database analysis file for each pool file record chain-chased. The default length and layout of each item are described by macro BS0AI.

If the user has loaded program ARC7, then Recoup issues an ENTRC to ARC7. instead of using the default item contents and item length.

ALCS enters ARC7 with the following conditions:

Register
Contents

R14
The address of a work area of size L1.

EBW004
The 4-byte file address of refer-to record.

EBW012
The 4-byte file address of refer-from record.

ARC7 must set up the work area as follows before issuing BACKC to return to Recoup:

Byte
Contents

0 and 1
Item length

2 through 7
Reserved

8 up to 255
Item contents

The item length can vary from one run of Recoup to another, but for a given Recoup run, the item length must be fixed and not more than 256 bytes.

Registers R14 through R07 can be used in installation-wide exit programs ARC1 through ARC7. Register save areas CE1URA through CE1UR7 must not be used in these programs.

Using non-standard database layout parameters in ALCS Recoup

ALCS normally requires the parameters describing the database to be provided in Recoup descriptor programs. Sometimes the parameters are available in a different machine readable form (such as TPFDF DBDEF macros). In this situation customers can undertake a manual effort to code ALCS descriptor programs from the parameters held in the non-ALCS format. However, this can be time consuming and requires the ALCS descriptor programs to be reviewed whenever the parameters held in the non-ALCS format are changed. To enable ALCS Recoup to use the information held in the non-standard form without coding descriptor programs, three installation-wide ECB controlled exits are provided in ALCS. These are called ARD0, ARD1, and ARD2.

Customers wishing to chain chase records using database layout parameters held in non-ALCS format without coding descriptor programs must code programs ARD0, ARD1, and (possibly) ARD2 as described below.

Customers with an appropriate version of TPFDF will find that programs ARD0 and ARD1 are suitable for using with TPFDF, and a sample ARD2 program is provided with TPFDF.

Return size of customer descriptor area - ARD0

Before starting chain-chase, ALCS will call program ARD0 (if it is loaded). Program ARD0 must return to ALCS the size of the storage area needed to contain the GROUP and INDEX macro expansions that ARD1 will create. This storage area is called the customer descriptor area (CDA). The entry to ARD0 is made only once for each Recoup run.

ARD0 must return the size of storage area required in bytes in R14.

The size required for each GROUP macro expansion is fixed and is given by GR1LEN in the GROUP macro. The size required for each INDEX macro expansion is fixed and is given by IN1LEN in the INDEX macro. Currently the sizes are the same and are both 72 bytes.

After the return from ARD0 and before entering ARD1, ALCS obtains the storage requested by ARD0. The storage obtained starts on a page boundary and the length is rounded up to finish on a page boundary. The storage obtained is in 'entry' key and is not page fixed. The address and length of the storage obtained is stored by ALCS in the common data area at fields CP0CDA and CP0CDAL respectively. These fields are both fullword fields. The maximum size of the customer descriptor area supported is 500000 bytes.

Build customer descriptor area - ARD1

If ARD0 is loaded and returns a valid storage estimate, ALCS enters ARD1 to allow ARD1 to create GROUP and INDEX macro expansion in the customer descriptor area. This entry is made repeatedly until ARD1 indicates that no further calls are needed.

The entry conditions to ARD1 are as follows:

Register
Contents

R00=0
The first call to ARD1

R00=4
A subsequent call to ARD1

R14
The next available byte in the customer descriptor area

The return conditions from ARD1 are as follows:

Register
Contents

R00=0
More calls to ARD1 are needed

R00=4
No more calls to ARD1 are needed

R14
The next available byte in the customer descriptor area

The difference between R14 at entry to ARD1 and R14 at return from ARD1 is the amount of storage used by ARD1 to create GROUP and INDEX macro expansions during the call to ARD1.

The GROUP and INDEX macro expansions created in the CDA are the same as produced by ALCS descriptor programs with the following restrictions:

- INDEX macros may not refer to GROUP macros by name. INDEX macros can only refer to GROUP macros by record ID. To tell ALCS that the INDEX macro is referring to the GROUP macro by record ID,the INDEX macro coded in a descriptor program contains the operand GROUP=(ID). This results in the field IN1GRP in the INDEX macro containing X'FFFF'. To allow a GROUP macro to be accessed by record ID, the GROUP macro coded in a descriptor program contains the operand METHOD=ID. This results in the field GR1SW1 in the GROUP macro expansion being set to GR1SW1D.

- GROUP macros may not refer to INDEX macros by name. To tell ALCS that the GROUP macro is not referring to an INDEX macro by name, the GROUP macro operand contains the operand INDEX=. This results in the field GR1IND in the GROUP macro expansion containing F'0'.

Note that the customer descriptor area does not contain any user subroutines. These continue to reside in the user programs (for example TPFDF programs).

Exits for customer routines - ARD2

ALCS Recoup allows six user exits for user written subroutines (see "User-written subroutines and programs" on page 491). These six user exits are supported as follows, when chain chasing parameters are in non-ALCS format:

ENTRTN
If ARD2 is loaded and the field GR1ENT is non-zero in a prime GROUP macro expansion in the CDA, then Recoup will enter ARD2 before processing the prime group.

Entry condition - R00=0

EXTRTN
If ARD2 is loaded and the field GR1EXT is non-zero in a prime GROUP macro expansion in the CDA, then Recoup will enter ARD2 after processing the prime group.

Entry condition - R00=4

RRECRTN
If ARD2 is loaded and the field GR1RTN is non-zero in a GROUP macro expansion in the CDA, then Recoup will enter ARD2 before reading the record.

Entry conditions - R00=8

EBW000/1
Expected record ID

EBW002
Expected RCC

EBW004/7
File address of the record to be read

EBW028/31
Address (in CDA) of GROUP macros containing record

Return condition:

Register
Content

R06=0
Do not read the record (end of group)

R06=4
Read the record

PRECRTN
If ARD2 is loaded and the field GR1PTN is non-zero in a GROUP macro expansion in the CDA, then Recoup will enter ARD2 after reading the record (and before processing any INDEX macros that may be associated with the record.)

Entry conditions R00=12:

EBW028/31
Address (in CDA) of GROUP macro expansion containing record

The record being processed is on ECB level 0

Return conditions:

Register
Content

R06=0
Ignore record (proceed to next in chain)

R06=4
Process the record normally

R06=8
Ignore record (end of group)

INDXRTN
If ARD2 is loaded and the field IN1RTN is non-zero in an INDEX macro expansion in the CDA, then Recoup will enter ARD2 before processing the index.

Entry conditions R00=16:

R05
Address (in CDA) of the INDEX macro expansion

EBW028/31
Address (in CDA) of the GROUP macro containing record

The record being processed is on ECB level 0

Return conditions:

Register
Content

R06=0
Ignore this INDEX macro

R06=4
Process this INDEX macro

ITEMRTN
If ARD2 is loaded and the field IN1ITM is non-zero in an INDEX macro expansion in the CDA, then Recoup will enter ARD2 before processing the item.

Entry conditions R00=20:

R02
The address of the item in the record

R05
Address (in CDA) of the INDEX macro expansion

EBW028/31
Address (in CDA) of the GROUP macro expansion containing record

Record being processed is on ECB level 0

Return conditions:

Register
Content

R06=0
Ignore this item

R06=4
Process the item

For all the above subroutines the following conditions apply:

- In each case entry is by ENTRC ARD2.
- In each case return is by BACKC.
- Registers R14 through R07 (except R06) may be used and do not need to be restored on return to ALCS.

- The ECB work area EBX064 through EBX103 is not referred to by ALCS and is available for use in the user exit. Other parts of the ECB work area (if used) must be saved and restored before return to ALCS.
- Data and storage levels DE and DF are not used by ALCS and are available for use in the user exit.
- The user area in the Recoup keypoint BK0USR (5 double words) is not referred to by ALCS and is available for use in the user exit.
- When in ARD2, the address of the Recoup keypoint can be obtained by:

```
BK0RP REG=Ryy             DSECT for Recoup keypoint
GLOBZ REGR=Rxx,FLD=@BRKPC Global directory base
L     Ryy,@BRKPC          Recoup keypoint base
```

- When in ARD2, the address of the prime GR0UP macro expansion in the CDA can be obtained as follows when the Recoup keypoint address is known:

```
GROUP REG=Rzz             DSECT for group
L     Rzz,CP0CDA          CDA address
A     Rzz,BK0PRI          Add displacement to prime group
```

Note that if there is no requirement to call user exits while processing GR0UP and INDEX macros in the CDA then program ARD2 need not be loaded.

Macro customization

This section describes the customization which you can apply to some of the ALCS macros.

Use System Modification Program Extended (SMP/E), or a functionally equivalent program, to change ALCS. To avoid conflicts between IBM service and user modifications, do not modify ALCS, except for the following components:

DXCSER
The global access serialization macro. See “ALCS DXCSER user macro” on page 443 for further information.

DXCZCUSR
ALCS generation macro to validate user data. See “ALCS DXCZCUSR user macro” on page 444 for further information.

RTCEQ
For TPF compatibility.

DXCUSAL
The ALCS program transfer vector macro. See “ALCS DXCUSAL user macro” on page 445 for further information.

For each modification, construct a user modification type of SYSMOD, called a USERMOD. IBM recommends that you use SUPERC (or a functionally equivalent program) to check your USERMODs against the ALCS source code before using them. Then use the SMP/E RECEIVE and APPLY commands to apply this USERMOD to ALCS. You must never ACCEPT USERMODs.

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*.

ALCS Operation and Maintenance describes how to apply USERMODS and SYSMODS to ALCS.

Application macros that the ALCS BEGIN macro issues

The ALCS BEGIN macro generates macroinstructions for two macros that are not part of ALCS; they are part of the application. They are:

- SYSEQ

- XMSEQ

By convention, applications use these macros to define assembler symbols that ECB-controlled programs commonly use. If the application does not use these macros, you must provide dummy versions.

The DXCURID macro defines your fixed-file record type symbols. Do not define them in the SYSEQ or XMSEQ macros.

ALCS DXCSER user macro

The DXCSER macro processes any global synchronization tokens that you define for your installation. These tokens are operands of the BEGIN macro SHR= and XCL= parameters. The ALCS BEGIN macro calls DXCSER.

ALCS does not require you to use, or even define, any global synchronization tokens. But if you do, you must modify the ALCS DXCSER macro to process your tokens. Note that the DXCSER provided with ALCS is only an example. It is unlikely that the tokens it defines will be useful for your installation.

Before modifying DXCSER, you must decide which tokens your installation needs. Once you have chosen the appropriate tokens, you must update the DXCSER macrodefinition.

For each of your serialization tokens, you must set one element of the subscripted local set symbol &NAME to the token (padded with spaces to 8 characters), and the corresponding element of the subscripted local set symbol &SETS to the corresponding bit string as follows:

```
&NAME(n)  SETC  'token  '
&SETS(n)  SETC  'bit_string'
```

You must also set the local arithmetic set symbol &NAMES to the highest subscript that you use.

For example, you might code the following sequence of instructions in DXCSER:

```
&NAME(001) SETC 'GLOBAL0 '         SUBSET 0
&SETS(001) SETC '10000000000000000000000000000000'
&NAME(002) SETC 'GLOBAL1 '         SUBSET 1
&SETS(002) SETC '01000000000000000000000000000000'
&NAME(003) SETC 'GLOBAL2 '         SUBSET 2
&SETS(003) SETC '00100000000000000000000000000000'
&NAME(004) SETC 'GLOBAL3 '         SUBSET 3
&SETS(004) SETC '00010000000000000000000000000000'
&NAME(005) SETC 'GLOBAL4 '         SUBSET 4
&SETS(005) SETC '00001000000000000000000000000000'
&NAME(006) SETC 'GLOBAL5 '         SUBSET 5
&SETS(006) SETC '00000100000000000000000000000000'
&NAME(007) SETC 'GLOBAL24'         SUBSET 24
&SETS(007) SETC '00101000000000000000000000000000'
&NAMES   SETA  7
```

This sequence of instructions uses bit 0 for the token GLOBAL0, bit 1 for the token GLOBAL1, and so on. The token GLOBAL24 is equivalent to both of GLOBAL2. and GLOBAL4.

Note that you must **replace** the existing SETCs for &NAME and &SETS, and the SETA for &NAMES, in the DXCSER macro supplied with ALCS.

See *ALCS Application Programming Guide* for a full explanation of global serialization.

ALCS DXCURID user macro

You must create a DXCURID macro that contains the fixed-file record type symbols that your applications require.

The DXCURID macro provides the association between the fixed-file record type symbols and the fixed-file record type numbers (explained in “Fixed-file records” on page 179). ALCS uses it to generate assembler EQU instructions that define the fixed-file record type symbols. Application programs use the DXCURID macro to obtain the fixed-file record type number for a fixed-file record type symbol. The ALCS generation macros use DXCURID to check the validity of fixed-file record type symbols.

For each application fixed-file record type, DXCURID must set one element of the subscripted global character set symbol &DXCFN to the fixed-file record type symbol. DXCURID must also set the corresponding element of the subscripted global arithmetic set symbol &DXCFV to the fixed-file type number as follows:

```
&DXCFN(n) SETC 'symbol'
&DXCFV(n) SETA value
```

DXCURID must also set the global arithmetic set symbol &DXCFM to the highest used subscript. DXCURID can use subscripts 2 through 4094; there is no requirement for the symbol value to be the same as the subscript value. There can be gaps in the use of subscripts. DXCURID ignores any subscripted &DXCFN which is a null string.

The following restrictions apply to the definition of fixed-file record type symbols and numbers in DXCURID:

- Symbols must not exceed 8 characters. Your installation can impose additional standards or restrictions for selecting fixed-file record type symbols.
- DXCURID must not define the following fixed-file record type symbols:
  - #KPTRI or #FACTEND.
  - Any symbol starting with the characters DXC

  These symbols are reserved for use by IBM.
- DXCURID must not use fixed-file record type number 0. This value is reserved for use by IBM (for #KPTRI.).
- DXCURID must define fixed-file record type symbol #CPRCR. You can use any non-zero fixed-file record type number.

Attention

Do not change the fixed-file record type numbers on a production ALCS system (although you can add new numbers). If fixed-file record type symbols are added or deleted (or if fixed-file record type numbers change on a test system), you must reassemble all ECB-controlled application programs which reference the affected symbols.

ALCS DXCZCUSR user macro

The DXCZCUSR macro is called by the COMDEF communication generation macro, during the communication generation Stage 1. Use DXCZCUSR to validate any user data specified on the COMDEF generation macro for a communication resource.

The entry conditions to DXCZCUSR are as follows:

Positional parameter &LDTYPE
Communication resource type specified on the COMDEF or COMDFLT macro.

Positional parameter &USERDAT
User data specified on the COMDEF or COMDFLT macro.

Keyword parameters &USER1 to &USER16
User data specified on the COMDEF and COMDFLT macro.

The return conditions from DXCZCUSR are as follows:

Global character set symbol &UDATA
Validated user data. This must consist of a string of not more than 254 hexadecimal digits, which is used to complete an assembler DC instruction. For example:

```
&UDATA SETC '0123456789ABCDEF'
```

results in the assembler instruction

```
       DC   X'0123456789ABCDEF'
```

Global arithmetic set symbol &USERR
Return code, one of the following:

0 through 3
User data is valid.

4 through 7
Discard the user data.

8 or over
Discard the user data and do not build the communication generation Stage 2.

ALCS DXCUSAL user macro

The DXCUSAL macro provides information on transfer vectors that are used in application programs (transfer vectors define additional entry points in application programs). For application programs that have multiple entry points, their transfer vectors can be defined to ALCS in one of two ways:

- By including a TRANV macro for each individual transfer vector, immediately after the BEGIN macro at the start of the application program. See *ALCS Application Programming Reference - Assembler* for a description of the TRANV macro.
- By including a SALPR statement in the DXCUSAL macro for each application program that contains transfer vectors.

IBM recommends that you use the TRANV macro for defining transfer vectors in an ALCS system. When an application is migrated from TPF to ALCS you may prefer to migrate the TPF transfer vector information to the DXCUSAL macro because the TRANV macro is not supported by TPF (and therefore will not have been used in the application programs).

The DXCUSAL macro is comprised of one or more SALPR statements, with each SALPR statement providing the association between the name of the transfer vector and the name of the program that it resides in. The format of the SALPR statement is:

```
SALPR (PROG,TV1,TV2,TV3,.........)
```

where

PROG defines the name of the application program.
TV1 defines the name of the first transfer vector in the program.
TV2 defines the name of the second transfer vector in the program.
TV3 defines the name of the third transfer vector in the program, and so on.

The following is an example of two SALPR statements in the DXCUSAL macro. These statements define transfer vectors for application programs UIL1 and XARA. Program UIL1 has three transfer vectors and program XARA has six transfer vectors:

```
SALPR  (UIL1,UIL2,UIL3,UIL4)
SALPR  (XARA,XARB,XARC,XARD,XARE,XARF,XARH)
```

When a program is assembled, the DXCUSAL macro is called (via the BEGIN macro). If the DXCUSAL macro contains a SALPR statement for that program, the transfer vectors required by that program are generated.

Your application macro library must contain a DXCUSAL macro. If your application programs do not have transfer vectors, or you have defined all your transfer vectors using the TRANV macro, then you must place a dummy DXCUSAL macro in the application macro library (it may contain only comment lines).

Defining fields in CE1USA

ALCS provides an area in the ECB called CE1USA, where you can define your own system-wide ECB fields.

Do not define fields in CE1USA if you can define them in one of the ECB work areas. Define only genuinely system-wide fields in CE1USA.

If you are porting application programs from other installations (ALCS or TPF), they can depend on fields defined in CE1USA at the installation that developed them. If they do, you will probably need to define the fields in your CE1USA. There are 2248 bytes available in CE1USA.

Defining fields in CE1USA - assembler

To define fields in CE1USA for assembler language programs, **do not** modify the ALCS EB0EB macro. Instead, write your own macro to generate instructions, for example:

```
EB0EB    DSECT ,                    CONTINUE THE EB0EB DSECT
         ORG   CE1USA               OVERLAY THE CE1USA AREA
  (define your fields here)
         ORG   ,                    RESET THE EB0EB LOCATION COUNTER
&SYSECT  CSECT ,                    CONTINUE THE CSECT
```

A convenient way to ensure that all programs use the same format for CE1USA is to generate these instructions in your SYSEQ macro. You can either generate the instructions directly in SYSEQ, or write a separate macro which you call from SYSEQ.

Naming fields in CE1USA

By convention, you should choose names of the form CU1*ccc* (*ccc* is any 3 characters) for fields within CE1USA.

Aligning fields in CE1USA

If your fields require special alignment (for example, on a halfword, fullword, or doubleword boundary), it is better to force the required alignment than to rely on an alignment that happens to result from fields defined earlier in CE1USA. For example, to force doubleword alignment of a field called CU1DBL, code:

```
CU1DBL   DC    D'0'           (D-type is aligned automatically)
```

or:

```
         DC    0D'0'          FORCE ALIGNMENT
CU1DBL   DC    X'...          (X-type is not automatically aligned)
```

or:

```
CU1DBL   DC    0D'0',X'...    (X-type is not automatically aligned)
```

Forcing alignment makes it easier to copy your field definitions into another installation's CE1USA when porting your application.

Defining fields in CE1USA - C language

To define fields in CE1USA for C language application programs, **do not** modify the <c$eb0eb.h> header file. Instead, write your own header file as follows:

```
/* C$MYUSA.H - define the ECB ce1usa fields for my application */

typedef struct
```

```
{
   short int    cu1fld1;
   int          cu1fld2;
   char         cu1fld3;
} my_usa;

#define CU1FLD1     (((my_usa *)(ecbptr()->ce1usa))->cu1fld1)
#define CU1FLD2     (((my_usa *)(ecbptr()->ce1usa))->cu1fld2)
#define CU1FLD3     (((my_usa *)(ecbptr()->ce1usa))->cu1fld3)
```

Use this in your application program as follows:

```
#include <c$eb0eb.h>
#include "c$myusa.h"


  CU1FLD1 = 0x1234;
  CU1FLD2 = 0x12345678;
  CU1FLD3 = 'A';
```

C language application programs

All ALCS programs require you to include the file <tpfeq.h>. Some functions require additional header files such as <tpfapi.h>.

<tpfeq.h> will also include <c$mi0mi>. <tpfapi.h> will also include <c$syseq.h> and <c$mi0mi.h>.

<c$syseq> and <c$mi0mi> are not part of ALCS - they are part of the application. They are equivalent to the SYSEQ and MI0MI assembler macros. You must define your own version of <c$syseq> (and <c$mi0mi> if you intend to use <tpfapi.h>) or provide dummy versions if your applications do not require them.

Note: The IBM C/C++ compiler provides a DSECT conversion utility which generates a structure to map an assembler DSECT. This utility is described in the *C/C++ User's Guide* and it could be used to convert MI0MI.

Reserved symbols and values

IBM reserves the following fixed-file record type symbols:

#KPTRI
All symbols that start with #CP.

IBM reserves the following fixed-file record type numbers:

00 (#KPTRI)

IBM uses the following record identifiers:

All IDs that start with X'A'
'CK'
'BK'
'BL'
'BS'
'BX'
'FP'.

Customization for TPFDF

If your installation uses IBM TPFDF, then you must make the customization changes described in the following publications:

- *TPF Database Facility: Program Directory*.
- *TPF Database Facility: Installation and Maintenance Manual*.

Including records in the application global area

When there is no alternative to using the global area, add a record to the global area, as follows:

1. Choose which global area you will use for the record. There are three global areas, 1, 2 and 3. If your installation is using global area protection (refer to the GLBLPROT parameter on the ALCS generation SCTGEN macro), global areas 1 and 3 are in protected storage, therefore the global area that you choose for your record must be determined by your requirement to place the record in either protected or unprotected storage.
2. Update the corresponding global area directory DSECT macro to include a directory slot for the record. See "Updating global area directory DSECTs" on page 448.
3. If your installation is using C language programs that access the global areas, you must recreate <c$globz.h>. "Building a global tag header file c$globz.h for C language programs" on page 52 describes how to do this.
4. Step "2" on page 448 can change the value of symbols (that is, the displacement to labels in the DSECT) that existing application programs use. If it does, reassemble/recompile and link-edit the affected programs.
5. If the new record is directly addressable, update the corresponding directory macro (GL*n*BA) to include the new record in the GL*n*BA DSECT. See "Updating global area directory DSECTs" on page 448.
6. To load the new record into the global area, update the global load control program for the directory. "Updating global load control programs" on page 450 explains how to do this.
7. Update the GLBLSZE parameter in the ALCS generation SCTGEN macro.

   Refer to the SCTGEN macro for a description of the GLBLSZE parameter and how to determine the size of the global area.
8. Reassemble and re-link program AGT1.

Updating global area directory DSECTs

About this task

ALCS supports 16 global area directories, directory 0 through directory 15 (hexadecimal F). There is a DSECT macro for each directory. The DSECT macros are GL0BA for directory 0, GL1BA for directory 1, and so on up to GLFBA. Directory 15 (GLFBA) is reserved for IBM use.

Note: GL*n*BA macros (except GLFBA) are **not** shipped as part of ALCS. However, examples are shipped with IPARS.

Each directory contains either 8-byte directory entries (starting on a doubleword boundary) or 4-byte directory entries (starting on a fullword boundary). You can select some directories with 8-byte directory entries (slots) and some with 4-byte directory entries.

If the directory entries are 8 bytes long, the first fullword in the entry contains the main storage address of the record, and the second fullword contains the file address of the record.

If the directory entries are 4 bytes long, the entry contains only the main storage address. File addresses are kept separate from the directory.

Application programs refer to the label on the first fullword when they refer to the corresponding global area record; that is, they load the storage address of the record from that fullword.

The GL*n*BA DSECT macros (GL0BA, GL1BA, and so on) generate the DSECTs that describe the global area directories. They also generate tables of **tags**. The ZDCOR and ZACOR commands use these tags to refer to the global area directories. The ALCS diagnostic file processor also uses these tags; it prints them in dumps of the global area directory.

ALCS Operation and Maintenance shows an example of this. These tags can be useful, but ALCS does not require a tag for every directory slot.

To add a slot to a global area directory:

Procedure

1. Decide which directory to use. Note that directory 15 (GLFBA) is reserved for IBM use.
2. Update the corresponding directory DSECT macro (GL*n*BA) to add the new directory slot.

   You need only perform the following steps if you are adding tags:
3. Update the GL*n*BA macro to generate a tag for the new directory slot.
4. Assemble, or update and assemble the application global area tag program. See "Global area tag exit programs - AGT0, AGT1, and so on" on page 367.
5. To include the new tag in system error dump listings, reassemble the CSECT DXCDTPGL, and relink-edit DXCDTPGT.

Results

Note that application programs refer to labels in the directories. A new directory slot can change the displacements to other existing directory slots. If it does, then you must reassemble (or recompile) and link-edit application programs that refer to the corresponding labels. To avoid this, add new slots after existing slots in the directory, and include spare slots where applicable to allow for expansion.

If directly addressable records follow a directory, a new directory slot can change the displacement to the directly addressable records. Avoid this by allocating spare slots in any directory that has directly addressable records following it.

C language application programs

C language programs refer to directory slots and directly addressable fields in the global area using tags defined in <c$globz.h>.

Note: These tags are different to the tags used with ZAC0R and ZDC0R.

If a new slot or field is added to the global area then <c$globz.h> must be recreated to allow the new slot or field to be used by C language programs. If the displacement of any existing slot or field is changed, all C language programs which refer to this slot or field must be recompiled, using the recreated <c$globz.h>, and relinked. See also "Building a global tag header file c$globz.h for C language programs" on page 52.

Adding directly addressable global records

About this task

Application programs issue GL0BZ to load a base register for a directory. Because the directory is less than 4KB, the base register can address storage beyond the end of the directory. If the total size of the directory and one or more records that follow it does not exceed 4KB, the application program can use the directory base register to address the directory **and** the records following the directory. These records are called **directly addressable records**.

ALCS allows one or more directly addressable global area records to follow each of the directories 0 through 15 (directory 15 is reserved for IBM use).

To make a record directly addressable:

Procedure

1. Extend the DSECT describing the directory so that it also describes the record or records that follow the directory. To do this, extend the directory by including the macro that describes the directly addressable record. By convention, the macros for the directly addressable records that follow GL*n*BA are GL*n*BB, GL*n*BC, and so on.
2. Add the directly addressable record to the appropriate global load control program. Be sure to load the record in the correct place; the directly addressable records must immediately follow the corresponding directories.

Updating global load control programs

ALCS uses information in 16 programs (GOA0 through GOAE, and CGAF) to load the application global area. GOA0 through GOAE and CGAF are called **global load control programs**. Use GOA0 through GOAE for user applications. GL*n*BA describes directory *n*. The instructions for loading are in program GOA*n* (*n* is 0 through E).

Global load control programs contain one item for each record that ALCS is to load, and an extra item (called the **logical global control item**) for each logical global. G01G0 LOAD macroinstructions generate these items. Each G01G0 LOAD macroinstruction generates one item (except when the NUMBER= parameter is used to allow one G01G0 macro to create several items). Each item loads one global record (except that logical global control items do not load any records).

Programs GOA0 through GOAE and CGAF comprise a series of G01G0 macroinstructions. Use one G01G0 macroinstruction for each record to be loaded into the global area (except where the NUMBER= parameter is used to allow one G01G0 macro to create several items). ALCS loads the records in the same order as the G01G0 macroinstructions. Ensure that directly addressed records are loaded first (immediately following the directory). The G01G0 macroinstructions for all records of a logical global must be contiguous, and must be preceded by the G01G0 macroinstruction for the logical global control item. The four formats of G01G0 are as follows:

At the start of a GOA*n* program:

```
G01G0 START
      ,SLOTS=number_of_slots
      [,SLOT1=first_slot]
      [,SLOTLEN={8|4}]
```

For each record to load:

```
G01G0 [LOAD]
      ,SLOT={number|label|NONE}
      ,ORDINAL=ordinal
      ,{LENGTH=length|DWORD=double_words}
      {,ID=id|IDSYM=id_symbol}
      [,AREA={1|2|3}]
      [,CHECK={NO|YES}]
      [,KEYPT={NO|YES}]
      [,TYPE={#GLOBL|record_type}]
      [,HDSTRIP={0|number}]
      [,IDCHECK={NOSUPPRESS|SUPPRESS}]
      [,LOGGING={NO|YES}]
      [,NUMBER={1,n}]
      [,PROTECT={NO|YES}]
```

For each logical global control item:

```
G01G0 [LOAD]
      ,SLOT={number|label}
      ,LOGGLOB=YES
      ,LOGGCNT=number
```

At the end of a GOA*n* program:

```
G01G0 END
```

Where:

START
The first G01G0 macroinstruction in the GOA*n* program. This instruction does not load a global record.

SLOTS=*number_of_slots*
Number of slots in the directory. An expression or a decimal number.

SLOT1=*first_slot*
Assembler label (symbol) for the first directory slot (from the GL*n*BA DSECT).

SLOTLEN={8|4}
Length of global area directory entries (in bytes).

LOAD
Load a global record. This is the default.

SLOT={*number*|*label*|NONE}
Slot in GL*n*BA, one of:

number
Slot number, a decimal number in the range 1 through 255.

label
Assembler label (symbol) for the slot (from the GL*n*BA DSECT). Use a symbol only if the G01G0 START macroinstruction specifies the SLOT1 parameter.

NONE
Use SLOT=NONE if a directory slot is not needed. SLOT=NONE may be used with KEYPT=YES to enable a logical global to use only 1 directory slot. If this is required then provide the slot name in the G01G0 macro for the logical global control item and use SLOT=NONE for all other G01G0 macros for the logical global.

Use contiguous slots for the records that make up a logical global.

ORDINAL=*ordinal*
Record ordinal number of the record. An expression or a decimal number.

{LENGTH=*length*|(DWORD=*double_words*}

LENGTH=*length*
Number of bytes to load (excluding header-stripped bytes); a decimal value. The minimum value is 8, and is rounded up to a multiple of 8.

DWORD=*double_words*}
Number of doublewords to load (excluding header-stripped bytes); a decimal value. The minimum value is 1.

{ID=*id*|IDSYM=*id_symbol*}
Record identifier:

ID=*id*
Two alphanumeric or 4 hexadecimal characters.

IDSYM=*id_symbol*
Symbol for the record identifier. An assembler EQU instruction in the data macro for the record must assign a value to this symbol.

AREA={1|2|3}
Global area 1, 2, or 3. Specify 3 if only 31-bit mode application programs access the record.

Note: When a logical global overlaps from one directory into another, the logical global must be in area 2 or area 3. Directly addressable records must be in area 1.

CHECK={NO|YES}
If YES then ALCS conditionally enters installation-wide exit USRGUPD to allow the user to check the contents of the global record before it is written out to file. See "Validate global record keypointing exit - USRGUPD" on page 267 for more details of this exit.

KEYPT={NO|YES}
The record is non-keypointable (NO) or keypointable (YES). Specify KEYPT=YES for all records of a logical global.

TYPE={#GLOBL|*record_type*}
Fixed file record type symbol of the record.

HDSTRIP={0|*number*}
Header stripping requirements:

0
No header stripping (default).

number
Number of bytes to be stripped, an even decimal number in the range 2 through 30.

Specify HDSTRIP for all records of a logical global except the first record.

IDCHECK={NOSUPPRESS|SUPPRESS}
Action that ALCS takes when it detects an ID check while loading this record:

NOSUPPRESS
Send an error message.

SUPPRESS
Load the record into the global area and insert the expected record ID at the beginning. Clear the rest to hexadecimal zeros.

Note: Use SUPPRESS for records where the only field that requires initialization is the record ID.

LOGGING={NO|YES}
If NO then bypass the logging of this keypointable global record.

NUMBER={1,*n*}
A G01G0 macro with **NUMBER**=*n* is equivalent to *n* G01G0 macros all with the same parameter except that the ordinals increase by one with each succeeding macro.

The NUMBER= operand can only be used if SLOT=NONE.

Example

```
G01G0 LOAD,SLOT=NONE,                                  -
      ORDINAL=88,                                      -
      LENGTH=1056,                                     -
      ID=XY,                                           -
      AREA=1,                                          -
      KEYPT=YES,                                       -
      NUMBER=50
```

This is equivalent to 50 G01G0 macros with ordinals 88, 88+1, 88+2, through to 88+49.

PROTECT={NO|YES}
If YES then this global record starts at a page boundary and ALCS inserts a page before the record with a different storage protection key. If you use the NUMBER= parameter to load multiple global records then a page is inserted before the first copy only.

LOGGLOB=YES
The control item for a logical global control record.

LOGGCNT=*number*
Number of records in the logical global.

END
The last G01G0 macroinstruction in the GOA*n* program. This macroinstruction does not load a global record.

Example

```
G01G0 LOAD,SLOT=NONE,                                  -
      ORDINAL=88,                                      -
      LENGTH=1056,                                     -
      ID=XY,                                           -
      AREA=1,                                          -
```

```
        KEYPT=YES,                                              -
        NUMBER=50
```

This causes ALCS to load 1056 bytes of the record type #GL0BL, record ordinal 19, into global area 1. The record identifier must be X'FD'. ALCS builds the directory item in slot 46. The record is keypointable.

Figure 107 on page 453 shows an example of a global load program.

```
         BEGIN NAME=GOA0,VERSION=00,AMODE=31
         SPACE 1
         GLOBZ REGR=NO             GENERATE GLOBAL AREA DSECT
         SPACE 1
         GO1GO START,                                              -
               SLOT1=@GLOBA,                       FIRST SLOT      -
               SLOTS=(@GLOBB-@GLOBA)/L'@GBLDE      NUMBER OF SLOTS
         SPACE 1
         GO1GO LOAD,SLOT=@GBLBC,   GLOBB                           -
               ORDINAL=4,                                          -
               LENGTH=1055,                                        -
               ID=GL,                                              -
               AREA=1,                                             -
               KEYPT=NO
         SPACE 1
         GO1GO LOAD,SLOT=@GBLCC,   GLOBC                           -
               ORDINAL=5,                                          -
               LENGTH=480,                                         -
               ID=GL,                                              -
               AREA=1,                                             -
               KEYPT=YES
         SPACE 1
⋮
         GO1GO END
         SPACE 1
         FINIS ,
         END   ,
```

Figure 107. Example of the use of a global load program

Diagnostic file processor global tags table - DXCDTPGT

You can provide the ALCS diagnostic file processor with a table of tags for the contents of the application global area directories. To do this, you must create a load module with the entry point DXCDTPGT. This must also be the name (or an alias) of the load module.

When you run the ALCS diagnostic file processor, it attempts to load (MVS LOAD macro) DXCDTPGT. If the load works, the ALCS diagnostic file processor uses DXCDTPGT to set up tags against the global area directories. If the load does not work, it prints the global area directories without tags.

The appropriate GL*n*BA DSECT macro can provide a parameter for generation of the table. Figure 108 on page 454 shows an example.

```
         ENTRY DXCDTPGT            START OF TABLE
         EJECT ,
*------------------------------------------------------------------------*
*        GLOBAL AREA FORMAT TABLE                                        *
*------------------------------------------------------------------------*
*        SIXTEEN ENTRIES, ONE FOR EACH GLOBAL DIRECTORY                  *
*        EACH ENTRY IS TWO FULLWORDS                                     *
*                                                                        *
*        A(START-OF-TAGS,LENGTH-OF-TAGS)                                 *
*------------------------------------------------------------------------*
         SPACE 1
DXCDTPGT DC    A(BSSGP0LL,BSSGP0A)
         DC    A(BSSGP1LL,BSSGP1A)
         .
         .
         .
         .
         DC    A(BSSGPFLL,BSSGPFA)
         EJECT ,
*------------------------------------------------------------------------*
*        GLOBAL DIRECTORY LABELS                                         *
*------------------------------------------------------------------------*
         SPACE 1
BSSGP0LL DC    0F'0'               START OF LABELS
         GL0BA ACTION=TABLE        GENERATE LABELS
BSSGP0L1 EQU   *-BSSGP0LL          LENGTH OF LABELS
         DC    3CL8' '             ROUND TO WHOLE NUMBER OF LINES
         ORG   BSSGP0LL+((((BSSGP0L1-1)/32)+1)*32)
BSSGP0A  EQU   *-BSSGP0LL          SIZE OF TAGGED AREA
         SPACE 1
BSSGP1LL DC    0F'0'               START OF LABELS
         GL1BA ACTION=TABLE        GENERATE LABELS
BSSGP1L1 EQU   *-BSSGP1LL          LENGTH OF LABELS
         DC    4CL8' '             ROUND TO WHOLE NUMBER OF LINES
         ORG   BSSGP1LL+((((BSSGP1L1-1)/32)+1)*32)
BSSGP1A  EQU   *-BSSGP1LL          SIZE OF TAGGED AREA
          .
          .
          .
          .
         SPACE 1
BSSGPFLL DC    0F'0'               START OF LABELS
         GLFBA ACTION=TABLE        GENERATE LABELS
BSSGPFL1 EQU   *-BSSGPFLL          LENGTH OF LABELS
         DC    4CL8' '             ROUND TO WHOLE NUMBER OF LINES
         ORG   BSSGPFLL+((((BSSGPFL1-1)/32)+1)*32)
BSSGPFA  EQU   *-BSSGPFLL          SIZE OF TAGGED AREA
         SPACE 1
         END   DXCDTPGT            GLOBAL AREA TAGS TABLE
        ...
```

Figure 108. Example diagnostic file processor global tags table.

IPARS - ALCS V2 (which is supplied with the ALCS shipment) contains an example diagnostic file processor global tags table (member DXCDTPGL) which you can use.

Customization for Online Communication Table Maintenance (OCTM) - DXCUTMOL

The OCTM facility includes an offline support program called DXCCTMOL. You can use this program in conjunction with the installation-wide exit program DXCUTMOL, to apply changes to the communication resources that are managed by OCTM.

The primary input to the OCTM offline support program is the OCTM sequential file created by the OCTM database backup function (activated by the ZOCTM BACKUP command).

When you run the OCTM offline support program, it calls (MVS LINK macro) DXCUTMOL for each communication resource on the input OCTM sequential file. Use this exit to identify changes that you require in the communication resources on the OCTM database. This can be a convenient way to apply changes to a large number of communication resources. You can also use this exit to delete communication resources.

A sample version of this exit program is provided with the product in the installation-wide exit library. This is called DXCUTM00. IBM advises that you use this as the basis for your installation-wide exit. You must create a load module with the entry point DXCUTMOL. This must also be the name (or an alias) of the load module.

This installation-wide exit DXCUTMOL is activated by the OCTM offline support program DXCCTMOL for every communication resource on the input OCTM sequential file (which also includes recent in-progress changes). DXCUTMOL is activated with register R01 pointing to a parameter list. Each parameter in the list points to a field containing information about the communication resource. Some of that information can not be modified by this exit program (for example, the resource name) but other information can be modified.

DXCCTMOL enters DXCUTMOL with the following conditions:

param_1

Restriction - Do not alter this field:

The address of the resource name (CRN). The CRN resides in an 8-byte field and is left adjusted.

param_2

Restriction - Do not alter this field:

The address of the resource identifier (CRI). The CRI resides in a 4-byte field and is right adjusted.

param_3

Restriction - Do not alter this field:

The address of the resource ordinal number. The ordinal number resides in a 4-byte field and is right adjusted. It is a hexadecimal number.

param_4

Restriction - Do not alter this field:

The address of the resource type (LDTYPE). The resource type resides in an 8-byte field and is left adjusted. It is one of the following:

MQTERM
WASTERM
NETVIEW
OSYS
TCPIPALC
VTAM3270
VTAMALC
X25ALC
X25PVC

param_5

The address of the test resource status. The test status resides in a 4-byte field. When a resource has test status (the ALCS STV function supplies input messages for the resource) this 4-byte field contains a non-zero value. When a resource does not have test status, this field contains hexadecimal zeros.

param_6

The address of the initial status for the resource. The initial status resides in a 4-byte field. When a resource has an initial status of active, this field contains hexadecimal zeros; when the initial status is inactive, this field contains hexadecimal 4; and when the initial status is shared, this field contains neither zeros nor 4.

param_7
The address of the logon status for the resource. The logon status resides in a 4-byte field. When a user is not required to logon at this communication resource, the 4-byte field contains hexadecimal zeros. When logon is required, this field contains a non-zero value.

param_8
The address of the logon default user-ID for the resource. The default user-ID resides in an 8-byte field and is left adjusted.

param_9
The address of the resource device type. The device type resides in an 8-byte field and is left adjusted. It is one of the following:

3270DSP
3270PRT
1977
1980
2915
4505

param_10
The address of the communications user data. The user data for the resource resides in a 127-byte field.

The OCTM offline program tests the following return codes in register 15 on return from DXCUTMOL:

R15 = 0
Nothing has been modified or deleted.

R15 = 4
Delete the resource.

R15 = 8
One or more parameters have been modified.

Chapter 8. Creating the database and general files

This section describes how to create the ALCS real-time database and general files. It also describes the input and control data formats for the following ALCS utility programs:

DXCSTCDR
System test compiler (STC) data record information library DRIL CREATE

DXCSTCED
STC input editor

DXCSTC
STC.

Before executing the online monitor, initialize the database and general file data sets, as follows:

1. Use MVS access method services (AMS) to allocate the database data sets and general file data sets. ALCS includes an ISPF panel to do this.
2. ALCS automatically preformats a data set if the high-used RBA is zero. Use this facility, or any technique described in the appropriate *VSAM Administration Guide* to initialize the database data sets and general file data sets. This is sometimes called **initial load** or **data set preformat**.

If the application requires any database record to contain initial data, and does not provide an initialization facility that executes under the control of ALCS, you should carry out the following steps:

1. Use MVS access method services (AMS) to allocate the data record information library (DRIL) data set. "Defining the DRIL data set" on page 463 describes how to do this.
2. Use the ALCS STC DRIL CREATE program (DXCSTCDR) to create and load the DRIL data. "Job control statements to run the ALCS STC DRIL CREATE" on page 463 describes how to do this. Records in the DRIL data set define the structure of application data records.
3. Use the ALCS STC input editor (DXCSTCED) and the ALCS STC (DXCSTC) to create a data file that contains the application data records. "Running the ALCS system test compiler" on page 468 describes how to do this. Some applications provide utility programs that build the input for the ALCS STC input editor.
4. Execute the ALCS monitor and use the ZDATA command (described in the *ALCS Operation and Maintenance*) to load these records from the data file.

Some applications provide a facility to define and load data to general files using VSAM. Alternatively, use the ZAFIL or the ZDATA command to load the data.

Using ISPF panels to create the data sets

This section describes how to allocate and initialize:

- The database
- The general-file data sets
- The configuration data sets

The DbData file

When you use the ISPF panels to run your DASD generation, details about the required VSAM data sets are recorded in the DbData file, which is a member of a partitioned data set (with a RECFM of FB and a record length of 80). The details recorded include:

- Number of data sets
- Names of data sets
- Sizes of data sets.

© Copyright IBM Corp. 2003, 2019

Note: If you do not use the ISPF panels to run your DASD generation, you can still ensure the DASD generation creates a DbData file by specifying the DBDATA parameter on the ALCS macro. See "ALCS macro" on page 70.

When you select **Create database data set** from the **Generation and database creation** menu, the **Read ALCS database data (DbData) file** panel is displayed. The ALCS panels load the DbData file when you select **Read and return** from the **File** pulldown on that panel.

Alternatively:

1. Select **Cancel** from the **File** pulldown on that panel
2. Select **Cancel** from the **File** pulldown on the **Read ALCS volume data (VolData)** panel
3. Select **Read database data (DbData) file** from the **File** pulldown menu on the **Create ALCS database data sets** panel

to display the **Read ALCS database data (DbData) file** panel.

The **Create database data set** panel shows the volume serial number (VolSer) of any data set that already exists. Figure 109 on page 458 shows an example of the panel.

```
  File  Options
┌──────────────────────────────────────────┐ ──────────────────────────────
│ 1  1. Read database data (DbData) file   │ data sets          Row 13 from 41
│    2. Read volume data (VolData)         │ ──────────────────────────
│    3. Create selected data sets online   │
│    4. Create selected data sets in batch │ e pull down.
│    5. Exit (PF3)                         │
└──────────────────────────────────────────┘
ALCS system  . . : SMPE      IBM-supplied SMP/E system definitions
Data set prefix  : DXC.V2R4M1

    ----------------------------Data set details-----------------------------
S   Data set name                                Records    VolSer   Status
      Messages                                                    CI Sz  Rc  Sz
_   DXC.V2R4M1.L1.C2.W0000002                      65536   IAL231   Exists
                                                                    512    504
_   DXC.V2R4M1.L2.C1.W0000001                      65536   IAL233   Exists
                                                                   1536   1528
_   DXC.V2R4M1.L2.C1.W0000002                      65536
                                                                   1536   1528
```

Figure 109. ISPF panel: The VolSer display from the DbData file

Use the following steps to select and create data sets (online):

1. Complete the entry.
   a. Add the required volume serial number
   b. For a general file, add the number of records.

      For realtime database data sets the number of records is rounded up to an integral multiple of the segment size (64K records). Overtype this value if you want to allocate a different number.
2. Select the data sets you want to create.
3. Select the **Create selected data sets online** option from the **File** pulldown menu.
4. If you are creating a large number of data sets or you are preparing a job to run at a later time, select the **Create selected data sets in batch** option from the **File** pulldown menu. This enters ISPF edit with the generated IDCAMS job stream for you to save in your own data set.

Loading volume serial data from VolData

The ISPF panels allow you to load volume serial data from VolData members. You can use this facility to avoid entering large amounts of data manually.

To do this, you must create your own VolData library with members named VOLC*c*L*l* or VOLC*c*. The library must be a partitioned data set with a RECFM of FB and a record length of 80. You may find it convenient to use the same library that contains the DbData member.

Members named VOLC*c*L*l* must contain one line for each data set of size L*l* and the first non-blank field of each line must be the VolSer where the corresponding copy C*c* data set is to reside.

Members named VOLC*c* are similar but contain the VolSers of the data sets which do not have a VOLC*c*L*l* member.

ALCS real-time database data sets have data set names of the form:

prefix.L*l*.C*c*.*snnnnnnn*

Where:

prefix
The prefix defined during DASD generation.

L*l*
ALCS record size L1 though L8.

C*c*
Copy 1 or copy 2 of the data set

snnnnnnn
The system ID (*s*) and data set sequence number (*nnnnnnn*).

Figure 110 on page 459 shows how the ISPF panels locate the volume serial data for a real-time data set.

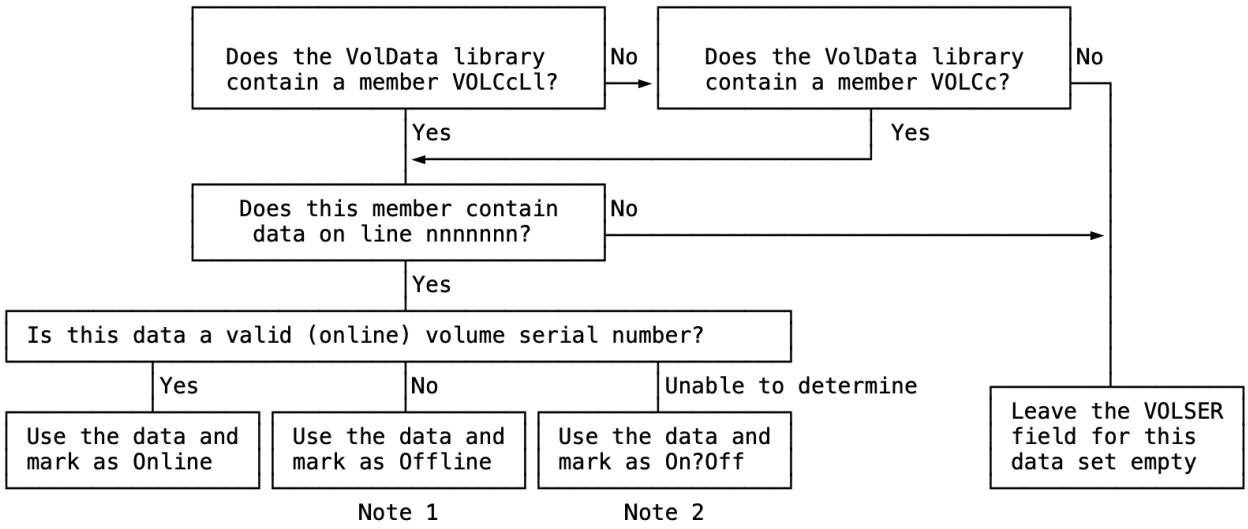


Figure 110. Determining the volume serial number for a data set

Notes:

1. The panels allow the use of offline data sets; they can generate JCL for volumes which:
   - Are currently not defined
   - Are varied offline
   - Exist on a remote system.
2. The panels may be unable to determine the status of a volume if the extended MCS console support is not enabled. See the *ALCS Program Directory* or RACF publications for more information on enabling MCS support.

The VolData members can contain volume serial numbers which do not currently exist for data sets not currently in use. This allows your system administrator to allocate (in advance) a range of volume serial numbers for ALCS to use.

The following examples clarify how this data is used:

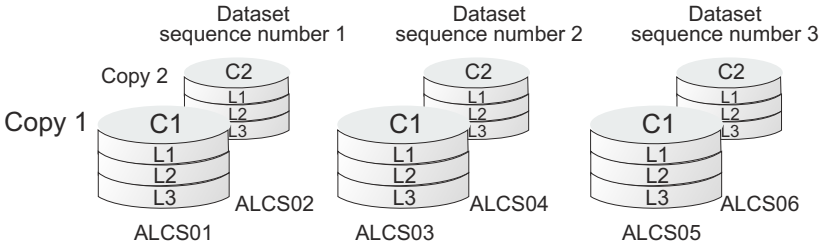


Figure 111. Example 1: Traditional layout: L1, L2, and L3 share a volume

The most compact way to define this system layout is:

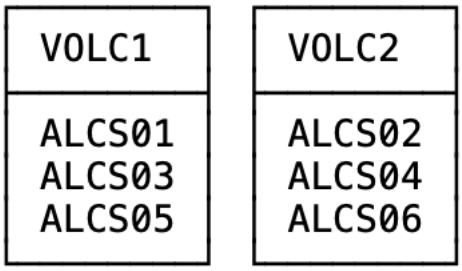


Figure 112 on page 460 shows the layout from Figure 111 on page 460 with an L4 size added.

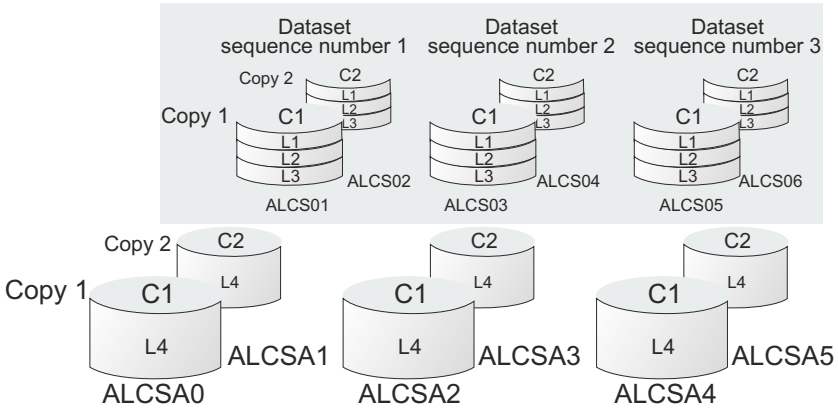
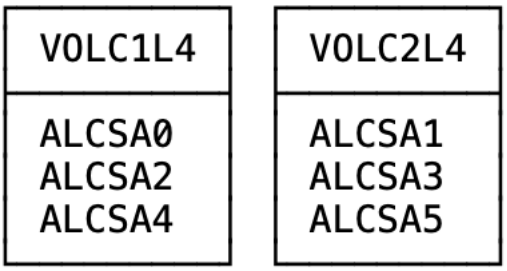


Figure 112. Example 2: Add some L4 size to the layout - one data set sequence number for each pack

The VolData members VOLC1 and VOLC2 remain unchanged because (for any given extent) VOLC*c*L*l* is searched for **before** VOLC*c*. Figure 110 on page 459 shows this sequence. To add the new size, simply add two new members as follows:



The traditional layout of Figure 111 on page 460 and Figure 112 on page 460 leads to a dual **hot spot** (accesses to the same area of DASD) at the dispense point for L1 and L2 long-term pool. You can place L1 and L2 different volumes to avoid this problem.

Figure 113 on page 461 shows this layout.

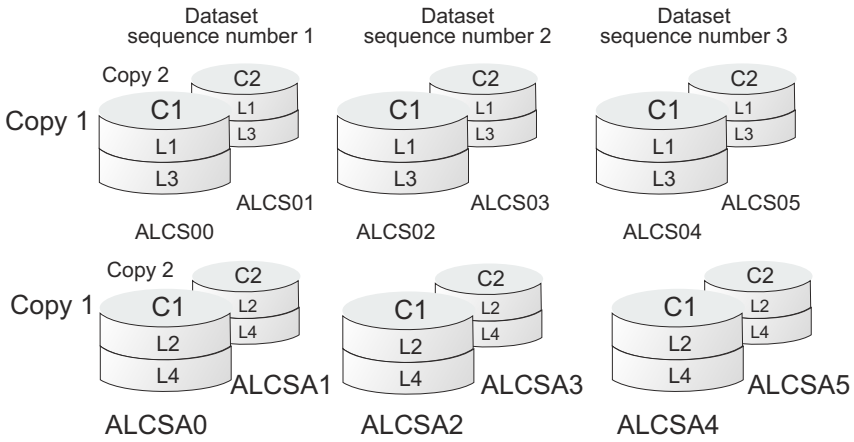


Figure 113. Example 3: Alternative layout to avoid dual hot spots

You can describe this layout with either:

- 8 explicit VOLC*c*L*l* members
- 2 default (VOLC*c*) and 4 explicit (VOLC*c*L*l*)

The following example shows 2 default and 4 explicit.

Default	Default	Explicit	Explicit	Explicit	Explicit
VOLC1	VOLC2	VOLC1L2	VOLC2L2	VOLC1L4	VOLC2L4
ALCS00 ALCS02 ALCS04	ALCS01 ALCS03 ALCS05	ALCS00 ALCS02 ALCS04	ALCS01 ALCS03 ALCS05	ALCS00 ALCS02 ALCS04	ALCS01 ALCS03 ALCS05

Defining and initializing the database and general file data sets manually

All ALCS database and general file data sets are VSAM entry-sequenced data sets (ESDSs) or relative-record data sets (RRDSs) cataloged in an integrated catalog facility (ICF) catalog.

Each data set must have only a single extent. Each data set contains records of one size only. ALCS supports up to eight record sizes, L1 through L8. Specify the record sizes in the ISPF system definition panels. If you are not using the ISPF panels, specify the record sizes in the ALCS generation (CISIZE= parameter of the ALCS macro). These values must be valid VSAM control interval sizes; see the appropriate *VSAM Administration Guide* for details. ALCS allocates one record to each VSAM control interval.

The VOLUMES= parameter of the ALCS generation DBGEN macro specifies the number of database data sets for each record size. From the record allocation data, the ALCS generation calculates the number of records required for each record size. Each database data set of a given record size contains the same number of records (see Chapter 8, “Creating the database and general files,” on page 457). The ALCS generation database DASD requirements report states this number of records.

Ensure that there is enough space for each of the data sets. To minimize database response time, it is advisable to allocate each data set to a different head disk assembly (HDA). Specify the number of records on the RECORDS= parameter of the AMS DEFINE CLUSTER command.

For database data sets, the DSNAME= parameter of the ALCS generation DBGEN macro instruction specifies the data set name prefix or prefixes (see "DBGEN macro" on page 186). The ALCS database generation DASD requirements report includes the full data set name for each data set. This is the data component name, not the VSAM cluster name.

When defining the data set, specify the name as indicated in the generation report for the data component; specify a different name for the cluster as a whole.

For general file data sets, the DSNAME= parameter of the corresponding ALCS generation GFGEN macro specifies the data set name (see "GFGEN macro" on page 193). This is the data component name, not the VSAM cluster name. When defining the data set, specify this name as the name of the data component; specify a different name for the cluster as a whole.

The CONTROLINTERVALSIZE parameter on the AMS DEFINE CLUSTER command must specify the appropriate size in bytes. The RECORDSIZE parameter must be 8 bytes less than the value specified for CONTROLINTERVALSIZE. For example, if the ALCS generation specifies on the ALCS macro

```
CISIZE=(512,1536,4096,4608)
```

define the record size L4 data sets as follows:

```
CONTROLINTERVALSIZE(4608) -
RECORDSIZE(4600 4600)
```

Always specify the NONSPANNED parameter in the AMS DEFINE CLUSTER command for all ALCS database and general file data sets. For details of the AMS DEFINE CLUSTER command, see the *&mvsspref* manual. The ZDASD VARY,ON command automatically preformats a data set if the high-used RBA is zero. Use this technique, or any technique described in the appropriate *VSAM Administration Guide*, to initialize the database data sets and general file data sets. This is sometimes called **initial load** or **data set preformat**. This process formats the allocated area of DASD. It writes each VSAM control interval with the correct VSAM control information, and writes DASD track format records. Use any of the techniques described in the appropriate *VSAM Administration Guide* to do this.

Creating a test data set

Use the following sequence of ALCS ISPF panel operations to create a test data set:

- Select **Generation and database creation** from the primary menu. (Figure 1 on page 1 shows this panel)
- Select **Create test database data set** (Figure 26 on page 61 shows this panel)
- Enter the required information
- Use the **File** pulldown menu to submit the job to MVS.

It is not necessary to load any data into the test data set before ALCS uses it.

Running the ALCS system test compiler DRIL CREATE

The ALCS system test compiler (STC) DRIL CREATE is a utility that creates data in the data record information library (DRIL) data set. The ALCS system test compiler (STC) uses the DRIL data set to construct data records for loading to the ALCS database or general files. The DRIL data set can simplify input for the ALCS STC; for example, it allows symbolic labels instead of numeric displacements.

Defining the DRIL data set

Before running ALCS STC DRIL CREATE, use VSAM access method services (AMS) to define the DRIL data set. The DRIL data set is a key-sequenced cluster, with fixed records of 29 bytes. The following DEFINE CLUSTER parameters are mandatory:

INDEXED
The DRIL data set is a key-sequenced cluster.

REUSE
This allows the DRIL data set contents to be changed by subsequent executions of the ALCS STC DRIL CREATE without the data set having to be deleted.

RECORDSIZE(29 29)
The DRIL data set consists of fixed length records with a length of 29 bytes.

KEYS(8 0)
The key is the first 8 bytes of each record.

For details of the AMS DEFINE CLUSTER command, see the appropriate *MVS Integrated Catalog Administration Access Method Services Reference* manual.

Job control statements to run the ALCS STC DRIL CREATE

The following paragraphs describe each job control statement.

EXEC statement

Include the EXEC statement:

```
EXEC  PGM=DXCSTCDR
```

where DXCSTCDR is the member name of the ALCS STC DRIL CREATE program.

JOBLIB or STEPLIB DD statement

Include a JOBLIB or STEPLIB DD statement to identify the library that contains the ALCS STC DRIL CREATE program load module.

SYSOUT DD statement

Include a DD statement to identify a sequential output data set for sort diagnostic messages. The SYSOUT data set must have a logical record length of 121 bytes. It must consist of fixed-length records with an ISO/ANSI printer control character in the first byte of each record. The ALCS STC DRIL CREATE program permits any blocking factor.

SYSLIST DD statement

Include a DD statement to identify a sequential output data set for DRIL CREATE diagnostic messages. The SYSLIST data set must have a logical record length of 121 bytes. It must consist of fixed-length records with an ISO/ANSI printer control character in the first byte of each record. The ALCS STC DRIL CREATE program permits any blocking factor.

DXCELST DD statement

Include a DD statement to identify a sequential output data set for DRIL CREATE edit diagnostic, error, or attention messages. The DXCELST data set must have a logical record length of 121 bytes. It must consist of fixed-length records with an ISO/ANSI printer control character in the first byte of each record. The ALCS STC DRIL CREATE program permits any blocking factor.

DXCMLST DD statement

Include a DD statement to identify a sequential output data set for a list of all the DRIL members that ALCS STC DRIL CREATE creates. The DXCMLST data set must have a logical record length of 121 bytes. It must consist of fixed-length records with an ISO/ANSI printer control character in the first byte of each record. The ALCS STC DRIL CREATE program permits any blocking factor.

SORTIN DD statement

Include a DD statement to identify a temporary work data set. The SORTIN data set must have a logical record length of 29 bytes. ALCS STC DRIL CREATE calls MVS sort routines that use this data set as input.

SORTOUT DD statement

Include a DD statement to identify a temporary work data set. The SORTOUT data set must have a logical record length of 29 bytes. The MVS sort routines create this data set. ALCS STC DRIL CREATE then copies this data set to the DRIL data set.

SYSIN DD statement

Include a DD statement to identify a data set that contains DRIL input data. The data set can be in the input stream, a member of a partitioned data set, or a sequential data set outside the input stream. The SYSIN data set must have a logical record length of 80 bytes. The ALCS STC DRIL CREATE program permits any blocking factor.

DXCDRIL DD statement

Include a DD statement to identify the DRIL data set, see “Defining the DRIL data set” on page 463.

System test compiler DRIL CREATE input

Input to create DRIL must define the whole DRIL. There is no DRIL update function.

DRIL input can include many members. Each DRIL member consists of a number of DRIL definition statements that define the format of a data record. DRIL definition statements use a syntax similar to the assembler language instructions that define DSECTs. Use the following sequence of DRIL member definition statements to define the format of a record:

- MACRO statement
- Name statement
- Length statement
- Field definition statement(s)
- MEND statement.

All DRIL member definition statements have this format:

Label
The label field starts in column 1. It can contain a string of up to 8 characters; a space in column 1 indicates that there is no label. The first space encountered indicates the end of this field. An asterisk (*) in column 1 indicates that the statement is a comment; the ALCS STC DRIL CREATE program ignores all comment statements.

Operation
The operation field starts with the first non-space character following the label field. It must contain one of the following DRIL operation codes:

```
MACRO               DC
ORG                 EQU
DS                  MEND
```

or the name of the DRIL member. The first space encountered indicates the end of this field.

Operand
The operand field starts with the first non-space character following the operation field. The first space encountered indicates the end of this field.

Comments
The comments field starts with the first non-space character following the operand field, and extends up to column 63. ALCS STC DRIL CREATE ignores the content of the comments field.

Control
Columns 64 through 71 contain field control information (see "DS and DC field definition statements" on page 465).

Sequence
Columns 73 through 80 optionally contain statement sequence numbers.

MACRO statement

The first statement of a DRIL member definition. The operation field contains the MACRO statement. The label, operand, and control fields contain spaces.

Name statement

The second statement of a DRIL member definition. The operation field contains the 5-character DRIL member name. When a DSECT macro defines the record format, use the DSECT macro name for the DRIL member name. The label, operand, and control fields contain spaces.

Length statement

The third statement of a DRIL member definition. It defines the length of the database record. Specify one of the user record lengths for the size L1 through L8 record sizes, (specified on the RECSIZE= parameter of the ALCS macro in the ALCS generation). The operation field contains DS. The operand field contains 0CL*n*, where *n* is the record length (number of bytes) in decimal. The label and control fields contain spaces.

DRIL field definition statements

These statements define fields and offsets within the data structure. The ALCS STC DRIL CREATE program permits the following statements:

```
        DS                          EQU
        DC                          ORG
```

The ALCS STC DRIL CREATE program processes these statements in much the same way as the assembler program processes the equivalent instructions. It maintains a location counter that corresponds to the offset of the field in the record. ALCS STC DRIL CREATE sets the location counter to zero when processing a MACRO member definition statement. The location counter is modified as appropriate by ORG, DS, and DC field definition statements. Unlike the assembler, ALCS STC DRIL CREATE program treats DS and DC statements identically.

DS and DC field definition statements

The label field contains the symbolic label of the data field. ALCS STC data record generation statements can use this label to reference the field. If a DSECT macro defines the record format, use the field name from that DSECT. The rules for valid labels are the same as those for valid assembler language symbols. Specify a blank label field if ALCS STC data record generation statements do not reference this field.

The operation field contains DS or DC.

The operand field specifies the type of data that the field contains and the length of the field. The format is:

```
[duplication]data_type[length]
```

Where:

duplication
The duplication factor of the field; a self-defining decimal value. When it is specified, ALCS STC DRIL CREATE increments the location counter by the length of the field multiplied by the duplication factor; the field length entry in the DRIL data set is set to the length of the field, multiplied by the duplication factor.

data_type
The data type of the field; a single alphabetic character. Specify one of the codes shown in the following table. When *length* is omitted, the *data_type* indicates the field length and alignment to ALCS STC DRIL CREATE as follows:

```
Code    Length     Alignment
        in Bytes

A       4          Fullword
B       1          Byte
C       1          Byte
D       8          Doubleword
F       4          Fullword
H       2          Halfword
Y       2          Halfword
X       1          Byte
```

length
The length in bytes of the data field. Specify L*n*, where *n* is a self-defining decimal value.

The control field specifies whether the field forms part of a table structure (that is, a repeating group of fields) and whether it contains an ALCS file address.

Columns
Control description

64-65
The total length of the fields in the group (that is, the table entry length); a 2-digit unsigned decimal number.

66-68
The number of repetitions of this group of fields (that is, the table entry count); a 3-digit unsigned decimal number.

69-70
Reserved.

71
Any character, other than a space, indicates that the field contains an ALCS file address.

The following example illustrates the use of the control field:

```
CC9DAT   DS    30CL12             WHOLE TABLE               12030
         ORG   CC9DAT
CC9ITM   DS    CL12               ONE TABLE ENTRY           12030
         ORG   *-12
CC9MFA   DS    F                  RECORD ADDRESS FIELD      12030
CC9DTG   DS    H                  HALFWORD FIELD            12030
CC9MSN   DS    XL2                TWO CHARACTER FIELD       12030
CC9ASI   DS    CL4                FOUR CHARACTER FIELD      12030
         DS    29CL12             29 MORE TABLE ENTRIES
```

Where:

CC9DAT is a table of thirty 12-byte entries.

CC9ITM is the entry name.
CC9MFA, CC9DTG, CC9MSN, and CC9ASI are the fields in the entry.
CC9MFA contains an ALCS file address.

EQU field definition statement

Defines either a 1-byte field or a field with the same characteristics as one defined in a previous DS, DC, or EQU statement. An EQU statement does not modify the contents of the location counter.

The label field contains the symbolic label of the data field. ALCS STC data record generation statements can use this label to refer to this field. The rules for valid labels are the same as those for valid assembler language symbols.

The operation field contains the statement EQU.

The operand field specifies the displacement within the record (location counter value) of the field. Specify one of the following, with or without **simple address adjustment**:

- A symbolic label defined in a previous DS, DC, or EQU statement; that is, use the displacement to the field with the specified label.
- An asterisk (*); that is, use the current setting of the location counter.

Note: Simple address adjustment is the arithmetic plus or minus sign (+ or -), followed by a self-defining decimal term.

To define the field with the same displacement, control type, and length as the previously defined field, specify an unqualified symbolic label of a previously defined field. Otherwise, the length of the field is 1 byte, and the displacement is the current setting of the location counter.

Examples of valid EQU statements:

```
CC9ALT   EQU   CC9DAT
CC9ASIX  EQU   CC9ASI+2
CC9END   EQU   *
CC9LAST  EQU   *-12
```

ORG field definition statement

Instructs ALCS STC DRIL CREATE to modify the location counter.

The label field must contain spaces.

The operation field contains the ORG.

The operand field specifies the new value of the location counter. Specify one of the following, with or without simple address adjustment:

- A symbolic label, defined in a previous DS, DC, or EQU statement; that is, set the location counter at the start of the field with the specified label.
- An asterisk (*); that is, maintain the current setting of the location counter.

Examples of valid EQU statements:

```
         ORG   CC9DAT
         ORG   CC9ASI+2
         ORG   *-12
```

MEND statement

The last statement of a DRIL member definition. The operation field contains the MEND statement. The label, operand, and control fields contain spaces.

Figure 114 on page 468 shows DRIL member definition statements that define a member XX0XX describing a 1992-byte record, and a member YY0YY describing a 456-byte record.

```
         MACRO
         XX0XX
         DS    0CL1992
         ...   (DRIL field definition statements)
         MEND
*
         MACRO
         YY0YY
         DS    0CL456
         ...   (DRIL field definition statements)
         MEND
```

Figure 114. Example of DRIL member definition statements

Running the ALCS system test compiler

Use the ALCS system test compiler (STC) to create one of the following sequential data sets:

- Data file data set (some systems call this type of data set a "pilot tape"), that is, a data set that contains records that the ZDATA command loads to the database or general files.
- Test unit data set (some systems call this type of data set a "test unit tape (TUT)"): that is, a data set that contains simulated input messages for processing by the ALCS system test vehicle (STV).

ALCS STC consists of two program load modules. The first program is the STC input editor, which validates the input data and converts it to an edited internal format. The second program, the system test compiler itself, processes the edited input together with data structure definitions on the DRIL data set. It produces a data file or test unit sequential data set. Always execute the two programs together.

Job control statements to run the ALCS system test compiler

This section explains how to run the ALCS system test compiler.

JCL to run the ALCS STC input editor

The following paragraphs describe each job control statement.

EXEC statement

Include the EXEC statement:

```
EXEC  PGM=DXCSTCED
```

where DXCSTCED is the member name of the STC input editor.

JOBLIB or STEPLIB DD statements

Include JOBLIB or STEPLIB DD statements to identify the library that contains the STC input editor and STC load modules and, for STC, the library that contains the ALCS DASD configuration load module.

SYSPRINT DD statement

Include a DD statement to identify a sequential output data set for diagnostic messages from the ALCS STC, and diagnostic error or attention messages concerning the input data. The SYSPRINT data set must have a logical record length of 121 bytes. It must consist of fixed-length records with an ISO/ANSI printer control character in the first byte of each record. The ALCS STC permits any blocking factor.

DXCESI DD statement

Include a DD statement to identify a temporary sequential output data set for the edited data; this is input to DXCSTC. The DXCESI data set must have a logical record length of 424 bytes.

SYSIN DD statement

Include a DD statement to identify a data set that contains STC input data. The control data set can be in the input stream, a member of a partitioned data set, or a sequential data set outside the input stream. The SYSIN data set must have a logical record length of 80 bytes. The ALCS STC permits any blocking factor.

JCL to run the ALCS STC

The following paragraphs describe each job control statement.

EXEC statement

Include the EXEC statement to execute the system test compiler containing (at least):

```
PGM=DXCSTC
PARM='/dt'
```

specifying the member name (*dt*) of the ALCS DASD configuration load module.

Ensure that the ALCS using the resulting data file, or test unit data set, includes the same DASD configuration load module.

SYSLIST DD statement

Include a DD statement to identify a sequential output data set for diagnostic messages from the ALCS STC. The SYSLIST data set must have a logical record length of 121 bytes. It must consist of fixed-length records with an ISO/ANSI printer control character in the first byte of each record. The ALCS STC permits any blocking factor.

DXCLIST DD statement

Include a DD statement to identify a sequential output data set for diagnostic, error, or attention messages. The DXCLIST data set must have a logical record length of 121 bytes. It must consist of fixed-length records with an ISO/ANSI printer control character in the first byte of each record. The ALCS STC permits any blocking factor.

SYSIN DD statement

Include a DD statement to identify a sequential temporary input data set for the edited input data produced by DXCSTCED as the DXCESI data set. The SYSIN data set must have a logical record length of 424 bytes.

DXCTUT DD statement

Include a DD statement to identify the output data file or test unit data set. On the DCB parameter specify RECFM=VB or RECFM=V. Specify other DCB parameters and the UNIT parameter to match those of an input sequential data set entry in the ALCS sequential file configuration of the ALCS system that uses this data set.

DXCDRIL DD statement

Include a DD statement to identify the DRIL data set. This data set contains information about the application record data structures. It is a VSAM key-sequenced cluster. See “Running the ALCS system test compiler DRIL CREATE” on page 462 for details on how to create the DRIL data set.

System test compiler input

The ALCS STC is designed to be compatible with input for the TPF and ALCS/VSE versions of STC.

There are three types of input statements for the ALCS system test compiler (STC):

Control
Control the processing of STC.

Data record definition
Define the contents of data records that STC writes to the data file sequential data set.

Message definition
Define the contents of message records that STC writes to the test unit sequential data set.

STC processes columns 1 through 72 of each statement. It does not process columns 73 through 80; use these columns for statement sequence numbers, if required. STC interprets as a comment statement any statement that has an asterisk (*) as the first non-space character, except where the preceding statement indicates a continuation. STC allows continuation for some data record definition and message definition statements. Code any character other than a space in column 72 to indicate that a continuation statement follows.

STC control statements

Print control statement

Use the @PRINT control statement to control printing of the input statements. The format of this control statement, starting in column 1, is:

@PRINT={NO|YES}

Where:

NO
Suppress listing of the input statements.

YES
List the input statements.

Page length control statement

Use the @PL control statement to control the number of print lines per page on the SYSPRINT output data set. The format of this control statement, starting in column 1, is:

@PL={55|*length*}

Where *length* is the number of lines per page; a decimal value. If this parameter is omitted this control statement defaults to printing 55 lines per page.

RUNID control statement

Use the RUNID control statement to name the following group of data record or message definition statements. The format of this control statement is:

Columns
Contents

1
Space

2-6
RUNID

7
Space

8-12
For data file generation: PILOT.
For test unit generation: a 5-character test unit name. This name must **not** be PILOT.

13
Space

14
For data file generation: a single alphabetic character (the "name" of the file).
For test unit generation: a space.

15-72
Space

DATA or MSG control statement

Use the DATA control statement to identify the statements as data record generation statements. Use the MSG control statement to identify the statements as message generation statements.

Include either a DATA or a MSG control statement immediately following the RUNID control statement. STC treats all following statements, up to the end of the input data set, as generation statements. DATA or MSG must start in column 2. All other fields in the statement are spaces. Figure 115 on page 471 shows an example.

```
//SYSIN    DD *
@PRINT=YES                        PRINT CONTROL
@PL=50                            PAGE LENGTH
 STC
 RUNID PILOT T
 DATA
 ... (data record generation statements)
/*
```

Figure 115. Example of STC control statements to define a data file

Data record generation statements

Use data record generation statements to specify the contents of data records that STC writes to the data file. These statements have a format similar to the format of assembler language instructions. Each statement contains:

Label
The label field starts in column 1; it can contain a string of up to 8 characters. A space in column 1 indicates that there is no label. The first space indicates the end of this field. The label field can be any of the following:

- A symbolic label (defined in DRIL)
- The STC label BSTA06
- A self-defining decimal value.

Operation
The operation field starts with the first non-space character following the label field. The first space indicates the end of this field. The operation field contains STC operation codes, one of the following:

GSTAR
Indicates the start of a group of records.

GEND
Indicates the end of a group of records.

ENT
Specifies the contents of a data field within a record.

REP
Specifies the contents of a data field within a group of records.

ENTIT
Specifies the contents of an item in a data table.

REPST
Specifies the contents of several items in a data table.

ADD
Specifies the contents of a data field within a group of records. A specified increment varies the data.

ADDST
Specifies the contents of several items in a data table. A specified increment varies the data.

SUB
Specifies the contents of a data field within a group of records. A specified decrement varies the data.

SUBST
Specifies the contents of several items in a data table. A specified decrement varies the data.

Operand
The operand field starts with the first non-space character following the operation field. The first space indicates the end of this field.

Comments
The comments field starts with the first non-space character following the operand field; it extends up to column 71. ALCS STC ignores the contents of the comments field.

Continuation
ALCS STC allows continuation for all data record generation statements, except GSTAR and GEND.

Indicate operand continuation by coding a non-space character in column 72. Code the continuation data starting in any column except column 1.

Sequence
Columns 73 through 80 optionally contain statement sequence numbers.

Operand format for data record generation statements

This section describes the operand field for the data record generation statements, except GSTAR and GEND.

Each group of data record generation statements, delimited by GSTAR and GEND, generates one record or a group of records. When generating several records, each field definition card within the group must specify:

- The data that the field contains
- Which record or records of the group contain the data.

If the DRIL defines the field as an entry within a table structure, the field definition statement must specify which table entry contains the field.

The operand field of STC data record generation statements combine these three pieces of information. The description of the operand format uses the following terms:

Data
The part (sub-field) of the operand specifying the data.

Parameters
The part (sub-field) of the operand specifying any records or table entries that STC initializes with the data.

The formats of the operand field are:

```
data[-parameters]
parameters-data
```

Use the second format, parameters followed by data, if and only if the data contains a hyphen character (-) and the parameters are not null.

The format of the parameters sub-field depends on the operation code; see the descriptions of each operation code later in this section. Specify the data sub-field with:

Character
Specify character data in the format C'*string*'. When the character string includes a quote (') character, code 2 quotes for each quote in the string. For example:

```
C'ABCD,1234 ###  ***'
C'ST. JAMES''S PARK'
C'''WORD'''
```

Numeric
Specify numeric data in the format N'*decimal_integer*'. STC initializes the field with the binary representation of the specified decimal integer. For example:

```
N'42'
N'1986'
```

Hexadecimal
Specify hexadecimal data in the format X'*hexadecimal_digits*'. STC interprets the alphabetic characters I, O, Z, and S as numeric 1, 0, 2, and 5, respectively. For examples:

```
X'FFE6'
X'OIAS'   (interpreted as X'01A5')
```

Binary
Specify binary data in the format B'*bit_string*'. For example:

```
B'1000'
B'11001111'
```

File address
Specify file addresses in the format:

```
(fixed_file_type_symbol)record_ordinal
```

Code *fixed_file_type_symbol* as a symbol up to 8 characters long specified in your record type equate macro (DXCURID). See “ALCS DXCURID user macro” on page 443. Code the *record_ordinal* as a decimal number with a maximum of 5 digits. For example:

```
(#GLOBL)19
(#CCTRI)548
```

ALCS STC pads numeric, hexadecimal, or binary data to the left with zero bits to form a whole number of bytes. If the label of the data record definition statement is a self-defining decimal term, STC determines the length of the field from the number of bytes of data specified. If the label of the data record definition

statement is a symbolic label (defined in DRIL), STC determines the length of the field from the DRIL definition; padding and truncation depends on the data type code defined in DRIL.

GSTAR data record definition statement

For each group of records, code one GSTAR statement as the first data record definition statement of the group.

The label field is one of:

- The name of the DRIL member that defines the record. ALCS STC uses the DRIL member to determine the size of the record.
- Length of the record, a decimal number. Specify one of the user record lengths for the L1 through L8 record sizes, as specified on the RECSIZE= parameter of the ALCS macro in the ALCS generation.

The operation field is GSTAR.

The operand field specifies the number of records in this group that STC creates. Code this as a decimal number in the range 1 through 50.

GEND data record definition statement

Code this as the last statement of each group of records. The label and operand fields contain spaces. The operation field is GEND.

ENT data record definition statement

Use the ENT operation to generate constant data in one field of a record, or all records of a group.

The label field specifies which field within the record(s) STC initializes. Code one of the following:

- Offset in the record, a decimal number.
- A symbolic label defined in the DRIL member named in the GSTAR record definition statement for this record or group of records.
- BSTA06. Use this to specify the file address of the record or the file addresses of the group of records.

The operation field is ENT.

The parameters sub-field of the operation field is null.

The data sub-field of the operation field specifies the data items for the field that STC generates, separated by commas. STC places the first data item in the first record, the second data item in the second record, and so on.

For example, to generate one 456-byte record with the character string SU920 at displacement 100 (decimal), code:

```
456    GSTAR 1
BSTA06 ENT   (#XXXRI)42
100    ENT   C'SU920'
       GEND
```

To generate three records described by the DRIL member CC9TR code:

```
CC9TR  GSTAR 3
BSTA06 ENT   (#CCTRI)1,(#CCTRI)2,(#CCTRI)3
CC9BID ENT   C'CT',C'CT',C'CT'
CC9CUR ENT   C'UKL',C'IRL',C'KES'
⋮
       GEND
```

STC generates all three records as shown in Table 35 on page 475.

Table 35. Example of STC generating records		
Field	**Ordinal number**	**Contents**
CC9BID	1,2,3	CT
CC9UR	1	UKL
	2	IRL
	3	KES

STC permits the operand field to contain more than one type of data item, and to have continuations. For example:

```
                                                    Col 72

CC9CUR ENT   C'LIT',                                X
             X'8042',                               X
             N'17'
```

REP data record definition statement

Use the REP operation to generate constant data in one field of some or all records of a group of records.

The label field specifies which field within the records STC initializes. Code one of the following:

- A self-defining decimal offset in the record.
- A symbolic label defined in the DRIL member named in the GSTAR record definition statement for this record or group of records.
- BSTA06 to specify the file address of the record or the file addresses of the group of records.

The operation field is REP.

The parameters sub-field of the operation field has the form:

```
first-last
```

Where:

first
Record number of the first record of the sub-group, a decimal number.

last
Record number of the last record of the sub-group, a decimal number.

STC initializes only the specified field on the records in the sub-group starting with *first* and ending with *last*.

The data sub-field of the operation field specifies the data item for the fields generated by STC.

For example, to generate the character string HAL at offset decimal 40 in the first three records of a group of six, and the character string JTC at offset decimal 40 in the remaining records, code the following:

```
456    GSTAR 6
⋮
40     REP   C'HAL'-1-3
40     REP   C'JTC'-4-6
⋮
       GEND
```

STC permits the operand field to contain more than one type of data item, and to have continuations.

ENTIT data record definition statement

Use the ENTIT operation to generate constant data in one field of one entry in a data table within one record.

The label field specifies which field within the record is initialized by STC. Code a symbolic label defined in the DRIL member named in the GSTAR record definition statement for this record or group of records. The DRIL member must define this field as a data table entry.

The operation field is ENTIT.

The parameters sub-field of the operation field has the form:

```
recno-entry
```

Where:

recno
Record number of the record within the group, a decimal number. If generating only one record, code 1.

entry
Table entry number, a decimal number.

The data sub-field of the operation field specifies the data item for the field.

For example, to generate the third and eighth table entries in the second of three records that the DRIL member CC9TR describes, code the following:

```
CC9TR  GSTAR 3
⋮
CC9MFA ENTIT (#CCTRI)20-2-3
CC9DTG ENTIT N'19'-2-3
CC9MSN ENTIT X'0080'-2-3
CC9ASI ENTIT C'BCNX'-2-3
CC9MFA ENTIT (#CCTRI)15-2-8
CC9DTG ENTIT N'42'-2-8
CC9MSN ENTIT X'8401'-2-8
CC9ASI ENTIT C'ELMT'-2-8
⋮
       GEND
```

STC permits the operand field to have continuations.

REPST data record definition statement

Use the REPST operation to generate constant data in one field of several entries in a data table within the first or only record of the group.

The label field specifies which field within the record is initialized by STC. Code a symbolic label defined in the DRIL member named in the GSTAR record definition statement for this record or group of records. The DRIL definition for this field must specify this field as a data table entry.

The operation field is REPST.

The parameters sub-field of the operation field is of the form:

```
first-last
```

Where:

first
First table entry number, a decimal number.

last
Last table entry number, a decimal number.

The data sub-field of the operation field specifies the data item for the field.

For example, to generate table entries 7 through 19 with identical data in the first of three records described by the DRIL member CC9TR, code:

```
CC9TR  GSTAR 3
⋮
CC9MFA REPST (#CCTRI)14-7-19
CC9DTG REPST N'55'-7-19
CC9MSN REPST X'8000'-7-19
CC9ASI REPST C'GUCW'-7-19
⋮
       GEND
```

STC permits the operand field to have continuations.

ADD and SUB data record definition statements

Use the ADD and SUB operations to generate variable data in one field of some, or all, records of a group of records.

The label field specifies which field within the records STC initializes. Code one of the following:

- Offset in the record, a decimal number.
- A symbolic label defined in the DRIL member named in the GSTAR record definition statement for this record or group of records.
- BSTA06 to specify the file address of the record or the file addresses of the group of records.

The operation field is ADD or SUB.

The parameters sub-field of the operation field is of the form:

```
increment-first-last
```

Where:

increment
Increment added to the data sub-field of the operation field for ADD operations, or the decrement subtracted from the data sub-field of the operation field for SUB operations. A decimal number.

first
Record number of the first record of the sub-group. A decimal number.

last
Record number of the last record of the sub-group. A decimal number.

STC initializes only the specified field on the records in the sub-group starting with *first* and ending with *last*; STC sets the field in the first record of the group to *data*, the field in the second record of the group to *data* plus *increment* (ADD) or minus *increment* (SUB), and so on.

The data sub-field of the operation field specifies the data item for the first field generated by STC. For example, to generate the first three records of a group of six described by the DRIL member CC9TR, code the following:

```
CC9TR  GSTAR 6
⋮
BSTA06 ADD   (#CCTRI)40-1-1-3
CC9SEQ SUB   N'98'-8-1-3
⋮
       GEND
```

STC generates the three records with record ordinal numbers of 40, 41, and 42 respectively. These records have field CC9SEQ set to 98, 90, and 82 respectively.

STC permits the operand field to have continuations. STC treats the data as a 4-byte binary number when incrementing. It ignores any overflow or underflow.

ADDST and SUBST data record definition statements

Use the ADDST and SUBST operations to generate variable data in one field of several entries in a data table within the first or only record of a group.

The label field specifies which field within the record STC initializes. Code a symbolic label defined in the DRIL member named in the GSTAR record definition statement for this record or group of records. The DRIL definition for this field must specify this field as a data table entry.

The operation field is ADDST or SUBST.

The parameters sub-field of the operation field is of the form:

```
increment-first-last
```

Where:

increment
Increment added to the data sub-field of the operation field for ADDST operations, or the decrement subtracted from the data sub-field of the operation field for SUBST operations. A decimal number.

first
First table entry number, a decimal number.

last
Last table entry number, a decimal number.

The data sub-field of the operation field specifies the data item for the first field that STC generates.

For example, to generate table entries 17 through 19 in the first of three records described by the DRIL member CC9TR code the following:

```
CC9TR  GSTAR 3
⋮
CC9MFA ADDST (#CCTRI)14-1-17-19
CC9DTG SUBST N'55'-5-17-19
CC9MSN SUBST X'8080'-1-17-19
CC9ASI ADDST C'AAAA'-1-17-19
⋮
       GEND
```

STC permits the operand field to have continuations. STC treats the data as a 4-byte binary number when incrementing. It ignores any overflow or underflow.

Message generation statements

Use "free format" message generation statements to generate simulated input messages for the ALCS system test vehicle (STV). The format of these message generation statements is:

```
cri text+
```

Where:

cri
Communication resource identifier (CRI) of the simulated terminal. Code a 6-hexadecimal-digit number starting in column 1.

text
Text of the simulated message. Code this starting in card column 7.

+
End-of-message character (X'4E'). Code this immediately following the message text. Omit this character for WTTY messages.

STC permits continuations to free format message generation statements. Code a non-space character in column 72 to denote continuation. Start the continuation statement in any column. STC ignores trailing spaces on the continued statement and leading spaces on the continuation statement.

ALCS STC recognizes the following special characters:

In non-WTTY messages:

¬	(hexadecimal 5F)	new line (#CAR)
+	(hexadecimal 4E)	end of message (#EOM)

In WTTY messages:

¬	(hexadecimal 5F)	carriage return (#CAR)
%	(hexadecimal 6C)	line feed (#LF)

For example, to generate the simulated message "A 15SEP LED VKO" originating from the terminal with CRI of 0200C7, code the following:

```
0200C7A 15SEP LED VKO+
```

or:

```
                                                    Col 72

0200C7A 15                                          X
    SEP LED VKO+
```

Chapter 9. Long-term pool space recovery - Recoup

The *ALCS Concepts and Facilities* gives an overview of the Recoup utility and explains the concepts of **groups** and **indexes** in ALCS application databases. This section explains how to use the different forms of the INDEX macro and GROUP macro to describe the application database to Recoup.

Specifying data structures to Recoup

To allow Recoup to identify the long-term pool file records that are in use, you supply a description of the application database. ALCS provides the INDEX macro to describe how one record points to another, and the GROUP macro to describe sets of records with the same features. Table 36 on page 480 shows where you can find detailed information about the these macros.

Table 36. Where to find more information about the INDEX and GROUP macros

Macro	
INDEX	"INDEX macro" on page 485
GROUP	"GROUP macro" on page 481

Writing descriptor programs

The descriptor programs consist mainly of GROUP and INDEX macros, but can also contain subroutines. IBM supplies sample descriptor programs BZ01 through BZ09.

Start each descriptor program with a BEGIN macro and finish with a FINIS macro followed by the assembler END instruction. Assemble, link-edit, and load descriptor programs in the same way as other ECB-controlled programs.

Descriptor programs can exceed 4KB in size.

Names of descriptor programs

For a descriptor program you can use any name that conforms to the naming conventions for ECB-controlled programs, but IBM recommends that the name begins with B. Avoid using names starting with BZ*nn*, which are reserved for IBM.

Recoup looks in program BZ00 to find a list of all the descriptor programs. Use the sample BZ00 included in IPARS to create your own BZ00 listing all your descriptor program names. See "Recoup descriptor program directory - BZ00" on page 434 for more information.

The number and size of descriptor programs depends on:

- How many prime groups you need to describe the part of the database containing long-term pool file records.
- How many prime groups you want to include in a single descriptor program. A descriptor program must start with a prime group, and may contain up to 18 prime groups.

The ZRECP command (described in the *ALCS Operation and Maintenance*) allows for partial Recoup runs. Ensure that a group name in one descriptor program is not repeated in another descriptor program.

Sequence of GROUP and INDEX macroinstructions

Every prime group that contains records having index references to other groups requires an index for each of those references. (There can be none, one, or more INDEX macros for any group.)

The way to code a descriptor program is to put all INDEX macros for a group immediately after the GROUP macro. Figure 116 on page 481 shows a database structure with three groups of records.

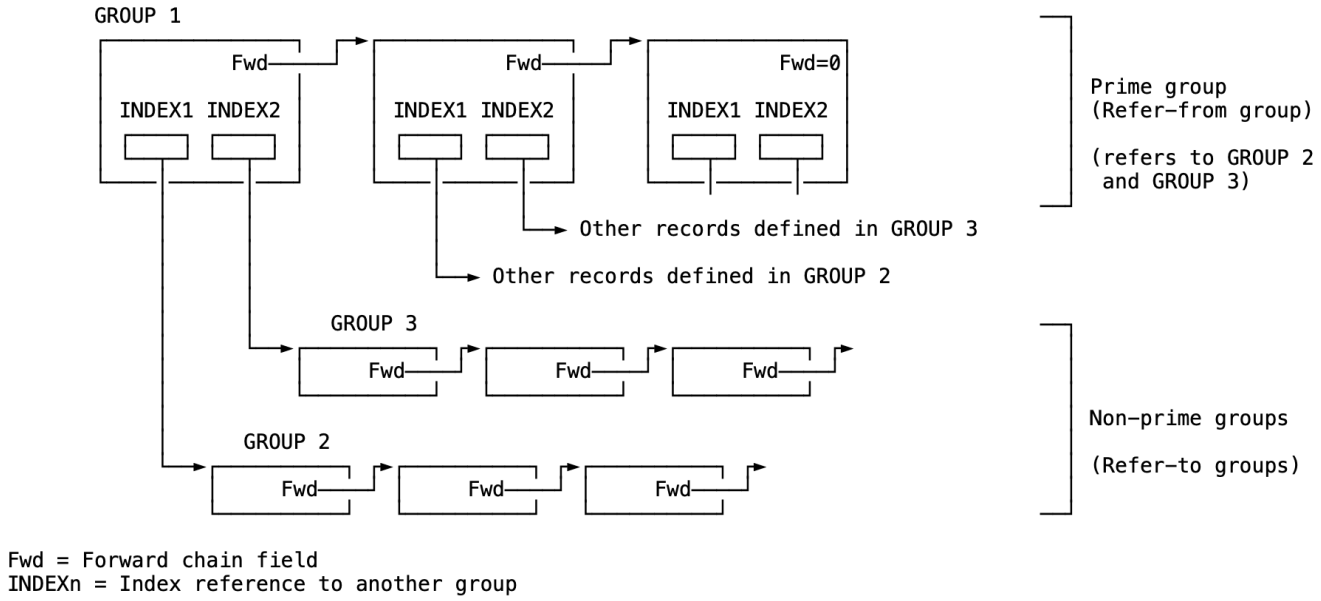


Figure 116. Group of records with index references

Figure 117 on page 481 shows the sequence of macros which describe the structure in Figure 116 on page 481.

```
label1  GROUP (GROUP macroinstruction to define GROUP 1)
                :
                INDEX  ..(INDEX macroinstruction to define INDEX1 in GROUP 1 records)
                :
                INDEX  ..(INDEX macroinstruction to define INDEX2 in GROUP 1 records)
                :
label2  GROUP  ..(GROUP macroinstruction to define GROUP 2)
                :
label3  GROUP  ..(GROUP macroinstruction to define GROUP 3)
                :
```

Figure 117. Sequence of GROUP and INDEX instructions

Notes:

1. If there is only one INDEX macro relating to a GROUP, the INDEX can have a label. Any GROUP macro can refer to this INDEX macro by using the label.
2. When an operand of a GROUP or INDEX macro specifies the displacement into a record, this can be the field name from the corresponding data macro. These field names are interpreted in the macro expansion as base and displacement (S type) addresses with general register 0 (RAC) as the base register.

GROUP macro

Use the GROUP macro to specify the structure and characteristics of a group of records. There are two types of group: prime group and non-prime group. The format of the GROUP macro is:

For prime group:

```
label GROUP PRIME=YES
            ,METHOD={(SEQ,...)|STOR|(GL,...)|(USER,...)}
            ,{ID=n|(IDSYM=id-symbol}
            [,FACEID=name]
            [,MACRO=name]
            [,FCH={n|fieldname}]
            [,BCH=([n|fieldname},{VALIDATE|RECOVER})]
            [,ENTRTN=label]
            [,EXTRTN=label]
            [,RRECRTN=label]
            [,PRECRTN=label]
            [,INDEX=label]
            [,PRIMECT={4000|n}]
            [,COUNT={250|n}]
            [,DUPREAD={BYPASS|WARN|REPEAT}]
            [,PRINT={NOGEN|GEN}]
            [,IDSUPR={NO|YES}]
            [,RTP={0|n}]
```

or for non-prime group:

```
label GROUP PRIME=NO
            [,METHOD=ID,{ID=n|(IDSYM=symbol}]
            [,MACRO=name]
            [,FCH={n|fieldname}]
            [,BCH=([n|fieldname},{VALIDATE|RECOVER})]
            [,RRECRTN=label]
            [,PRECRTN=label]
            [,INDEX=label]
            [,COUNT={250|n}]
            [,RCC={NO|YES}]
            [,DUPREAD={BYPASS|WARN|REPEAT}]
            [,PRINT={NOGEN|GEN}]
            [,RTP={0|n}]
```

label
Any valid assembler label.

Each GROUP macro requires a unique label. For prime groups, the label appears in progress messages that Recoup generates; so use a label that identifies the group.

PRIME={NO|YES}
The group is a prime group (YES) or a non-prime group (NO).

METHOD={(SEQ,..)|STOR|(GL,..)|(USER,..)}
For prime groups, the method of finding the records in the group.

METHOD=(SEQ,*n1*,*n2*)
The prime group is a set of fixed file records with the same fixed record type (specified by the FACEID= parameter) and a sequential set of ordinal numbers.

n1
Starting ordinal in decimal. If omitted, it defaults to the first ordinal (ordinal 0).

n2
Ending ordinal in decimal. If omitted, it defaults to the last ordinal.

METHOD=STOR
The prime group is all fixed records loaded into the application global area (under the control of programs GOA0 through GOAE and CGAF) with the record ID specified by the ID or IDSYM= parameter.

METHOD=(GL,*n1*,*n2*)
The prime group is one record; the file address is in a directly addressable global record.

n1
Name of the field containing the reference.

n2
One of the following:

F
Field *n1* contains a file address. Recoup reads the record from file.

G
Field *n1* is the name of a directory slot containing the main storage address of a global record. Recoup calculates the corresponding file address and reads the record from file.

C
Field *n1* contains the main storage address of a global record. Recoup uses the record already in main storage.

METHOD=(USER,*n1*,*n2*,*n3*,*n4*)
The user provides the method that defines the prime group. See "User-written program to calculate address of prime group record" on page 493.

n1
Four-character name of a user-supplied program. Recoup enters this program to calculate the file address of each record in the group. A sample of such a program is supplied with IPARS - ALCS V2. This sample program is called BZ99. It performs IPARS flight ordinal number processing.

n2*, *n3*, and *n4
Puts user information in the prime group macro expansion as follows:

- `GR1P2 DC AL4(`*n2*`)`
- `GR1P3 DC AL4(`*n3*`)`
- `GR1P4 DC AL4(`*n4*`)`

The user-supplied program can use this information as required.

METHOD=ID
The record ID of a refer-to record can be in the refer-from record. In this case, the GROUP macro that the INDEX refers to is not known when the descriptor program is coded. Recoup has to locate the GROUP macro by means of the record ID. To enable Recoup to locate GROUP macros in this way, code METHOD=ID in the GROUP macro.

{ID=*id*|IDSYM=*id-symbol*}
Record ID for the group. This is mandatory for prime groups, and for non-prime groups if METHOD=ID. In other cases, Recoup checks the record ID against an ID specified in the ID= parameter of the INDEX macro, which defines the reference to the non-prime group. It can, however, improve the readability of the descriptor program if you include ID or IDSYM in the GROUP macro as well. One of:

ID=*id*
Two alphanumeric or 4 hexadecimal characters.

IDSYM=*id-symbol*
Symbol for the record ID. An assembler EQU instruction in the data macro for the record must assign a value to this symbol.

FACEID=*name*
Fixed file record type. This is mandatory if METHOD=SEQ. Omit this operand if METHOD=STOR, METHOD=GL, or METHOD=ID.

MACRO=*name*
Name of the data macro that describes records in the group. This is required if any fields are referred to by name in the GROUP macro operands or associated INDEX macro operands.

Attention: The GROUP macro generates a USING instruction to establish register 0 (RAC) as the base register for the data macro DSECT. The GROUP macro does not generate a DROP instruction for this base register. If your descriptor program contains user-written subroutines which call the data macro then you may need to include an explicit DROP 0 instruction before calling the data macro, to avoid assembly warning messages.

FCH={*n*|*fieldname*}
Displacement from the start of the record to the forward chain field. Specify either an absolute displacement, or the field name of the forward chain field. If the forward chain field is at the standard displacement (8), specify the absolute displacement. Otherwise, to improve descriptor program readability, use the field name.

Omit FCH if the group has only one record.

BCH=([*n*|*fieldname*},{VALIDATE|RECOVER})
Displacement from the start of the record to the backward chain field. Specify either an absolute displacement or the field name of the backward chain field. It improves descriptor program readability to use the field name.

VALIDATE
Check the backward chain field of each record in the group. If it is not correct, Recoup places an attention message on the ALCS diagnostic file. If an error occurs while Recoup is reading a record of the group, the remainder of the group is read by backward chaining.

RECOVER
Do not check the backward chain field. If an error occurs while Recoup is reading a record, Recoup reads the remainder of the group by backward chaining.

If BCH is omitted, the backward chain field (if any) is ignored.

ENTRTN=*label*
Label of a user-supplied subroutine. (See "User-written subroutines" on page 491.) Recoup enters this subroutine before it processes the prime group.

ENTRTN is optional for prime groups; it does not apply to non-prime groups.

EXTRTN=*label*
Label of a user-supplied subroutine. (See "User-written subroutines" on page 491.) Recoup enters this subroutine after it processes the prime group.

EXTRTN is optional for prime groups; it does not apply to non-prime groups.

RRECRTN=*label*
Label of a user-supplied subroutine. (See "User-written subroutines" on page 491.) Recoup enters this subroutine before it reads each record in the group.

PRECRTN=*label*
Label of a user-supplied subroutine. (See "User-written subroutines" on page 491.) Recoup enters this subroutine after it reads each record in the group, but before any processing of the record.

INDEX=*label*
Label of the INDEX macro that describes records in this group. This parameter is optional.

If INDEX is used, one (and only one) INDEX macro describes the records in this group.

If INDEX is omitted, all INDEX macros between this GROUP macro and the next GROUP macro describe the records in this group.

PRIMECT={4000|*n*}
Maximum number of records expected in all groups chained from any one record in this prime group. If the count of reads reaches this value, Recoup issues an attention message, but chain chasing proceeds normally. If the count of reads reaches twice this value Recoup issues an error message and assumes that there is corruption of the database. Chain-chasing from that prime group record stops.

COUNT={250|*n*}
Maximum number of records expected in this group. If the count of reads reaches this value, Recoup issues an attention message but chain-chasing proceeds normally. If the count of reads reaches twice this value, Recoup issues an error message and assumes that there is corruption of the database. Chain chasing of the group stops.

RCC={NO|YES}

YES
Check the record code check (RCC).

The RCC= parameter of the INDEX macro that defines the reference to this group can specify the RCC for the group. If it does, Recoup checks that all the records in the group have the specified RCC.

If the INDEX macro does not specify the RCC, Recoup only checks that all the records in the group have the same RCC as the first record.

RCC checking is not used for prime groups.

NO
Do not check the RCC.

DUPREAD={BYPASS|WARN|REPEAT}
Action required if a long-term pool file record is read more than once in a chain-chase run.

BYPASS
Do not chain-chase records referred to by this record.

WARN
Do not chain-chase records referred to by this record. Put an attention message on the ALCS diagnostic file.

REPEAT
Chain-chase records referred to by this record, even though they have been read before.

PRINT={NOGEN|GEN}

NOGEN
Do not print the macro expansion.

GEN
Print the macro expansion.

IDSUPR={NO|YES}

YES
Suppress record ID checks on records that the application has not yet used.

NO
Do not suppress record ID checks on records that the application has not yet used.

RTP={0|*n*}
Record ID qualifier. *n* may take any integer value from 0 through 9.

INDEX macro

Use the INDEX macro to specify the location and type of a reference in the refer-from record to the first record in another group. You need to specify:

- The location of the first item in the record.
- The length of each item. This can be a constant (if all items are the same length), or variable (each item contains a field with the item length).
- The displacement of the reference field in each item.
- The number of items in the record.
- The name of the GROUP macro instruction that describes the group of records that the reference refers to. (You do not need to specify this if the reference refers to a single record that does not itself contain references.)

The format of the INDEX macro is:

```
[label] INDEX  REF=({FA,...|FACE,...|AAA,...|USER,...})
                         ,ID=({C,id|(S,id-symbol|(I,n})
                         [,RCC=({P,n|(I,n|(C,rcc})]
                         [,GROUP=label|(GROUP=(ID)]
                         [,item-information[,subitem-information]]
                         [,INDXRTN=label]
                         [,ITEMRTN=label]
                         [,ITEMKEY=n]
                         [,PRINT={NOGEN|GEN}]
                         [,RTP={0|n}]
```

Where:

label
Any valid assembler label. Use it in conjunction with the INDEX= parameter in the GR0UP macro to associate an index with a particular group.

REF=({FA,..|FACE,..|AAA,..|USER,..})
Type and location of the reference to another record:

REF=(FA,*n1*)
Reference is a file address. *n1* is one of the following:

- If the record is not divided into items, it is the displacement from the start of the **record** to the embedded reference.
- If the record is divided into items but the items are not divided into sub-items, it is the displacement from the start of the **item** to the embedded reference.
- If the record is divided into items and the items are divided into sub-items, it is the displacement from the start of the **sub-item** to the embedded reference.

REF=(FACE,*n1*,*n2*,*n3*)
Reference is a record ordinal number.

n1
Displacement of the embedded reference (as for **REF**=(FA,*n1*)).

n2
Length (in bytes) of the reference.

n3
Fixed file record type of the refer-to record.

REF=(AAA,*n1*)
Reference is a 3-byte CRI. This is intended for use with IPARS, or similar applications that use the AAA record. Recoup uses the IPARS program WGR1 to calculate the AAA file address from the CRI (see "AAA address compute/find utility program - WGR1" on page 420).

n1
Displacement of the reference (as for **REF**=(FA,*n1*)).

REF=(USER,*n1*,*n2*)
A user-written program locates and interprets the reference (see "User-written program to calculate file address for non-standard reference" on page 494).

n1
Four-character name of a user-supplied program that locates and interprets the file address.

n2
Displacement of the reference (as for **REF**=(FA,*n1*)).

ID=({C,*id*|(S,*id-symbol*|(I,*n*})
Record ID of the refer-to record; one of:

(C,*id*)
id is the record ID; specify 2 characters or 4 hexadecimal digits.

(S,*id-symbol*)
id-symbol is the symbol for the record ID.

(I,*n*)
The record ID of the refer-to record is at displacement *n* from the beginning of the item in the record that this index describes.

RCC=({P,*n*|(I,*n*|(C,*rcc*})
Check the RCC:

(P,*n*)
RCC of the first record in the refer-to group is at displacement *n* from the start of the refer-from record.

(I,*n*)
RCC is at displacement *n* from the start of the item in the refer-from record. This format is valid only if the record is divided into items.

If there are sub-items and the RCC is not the same for all sub-items within the item, RCC checking is not possible. If there are sub-items, **RCC**=(I,*n*) means that for all sub-items the RCC is at displacement *n* from the start of the item.

(C,*rcc*)
Where *rcc* is the RCC. Specify 2 hexadecimal digits (X'*nn*'), or 1 character (C'*c*'), or a decimal number (*ddd*).

GROUP=*label* | GROUP=(*ID*)
GROUP=(ID) means that the GROUP macro is to be located by the record ID that Recoup finds from the ID= parameter of the INDEX macro.

You must specify METHOD=ID in the GROUP macro before you locate the GROUP macro in this way.

If the refer-to group is a chain of records, then specify the label of the GROUP macro in this parameter. If the refer-to group is a single record that does not contain any references, omit this parameter.

item-information
Specify this if the refer-from record is divided into items. The three formats are:

For item count:

```
FI={n|fldname}
                          ,CNT={(N,n)|(n1,n2)}
                          ,LI={(N,n)|(n1,n2)}
```

or, for NAB:

```
FI=([n|(fldname}[,REVERSED])
                          ,NAB=(n1,n2)
                          ,LI={(N,n)|(n1,n2)}
```

or, for AIX/DIX:

```
FI={n|fldname}
                          ,AIX=(n1,n2),DIX=(n1,n2)
                          ,LI=(N,n)
```

Where:

FI=([*n*|*fldname*})
The refer-from record is divided into items, each of which contains a reference to a refer-to record.

n
Displacement from the start of the record to the first item.

fldname
Field name of the start of the first item. Use of the field name improves descriptor program readability.

REVERSED
Items are placed in the record in reversed order. Items are usually placed in a record so that the displacement of item *x*+1 is greater than the displacement of item *x*. Figure 120 on page 489 shows the normal order and Figure 121 on page 489 shows the reversed order. **FI**=(*n*,REVERSED) is allowed only with NAB.

CNT={(N,*n*)|(*n1*,*n2*)}
Count of the number of items in use, one of:

(N,*n*)
Item count is constant and equals *n*.

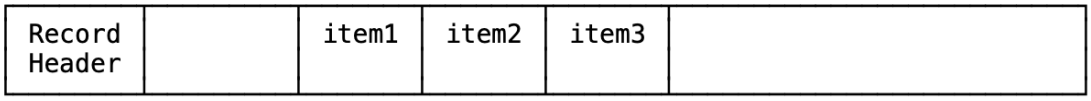


Figure 118. Example item count using a constant

Or the record contains a field that contains the item count, where:

n1
Length of the count field.

n2
Displacement from the start of the record to the count field.

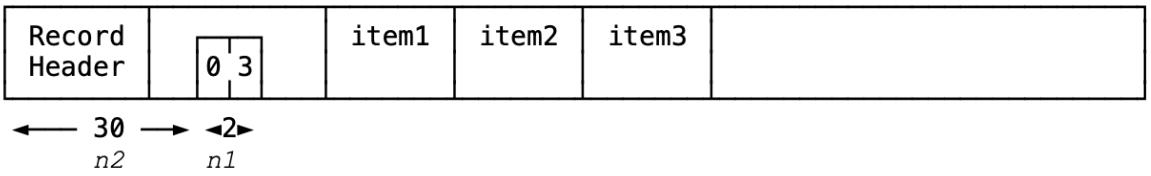


Figure 119. Example item count using a field

NAB=(*n1*,*n2*)
A next available byte field points to the first free item.

n1
Length of the NAB field.

n2
Displacement from the start of the record to the NAB field.

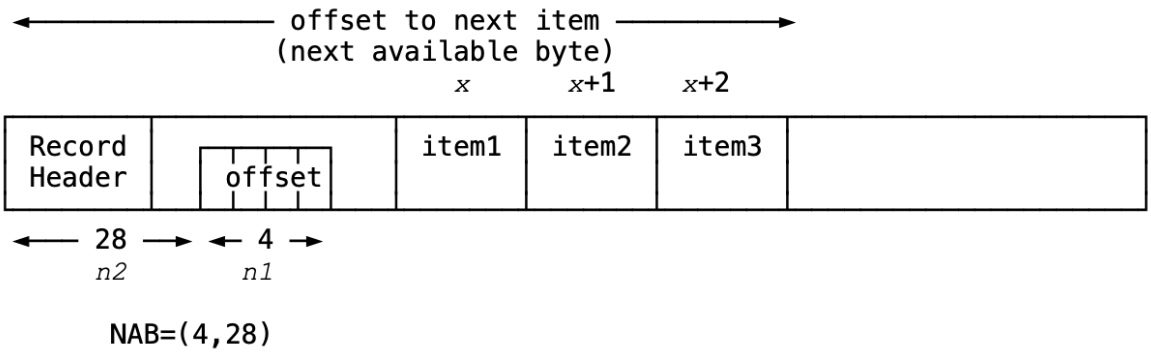


Figure 120. Use of NAB (normal order)

Items can alternatively be placed in the record in reversed order. See Figure 121 on page 489.

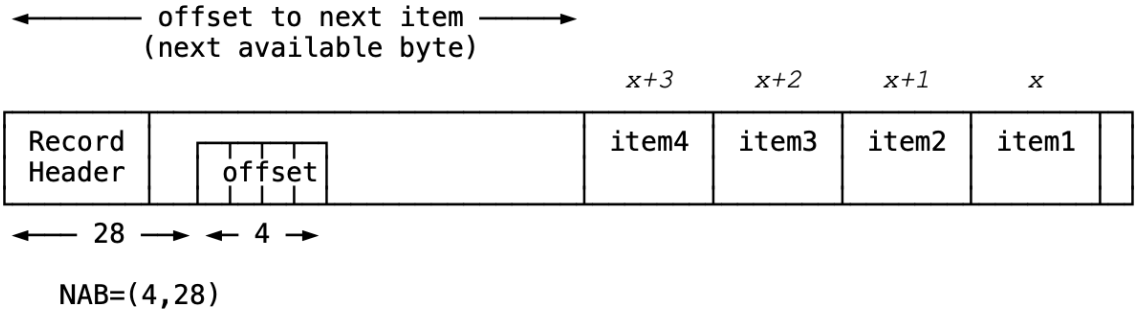


Figure 121. Use of NAB (reversed order)

AIX=(*n1*,*n2*),DIX=(*n3*,*n4*)
A delete item index field (DIX) points to the item before the first item in use. An add item index field (AIX) points to the item after the last item in use. Figure 122 on page 489 shows the use of AIX and DIX fields.

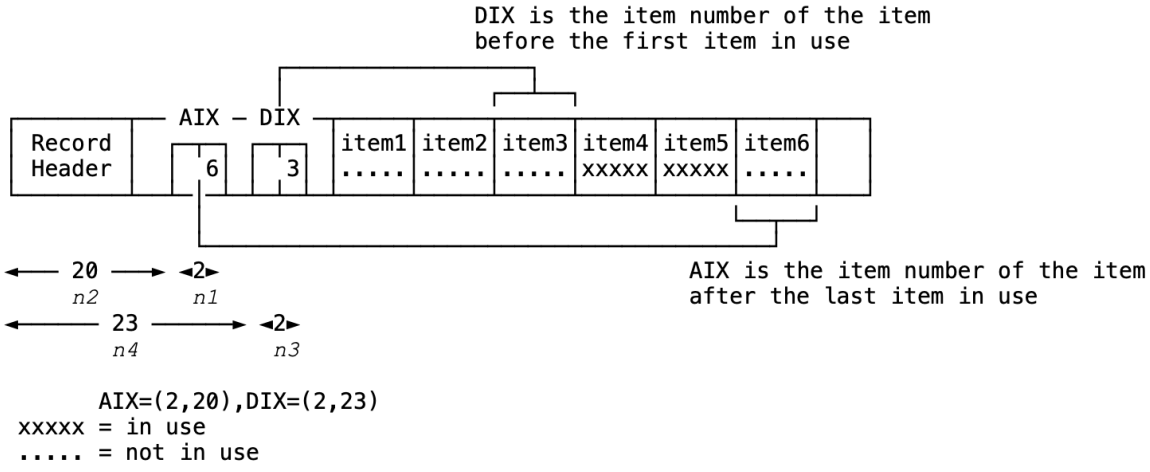


Figure 122. Use of DIX and AIX

n1
Length of the AIX field.

n2
Displacement from the start of the record to the AIX field.

n3
Length of the DIX field.

n4
Displacement from the start of the record to the DIX field.

LI={(N,*n*)|(*n1*,*n2*)}
Length of the items:

(N,*n*)
Item length is constant and equals *n*.

or (for item count or NAB only) the item contains a field that contains the item length, where:

n1
Length of the length field.

n2
Displacement from the start of the item to the length field.

subitem_information
Specify this if the items in the refer-from record are subdivided into sub-items. The format is:

```
SI=n
                          ,SIC={(N,n)|(n1,n2)}
                          ,LSI={(N,n)|(n1,n2)}
```

Where:

SI=*n*
The items in the refer-from record are subdivided into sub-items, each of which contains a reference to a refer-to record.

n
Displacement from the start of the item to the first sub-item.

SIC={(N,*n*)|(*n1*,*n2*)}
Sub-item count (that is, number of sub-items in an item).

(N,*n*)
Sub-item count is fixed and its value is *n*.

(*n1*,*n2*)
Each item contains a field that contains the number of sub-items in that item:

n1
Length of the sub-item count field.

n2
Displacement from the start of the item to the sub-item count field.

LSI={(N,*n*)|(*n1*,*n2*)}
Length of sub-item.

(N,*n*)
Length of the sub-item is fixed and its value is *n*.

(*n1*,*n2*)
Each sub-item contains a field that contains the length of the sub-item:

n1
Length of the sub-item length field.

n2
Displacement from the start of the sub-item to the sub-item length field.

INDXRTN=*label*
Label of a user-supplied subroutine. (See “User-written subroutines” on page 491.) Recoup enters this subroutine before processing this index.

ITEMRTN=*label*
Label of a user-supplied subroutine. (See "User-written subroutines" on page 491.) Recoup enters this subroutine before processing each item.

ITEMKEY=*n*
Check the 1-byte item key. *n* identifies the item type that the INDEX macro refers to; specify 1 character or 2 hexadecimal digits. Recoup checks the item-key before processing an item. If the item key in the item is not the same as the item-key for the INDEX, the item is ignored. Omit ITEMKEY if item-key checking is not required.

Code a different INDEX macro instruction for each item key.

The item key identifies the type of item and allows a record to contain several different types of item. The item key is at a fixed position in the item. If the item length is fixed, the item key is in the first byte of the item. If the item length is variable, the item key is in the third byte of the item.

PRINT={NOGEN|GEN}

NOGEN
Do not print the macro expansion

GEN
Print the macro expansion

RTP={0|*n*}
Record ID qualifier. *n* may take any integer value from 0 through 9.

User-written subroutines and programs

If you need subroutines in a descriptor program, put them at the end of the program after all the GROUP and INDEX macros.

Do not use general registers 8 through 13 (RAP, REB, RLA, RLB, RLC, RLD) in your own subroutines; they are reserved for ALCS use, as they are in other application programs. Also, general register 6 (RGE) contains the return address, and should only be used if it is saved and restored. There are no restrictions on the use of general registers 1 through 5 (RAC through RGD), 7 (RGF), and 14 through 15 (RDA, and RDB).

Space is reserved in the ECB work area for use by user-written subroutines and programs. This is area EBX064 through EBX103.

Space is reserved in the Recoup keypoint for use by user-written subroutines and programs. The BK0RP DSECT macro defines the format of the Recoup keypoint. The user area is the 40 bytes at label BK0USR.

Storage and data levels are reserved for use by user-written subroutines and programs. These are levels DE and DF.

The GROUP macro generates a DSECT for the group, and the INDEX macro generates a DSECT for the index. User-written subroutines can use labels in these DSECTs to refer to fields in the GROUP and INDEX macro expansions.

User-written subroutines

Product-sensitive Programming Interface

ALCS Recoup provides six exits for user-written subroutines. These exits allow you to specify additional processing, and they permit chain chasing of non-standard database structures.

The six installation-wide exits and the return arrangements are as follows:

ENTRTN
A routine that the descriptor processor enters before processing the prime group.

The subroutine returns to main line by:

```
        BR    R06
```

EXTRTN
A routine that the descriptor processor enters after processing the prime group. On entering the subroutine, general register 2 (RGA) points to the prime group macro expansion in the descriptor program; general register 3 (RGB) points to the Recoup keypoint.

The subroutine returns to main line by:

```
        BR    R06
```

RRECRTN
A routine that the group processor enters before reading each record in a group. On entering the subroutine, general register 3 (RGB) points to the Recoup keypoint, and information about the record to be read is in EBW000 through EBW007:

EBW000
The 2-byte record ID.

EBW002
The 1-byte record code check.

EBW004
The 4-byte file address.

To bypass reading this record and all remaining records in this group, the subroutine returns to main line by:

```
        B     0(R06)
```

To process the record, the subroutine returns to main line by:

```
        B     8(R06)
```

PRECRTN
A routine that the group processor enters after reading, but before processing, each record in a group. Use PRECRTN to bypass processing of selected records.

Your routine can modify (in memory) the record that Recoup has read from DASD. On return Recoup will use the layout specified in the GROUP macro to locate the chain fields and so on within the record **as modified by your routine**. It is safe to do this because Recoup does not write the (updated) record back to DASD.

Attention

If you modify the record contents in your PRECRTN, you cannot use ZREL0 with this record. On entering the subroutine, general register 3 (RGB) points to the Recoup keypoint; the record to be processed is on ECB level 0.

To bypass all indexes relating to this record, but process the remaining records in the group, the subroutine returns to main line by:

```
        B     0(R06)
```

To process the record, the subroutine returns to main line by:

```
        B     8(R06)
```

To bypass all indexes relating to this record, and all remaining records in this group, the subroutine returns to main line by:

```
B     16(R06)
```

INDXRTN
A routine that the index processor enters before processing the index. Use it to bypass processing of selected indexes for particular records.

On entering the subroutine, general register 1 (RG1) points to the record that the index describes; the record is on ECB level 0. General register 5 (RGD) points to the index macro expansion in the descriptor program.

To bypass the index for the record on level 0, the subroutine returns to main line by:

```
B     0(R06)
```

To process the index, the subroutine returns to main line by:

```
B     8(R06)
```

ITEMRTN
A routine that the index processor enters before processing each item; use it to bypass processing of selected items.

On entering the subroutine, general register 2 (RGA) points to the item in the record that the index describes; the record is on ECB level 0. General register 5 (RGD) points to the INDEX macro expansion in the descriptor program.

To bypass the item, the subroutine returns to main line by:

```
B     0(R06)
```

To process the item, the subroutine returns to main line by:

```
B     8(R06)
```

End of Product-sensitive Programming Interface

User-written program to calculate address of prime group record

If the GROUP macro for the prime group specifies:

```
METHOD=(USER,nnnn,...
```

the Recoup prime group processor enters the user-written program *nnnn* for each start-of-chain record. The user-written program sets up the following information for a prime group record:

EBX000
The 2-byte expected record ID.

EBX002
The 1-byte expected RCC.

EBX004
The 4-byte file address.

On entry to the user-written program, the fullword EBX016 contains the PRIMECT value specified or defaulted in the corresponding GR0UP macro. The user-written program can (optionally) replace this with a different value.

The user-written program then returns control to Recoup using the BACKC macro.

When the addresses of all start-of-chain records are returned to Recoup, the user-written program sets EBX004 (4 bytes) to hexadecimal zeros, and returns control to Recoup using the BACKC macro.

User-written programs that the prime group processor enters can use CE1URA through CE1UR7.

User-written program to calculate file address for non-standard reference

If the INDEX macro specifies:

```
        REF=(USER,mmmm,...
```

the Recoup index processor enters user-written program *mmmm* to convert a non-standard reference to a file address. The reference can be 1 to 4 bytes. The Recoup index processor sets up the reference in the low-order bytes of the 4-byte field that starts at CE1UR0, and enters the user-written program using the ENTRC macro. The user-written program sets up the 4-byte file address in EBX004 and returns control to Recoup by BACKC macro. User-written programs that the index processor enters cannot use CE1URA through CE1UR7.

Chapter 10. Implementing ALCS e-mail

Introduction

ALCS allows you to use your existing ALCS communications network for sending and receiving e-mail messages. E-mail messages can be sent between ALCS users, they can be sent to and from other employees on your corporate e-mail system, and to all users of the internet. ALCS uses the Simple Message Transfer Protocol (SMTP) for sending and receiving e-mail messages. E-mail messages can be sent using the ALCS `ZMAIL` command and they can also be sent by ALCS application programs. E-mail messages can be received by ALCS terminal users and also by ALCS application programs. When ALCS terminals are used for sending or receiving e-mail messages, they should contain only textual data (for example, ALCS terminals can not handle e-mail attachments). However e-mail messages can include characters that use Double Byte Character Set (DBCS) support.

Configuring your system for e-mail

Configuring your Intranet for ALCS e-mail

The ALCS e-mail receiver understands e-mail addresses of the form:

crn@*domain*

Where *crn* is one of:

- CRN of an ALCS terminal
- CRN of an ALCS application
- The special name 'Postmaster'

and *domain* is the ALCS mail domain name. The *crn* and *domain* are not case sensitive. The simplest choice of mail domain name is the domain name of the IP stack which your ALCS connects to, but you can use a different name if you prefer. The ALCS mail domain name resolves to an IP address (the IP address of the IP stack which your ALCS connects to), and a port number (we expect you will use port 25, which is the well-known port number for SMTP).

To attach ALCS to your corporate e-mail system, you must configure your domain name servers (DNSs) or other mail routing tables, or both, to recognize the ALCS mail domain name and route messages accordingly.

Note that if you want ALCS to send messages to and receive messages from the Internet, you will already have - or should consider getting - an e-mail gateway that connects your intranet to the Internet. This e-mail gateway may need to be configured to recognize the ALCS mail domain name.

Define TCP/IP Listener in the ALCS system configuration table

The ALCS e-mail support uses the ALCS TCP/IP Listener for receiving inbound e-mail. You must specify the port number (usually port 25) when you start the Listener. You can request ALCS to start the Listener, using a specific port number, during ALCS startup (by coding ALCS `SCTGEN` system generation parameters) or via the ALCS `ZCTCP` command. If you use the `ZCTCP` command, the format of that command is:

```
ZCTCP LISTEN,START,PORT=25
```

© Copyright IBM Corp. 2003, 2019

or:

```
ZCTCP LISTENn,START,PORT=25
```

where *n* is the number of the Listener which you want to use (1 to 8).

If you wish to activate the TCP/IP Listener during ALCS startup, you must enable ALCS TCP/IP support, including the TCP/IP Listener and port number, by including the following parameters on the SCTGEN generation macro

```
[label] SCTGEN TCPIP=(YES,[n]),TCPLIST=(YES,[n]),TCPPORT=(port_number_1,...)
```

You may also wish to include the SCTGEN TCPNAME parameter in your system configuration (this provides ALCS with the name of the TCP/IP address space that ALCS uses for its initial connection on ALCS startup). Alternatively, you can provide this via the ZCTCP command.

Because the e-mail support uses the ALCS TCP/IP Listener, there is no requirement to define any TCP/IP communication resources in the ALCS communication generation.

Define e-mail support in the ALCS system configuration table

Additional, optional SCTGEN parameters have been provided for e-mail support:

```
EMAILDOMAIN=domain_name
EMAILMTA=ip_address|domain_name
EMAILPORT=25|port_number
EMAILTIMEOUTR=30|t1
EMAILTIMEOUTS=120|t2
EMAILPOSTMASTER=ROC|crn
```

Refer to “SCTGEN macro” on page 72 for details about these parameters.

All of the these e-mail SCTGEN parameters can be set or changed by the ZMAIL SET command when ALCS is running. You could initially define each of the above e-mail parameters by using the ZMAIL SET command. Any parameters that you set or change with this command are not retained over an ALCS restart.

Note: When a domain name is specified for the EMAILMTA= operand on the SCTGEN macro, the first ZMAIL send command will invoke the conversion of the domain name to an IP address. This could cause a DXC8073E message.

Define the MAIL application in the communication generation

You must define the 'MAIL' application in your ALCS communication generation by including a COMDEF macroinstruction of the form:

```
[label]  COMDEF LDTYPE=ALCSAPPL,NAME=MAIL,PROG=MAIL,FMSG=NO
```

In this example, the name of the input message editor program is given as *MAIL*. ALCS does not pass any messages to this application, therefore you can use any program name you wish for this 'dummy' application (there are no programs called MAIL in the ALCS e-mail support).

Define the e-mail record ID in the database generation

You must define the pool record ID, *AB10*, in the database generation. Although you may use any record size for the e-mail message record, we recommend that you use a 4000-byte record (size L3). The

following is an example of how the USRDTA macro should be coded when you define the e-mail record ID in the database generation.

```
[label]  USRDTA ACTION=ADD,TYPE=L3ST,ID=AB10
```

Implement ALCS TCP/IP concurrent server exit program ATCP

The ALCS e-mail support uses the ALCS TCP/IP Listener. When the Listener is started, it waits for connections on a specific port (for e-mail, this is usually port 25). When ALCS receives a connection on that port, it creates a new ECB and passes that ECB to the TCP/IP concurrent server exit program ATCP. You must therefore implement a version of the ATCP exit program that will activate the ALCS e-mail support. Your e-mail version of ATCP should be coded with an ENTDC CSMR instruction so that it immediately passes all connection requests directly to the ALCS e-mail SMTP Receiver program.

The following is an example of the code that is required in your ATCP exit program.

```
         BEGIN NAME=ATCP,VERSION=00,AMODE=31
         SPACE 1
*========================================================================*
*        ALCS APPLICATION TCP/IP SOCKETS EXIT                            *
*========================================================================*
         SPACE 1
******** ENTRY POINT TO ATCP
         SPACE 1
ATCP     DC    0H'0'
         CLC   CE1PORT,=F'25'      IS IT PORT 25
         BNE   ATCP0100            BRANCH IF NOT
         SPACE 1
*------------------------------------------------------------------------*
*        PORT NUMBER 25 USED FOR E-MAIL                                  *
*------------------------------------------------------------------------*
         SPACE 1
         ENTDC CSMR                CALL E-MAIL
         SPACE 1
ATCP0100 DC    0H'0'
         SPACE 1
*------------------------------------------------------------------------*
*        OTHER PORT NUMBERS                                              *
*------------------------------------------------------------------------*
         SPACE 1
         EXITC ,
         SPACE 1
         LTORG ,
         SPACE 1
         FINIS ,
         SPACE 1
         END   ,
```

Sending and Receiving e-mail

After you have configured your ALCS system for e-mail, you can enhance your ALCS applications to build and send e-mail messages. In addition to this, your end users can compose and send e-mail messages from their terminals using the ZMAIL command. *ALCS Operation and Maintenance* describes the ZMAIL command. End users can also receive inbound e-mail messages addressed to their terminals.

Composing outbound e-mails in an ALCS application

Application programs can build e-mail messages and send them, depending on your configuration, to recipients within your organization (on your intranet) or to recipients on the Internet, or both.

Applications build e-mail messages in EBCDIC in storage blocks and send them by calling program CSMS (see “Send a message using SMTP - CSMS” on page 430).

Application programs should activate program CSMS via an ENTRC macro so that when they receive control back from CSMS they can check the return code and determine if the e-mail message has been successfully transmitted.

The following is an example of an ALCS application program that sends a test message to Tom Anybody at IBM UK.

```
         BEGIN NAME=ESND,VERSION=00,AMODE=31
         SPACE 1
*=====================================================================*
*        SAMPLE ALCS APPLICATION CODE FOR SENDING AN E-MAIL MESSAGE   *
*=====================================================================*
         SPACE 1
         DXCSMTM REG=R06           RFC 822 MESSAGE BLOCK
         SPACE 1
ESND     DC    0H'0'
         GETCC LEVEL=D1,SIZE=L3    GET A STORAGE BLOCK
         L     R06,CE1CR1          LOAD ADDRESS OF STORAGE BLOCK
         SPACE 1
         LA    R14,ORIG            LOAD ADDRESS FOR ORIGINATOR
         L     R15,=A(ORIGL)       LOAD MOVE-FROM LENGTH (PAD = X'00')
         LA    R00,SMTPORIG        LOAD MOVE-TO ADDRESS
         LA    R01,L'SMTPORIG      LOAD MOVE-TO LENGTH
         MVCL  R00,R14             MOVE IN ADDR FOR RETURNED MESSAGES
         SPACE 1
         MVC   SMTPLENG,=A(MSG1L)  MOVE IN MESSAGE LENGTH
         LA    R14,MSG1            LOAD MOVE-FROM ADDRESS
         L     R15,=A(MSG1L)       LOAD MOVE-FROM LENGTH
         LA    R00,SMTPTEXT        LOAD MOVE-TO ADDRESS
         LR    R01,R15             LOAD MOVE-TO LENGTH
         MVCL  R00,R14             MOVE IN MESSAGE TEXT
         SPACE 1
         FLIPC D0,D1               MOVE RFC 822 MESSAGE TO LEVEL D0
         ENTRC CSMS                SEND E-MAIL
         SPACE 1
ESND010  DC    0H'0'
         BACKC ,                   RETURN OR EXIT
         EJECT ,
*---------------------------------------------------------------------*
*        CONSTANTS                                                    *
*---------------------------------------------------------------------*
         SPACE 1
MSG1     DC    C'To: Tom_Anybody@uk.ibm.com',AL1(#CAR)
         DC    C'Subject: testing',AL1(#CAR),AL1(#CAR)
         DC    C'Test message from ALCS program ESND - please ignore'
MSG1L    EQU   *-MSG1
         SPACE 1
ORIG     DC    C'helpdesk@uk.ibm.com'
ORIGL    EQU   *-ORIG
         SPACE 1
*---------------------------------------------------------------------*
*        LITERAL POOL                                                 *
*---------------------------------------------------------------------*
         SPACE 1
         LTORG ,
         EJECT ,
         FINIS ,
         END   ,
```

Note that ALCS does not call the ECB-controlled installation-wide exit program ASM1 (see “ZMAIL message customization program - ASM1” on page 394) for messages which applications send. But it does call exit program ASM0 (see “Outgoing e-mail customization program - ASM0” on page 393). You can use ASM0 to add a standard disclaimer, information for customers, or any other text to the end of each e-mail message that ALCS sends.

If, for example, you implement an ASM1 program to add a standard disclaimer to ZMAIL messages, and ASM0 to add standard customer information to all messages, then outbound e-mail messages sent using ZMAIL will contain the standard disclaimer followed by the customer information. Outbound messages sent by applications will only contain the customer information.

You should consider including a 'Reply-to:' or 'From:' header line in messages that your applications build so that recipients will know where to send any follow-up e-mail.

You should also consider where messages that could not be delivered are returned to. For example, if your application program has sent an e-mail message to an invalid e-mail address, your e-mail gateway will want to return that message to the sender. You may not want returned e-mail messages to be sent to your application, therefore you should identify an e-mail address within your organization to which all returned messages can be sent (for example you could create an e-mail address called 'helpdesk@domain.name' that is to receive all returned messages). When you prepare your e-mail message using the DXCSMTM message format, you can tell ALCS the e-mail address to which returned messages must be sent. Place the e-mail address for returned messages in field SMTPORIG.

Receiving inbound e-mails in an ALCS application

ALCS delivers inbound e-mail messages which are addressed to ALCS applications. The input edit program specified in the application communication resource definition (in the ALCS communication generation) receives control with the inbound message in EBCDIC in a storage block attached on storage level zero (D0).

The message is in the CM1CM input message format. Intermediate blocks end with an #EOMi, The last block and the single block when only one block is needed end with an #EOMc.

The prime block is not assigned a pool file address. Pool file addresses are also not assigned to single blocks used when only one block is needed.

When email consists of more than one block, the blocks are chained using the standard forward chain field (SMTPHFCH).

Note that if you intend an ALCS application to receive inbound e-mail messages, you must provide an ECB-controlled installation-wide exit to authorize that application (see “Incoming e-mail application authorization program - ASM6” on page 401).

As with inbound messages addressed to terminals, ALCS removes some header lines and some MIME message parts before passing an e-mail message to an application. You can customize this behavior using ECB-controlled installation-wide exits, see “Incoming e-mail header selection program - ASM2” on page 394 and “Incoming e-mail MIME-part selection program - ASM3” on page 395.

Composing outbound e-mails with the ZMAIL command

Depending on your configuration, end users can send messages to recipients within your organization (on your intranet) or to recipients on the Internet, or both. End users who send messages to recipients on the Internet should be aware that:

- Recipients of the messages probably know that the message comes from your organization.

  You may want to remind your end users of any rules your organization has relating to direct communication with the general public.
- Internet e-mail is not particularly secure. That is, people other than the intended recipients may be able to read your messages.

  Your end users should avoid including confidential or other sensitive information in e-mail messages.

  For these and other reasons, organizations often add a standard "disclaimer" paragraph at the end of e-mail messages. You can use an ALCS ECB-controlled installation-wide exit (see “ZMAIL message customization program - ASM1” on page 394) to add this standard disclaimer or any other text to the end of each message prepared by an end user.
- End users will usually include their own e-mail address in messages. The normal way to do this is to include a 'From:' header line.

  The e-mail address that an end user specifies does not need to be their ALCS e-mail address - it could, for example, be the e-mail address they normally use on the corporate Lotus Notes® e-mail system.

  In some cases, an end user might want the recipient to send any 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.

- The description of the ZMAIL command in *ALCS Operation and Maintenance* includes the special syntax which end users with ALC terminals can use to enter e-mail messages in mixed case. ALCS provides an ECB-controlled installation-wide exit (see “ALC e-mail custom conversion program - ASM5” on page 396) which allows you to customize this special syntax.

Receiving inbound e-mails at an ALCS terminal

ALCS delivers inbound e-mail messages which are addressed to ALCS terminals using the ALCS unsolicited message facility. End users retrieve and display these e-mail messages using the ALCS ZSNDU command.

Inbound e-mail messages typically include a large amount of header information which is of little or no value to human recipients. This header information consists of special header lines which a network administrator can use for problem determination and so on. For example:

- 'Received-from:' header lines identify the various e-mail systems (MTAs) responsible for routing and delivery of the message.
- Special header lines identify the originator's e-mail software.
- The 'Message-ID:' header associates the message with an identification code which is unique (that is, every Internet e-mail message has a unique Message-ID).

Before it displays an inbound e-mail message at a terminal, ALCS removes header lines which are unlikely to be interesting to a human recipient. But ALCS provides an ECB-controlled installation-wide exit (see “Incoming e-mail header selection program - ASM2” on page 394) which allows you to customize this process.

An inbound message can also contain information which ALCS can not display in any meaningful way at a terminal. For example, a message might contain multimedia information (images, sounds, and so on). E-mail protocols require the originator's e-mail software to send this type of information using MIME format messages. The MIME format allows the receiving e-mail software (in this case, ALCS) to identify message parts (such as images and sounds) which require special handling.

By default, ALCS identifies and removes any message parts which it can not display at a terminal. But ALCS provides an ECB-controlled installation-wide exit (see “Incoming e-mail MIME-part selection program - ASM3” on page 395) which allows you to customize this process.

Postmaster

Both humans and e-mail processing software in MTAs can send messages to the special address 'Postmaster@*domain* ' , where *domain* is the mail domain of your ALCS system.

This special e-mail address is used mainly for messages about e-mail problems - for example messages to let you know that an MTA could not deliver a message from ALCS and could not identify the message's originator, or to let you know that a human is having problems sending e-mail to a user on your ALCS system.

By default, ALCS delivers all messages addressed to 'Postmaster' to your ALCS RO CRAS. But you can choose a different destination for these messages (see “SCTGEN macro” on page 72 and the ZMAIL command).

As with inbound messages addressed to terminals, ALCS removes some header lines and some MIME message parts before printing an e-mail message addressed to 'Postmaster'. But because these messages often report e-mail problems, ALCS includes much more of the header information than it does in messages addressed to terminals. You can customize this behavior using ECB-controlled installation-wide exits, see “Incoming e-mail header selection program - ASM2” on page 394 and “Incoming e-mail MIME-part selection program - ASM3” on page 395.

Chapter 11. Migrating from predecessor ALCS versions and releases

This section describes the steps required to migrate to ALCS Version 2 Release 4.1 (ALCS 2.4.1) from the following predecessor versions and releases:

- ALCS Version 2 Release 1.3 (ALCS 2.1.3)
- ALCS Version 2 Release 2.1 (ALCS 2.2.1)
- ALCS Version 2 Release 3.1 (ALCS 2.3.1)

This section includes activities that apply to most installations migrating to ALCS 2.4.1 and is intended to assist in the planning of each task in the migration.

"Migration from ALCS 2.2.1" on page 515 includes extra information that applies only to migration from ALCS 2.2.1. If you are migrating from ALCS 2.3.1, and you have not applied all the maintenance fixes (PTFs) applicable to ALCS 2.3.1, then you may also want to review this section.

"Migration from ALCS 2.1.3" on page 527 includes extra information that applies only to migration from ALCS 2.1.3; if you are migrating from ALCS 2.2.1 or ALCS 2.3.1 then you can ignore it.

In this section, when we refer to "your new ALCS system" or simply "your new ALCS" we mean the ALCS 2.4.1 system that you are migrating to. When we refer to "your old ALCS system" or "your old ALCS" we mean the predecessor ALCS version or release that you are migrating from.

Introduction to migration

Your new ALCS is designed to make migration from your old ALCS as simple and trouble-free as possible. In particular:

- There should be little or no change to your application programs, database, or operational procedures.
- Most of the new functions and facilities of your new ALCS are optional. You can start to exploit them immediately after cutover, or you can wait and exploit them later.
- You should be able to cutover to your new ALCS with an outage of, at most, a few minutes; it should take the same time as any other ALCS planned restart.
- You should be able to fall back to your old ALCS system, with an outage of, at most, a few minutes. (We hope you do not need to do this.)

Non-standard ALCS systems

Migration to your new ALCS requires more consideration when:

- You have programs that use unpublished interfaces
- You have modified ALCS product components.

If you have programs that use unpublished interfaces, you must decide whether to:

- Rework your programs to use published interfaces. This may be easier to do with your new ALCS because it provides new and enhanced published interfaces.
- Rework your programs to account for changed or removed interfaces. If you decide to do this, there is no guarantee that your programs will continue to work when you apply maintenance fixes or migrate to a future version or release of ALCS.

If you have modified ALCS product components, you must decide whether to:

- Remove the modification. You may find that your new ALCS includes a new generation option or other enhancement that makes your modification unnecessary.

© Copyright IBM Corp. 2003, 2019

- Exploit a published ALCS installation-wide exit. You may find that your new ALCS includes a new installation-wide exit where you can implement your customization without changing ALCS itself.
- Propagate the modification into your new ALCS. If you decide to do this, you may have to rework and reapply you change when you apply maintenance fixes or migrate to a future version or release of ALCS.

Migration overview

About this task

Migration to your new ALCS from your old ALCS includes a number of important tasks. This section gives an overview of the primary tasks to perform.

Procedure

1. **Review the new ALCS publications**

   The publications contain information on the new features in ALCS. They also provide important information for migration planning. ALCS 2.4.1 includes the following publications (see Bibliography):

   - *ALCS Concepts and Facilities*
   - *ALCS Installation and Customization* (this book)
   - *ALCS Operation and Maintenance*
   - *ALCS Application Programming Guide*
   - *ALCS Application Programming Reference - Assembler*
   - *ALCS Application Programming Reference - C Language*
   - *ALCS Messages and Codes*
   - *ALCS Program Directory*
   - *Licensed Program Specifications*

   In particular, *ALCS Concepts and Facilities* provides an overview of ALCS, and Chapter 7, “Customizing ALCS,” on page 233 and Chapter 4, “Generating ALCS,” on page 60 describe the customization and generation of ALCS.
2. **Unpack and install the ALCS shipment**

   The ALCS 2.4.1 product shipment contains program and macro libraries for your new ALCS and for the corresponding version of the IPARS application. The unpacking process is described in the *ALCS Program Directory* that comes with the ALCS shipment.
3. **Apply maintenance**

   When the installation is complete, we recommend that you apply all the maintenance fixes (PTFs) applicable to your new ALCS.

   To be confident that you can safely fall back (if necessary) to your old ALCS, we advise you to apply all the maintenance fixes applicable to your old ALCS.

   Consult your IBM representative for information on the current level of maintenance.
4. **Review ISPF panels, JCL, or other procedures**

   If you plan to use the ALCS ISPF panels, you will probably want to customize them so that (for example) they recognize and use the libraries that you use for your application program source and object modules, and so on. See Chapter 1, “Using ALCS ISPF panels,” on page 1.

   If you prefer to develop your own JCL procedures, you can use the ISPF panels to generate sample JCL.

   You may already have your own JCL or other procedures for assembling your application and your installation-wide exit routines. If so, you may need to update them to specify the high-level qualifier for your new ALCS libraries.

5. **Review and migrate ALCS customization**

   You may have customized your old ALCS by:

   - Using ALCS installation-wide exits
   - Applying SMP/E user modifications to ALCS product code

   You should review your customization before you migrate it. You may find that your new ALCS includes enhancements that remove the need for your customization.

   You should also review the new installation-wide exits that your new ALCS provides and check if you want to exploit them.

   It is relatively easy to migrate installation-wide exit routines to your new ALCS. In many cases the migration requires little or no code changes. You must reassemble them using the ALCS 2.4.1 macros.

   "ALCS customization" on page 504 discusses this topic in more detail.
6. **Review new and changed ALCS application interfaces**

   You can run your existing application programs on your new ALCS without any reassembling, recompiling, or relink-editing.

   However, you may want to change some of your programs to exploit new features of your new ALCS.

   "ALCS application program interfaces" on page 508 discusses this topic in more detail.
7. **Generate your new ALCS system**

   Your new ALCS provides a generation function very similar to that for your old ALCS.

   You can use your existing generation input without modification.

   You may want to modify your existing generation input to exploit some new features and options of your new ALCS, especially as some of the new options remove the need to migrate user modifications and installation-wide exit routines.

   "ALCS generation" on page 509 discusses this topic in more detail.
8. **Test your new ALCS system**

   You will probably want to perform a variety of tests on your new ALCS system before you use it in production.

   "Testing your new ALCS system" on page 514 discusses this topic in more detail.
9. **Cutover to your new ALCS system**

   The actual cutover process is similar to a normal ALCS planned shutdown and restart.

   "Cutover and fallback" on page 515 discusses this topic in more detail.

Coexistence of your old and new ALCSs

While you install and test your new ALCS, you need to continue running your old ALCS production and test systems. ALCS does not place any restrictions on this. You can run your new ALCS systems and your old ALCS systems on the same MVS image - assuming, of course, that:

- Your MVS system has the appropriate program versions and releases installed.
- Your processor is fast enough, has enough memory, and so on to run the systems together.

Libraries

You must install your new ALCS into separate product libraries, and you may want to use separate libraries for other things, for example:

- You may want to have separate libraries for your configuration-dependent tables. These tables (load modules) are not compatible between your new and your old ALCS.
- You may also want to have separate libraries for the ALCS generation input.

Note that you do not need separate libraries for generation input that you do not change.

- You may want to have separate libraries for your installation-wide exits. This is especially likely if you plan to exploit new exits and interfaces provided with your new ALCS.
- You may want to have separate libraries for your application programs that you intend to change during migration to your new ALCS.

  Note that you do not need separate libraries for application programs that you do not change, reassemble, or recompile. Your new ALCS can use application program load modules from your old ALCS.

Database

In this section, we use "database" to refer collectively to:

Real-time database data sets
General files (general data sets)
Configuration data sets.

While you are testing your new ALCS, you may want to have separate databases. However if you wish, you can use the ALCS test database facility to share the same test *database* between your new and your old ALCS test systems, but you need separate test *data sets*.

When you cutover your production system to your new ALCS, you use your existing database without change; you do not need to reorganize your real-time database or add new fixed-file records. Your new ALCS does not make incompatible changes to your database (unless you tell it to); if you need to fall back, your old ALCS can still use the database.

Sequential files

ALCS does not support sharing of output sequential files. You will probably want to use different data set names for output sequential files on your new ALCS.

Application sequential files

Your new ALCS uses the same formats as your old ALCS for application real-time and general sequential files.

Database update log

If you need to restore the real-time database, your new ALCS can use database update log data sets from your old ALCS.

Provided that you apply all the maintenance fixes (PTFs) applicable to your old ALCS, your old ALCS can use database update log data sets from your new ALCS. If you are in doubt about the applicable maintenance, consult your IBM representative.

Diagnostic file

The content of the diagnostic file that your new ALCS produces is not the same as your old ALCS - it contains additional information. We recommend that when you are processing ALCS diagnostic files, you use the diagnostic file processor (DXCDTP) that comes with the ALCS which produced the diagnostic files.

ALCS customization

ALCS installation-wide exits

ALCS implements many installation-wide exits that let you customize your ALCS system. If you use any of the installation-wide exits in your old ALCS, you may want to migrate them to your new ALCS. ALCS provides two types of installation-wide exit:

Monitor
These programs execute as extensions to the ALCS online monitor program. They cannot use the general-use programming interfaces and sometimes use product-sensitive interfaces.

ALCS normally uses the MVS CALL/RETURN linkage mechanism to call installation-wide exits. Usually ALCS calls these programs conditionally; that is, ALCS tests if the program exists and calls it only if the program exists.

ECB-controlled
These programs are similar to any ALCS application program. They can use the general-use programming interfaces and sometimes use product-sensitive interfaces.

ALCS normally calls ECB-controlled installation-wide exits using the normal ENTER/BACK linkage mechanism. Usually ALCS calls these programs conditionally; that is, ALCS tests whether or not the program exists and only calls it if it does.

Chapter 7, “Customizing ALCS,” on page 233 contains a description of every installation-wide exit and ALCS service. Please review this chapter prior to migrating your exit routines.

Installation-wide monitor exits

Full details of ALCS installation-wide monitor exits are in “Implementing installation-wide monitor exits” on page 237.

Installation-wide monitor exits - new

Your new ALCS implements the following new installation-wide monitor exits:

USRMQB0
Allows you to identify the originating terminal, for messages received by the MQ Bridge. This exit replaces installation-wide ECB-controlled exit AMQ0 in your old ALCS.

USRMQB1
Allows you to remove any special headers, for messages received by the MQ Bridge. This exit replaces installation-wide ECB-controlled exit AMQ1 in your old ALCS.

USRMQB2
Allows you to add any special headers, for messages sent by the MQ Bridge. This exit replaces installation-wide ECB-controlled exit AMQ2 in your old ALCS.

USRMQB3
Allows you to add or modify information in the MQ message descriptor, for messages sent by the MQ Bridge. This exit replaces installation-wide ECB-controlled exit AMQ3 in your old ALCS.

Installation-wide monitor exits - changed

Your new ALCS changes the following installation-wide monitor exits:

USRCOM2
Two changes:

- In previous releases of ALCS, the input message was always in a storage block. Your new ALCS allows the input message to be in an area of heap storage when the large message facility is used.
- This exit can test new fields in the communication resource data in order to detect high input message rates from automated terminals, and to restrict the number of messages that are passed to an application.

USRCOM4
In previous releases of ALCS, the output message was always in a storage block. Your new ALCS allows the output message to be in an area of heap storage when the large message facility is used.

USRDCR1
In previous releases of ALCS, the message length for TCP/IP input and output messages was always in a 2-byte field. Your new ALCS allows the message length to be in a 4-byte field when the large message facility is used.

USRGFS
New entry conditions added for 64-bit addresses when VFA buffers are above the bar.

USRLOG
New entry conditions added for 64-bit addresses when VFA buffers are above the bar.

USRPIDC
New entry conditions added for 64-bit addresses when VFA buffers are above the bar.

USRTCP2
New entry and return conditions added for 64-bit addresses when the large message facility is used.

USRTCP3
In previous releases of ALCS, the message length for TCP/IP output messages was always in a 2-byte field. Your new ALCS allows the message length to be in a 4-byte field when the large message facility is used.

USRVFA
New entry conditions added for 64-bit addresses when VFA buffers are above the bar.

ALCS services for installation-wide monitor exits - new

Your new ALCS implements the following new service for installation-wide monitor exits:

USTRECB
Validate storage belonging to an ECB.

Installation-wide ECB-controlled exits

You can migrate ECB-controlled installation-wide exit programs without change, provided that they use only general-use programming interfaces. If they use other programming interfaces, including product-sensitive interfaces, then you must review them before migration. IBM recommends reassembly in these cases. See “Implementing installation-wide ECB-controlled exits” on page 359.

Exceptions are the MQ Bridge exit programs AMQ0, AMQ1, AMQ2, and AMQ3; you must migrate any functionality from these exits into the new installation-wide monitor exits USRMQB0, USRMQB1, USRMQB2, and USRMQB3 instead.

Installation-wide ECB-controlled exits - new

Program names

The program names for the new exits in ALCS may already be used by your installation for application programs which run under your old ALCS. This should not occur if you avoid the names that IBM reserves (all program names which start with A, B, or C). If you have a duplicate program name, you should change the name of that application program to one that does not begin with A, B, or C.

Your new ALCS implements the following new installation-wide ECB-controlled exit:

ATH1
Allows you to modify processing of the ALCS throttle.

Installation-wide ECB-controlled exits - removed

Your new ALCS removes the following installation-wide ECB-controlled exits:

AMQ0
Replaced by installation-wide monitor exit USRMQB0.

AMQ1
Replaced by installation-wide monitor exit USRMQB1.

AMQ2
Replaced by installation-wide monitor exit USRMQB2.

AMQ3
Replaced by installation-wide monitor exit USRMQB3.

ALCS system enhancements

Your new ALCS contains many new functions that have been included at the request of ALCS users. Some of these functions will replace system enhancements that you may have applied to your old ALCS. Although you may have used installation-wide exits to enhance the functionality of your old ALCS, you may also have introduced system enhancements in two other ways:

1. You may have applied user modifications to the base ALCS product (through SMP/E) to provide system functions which cannot be obtained through installation-wide exits.
2. You may have written additional ECB-controlled programs which provide system functions (these programs may be indirectly activated by an installation-wide exit).

This section provides guidance on the migration of these system enhancements.

User modifications

You may have applied user modifications to:

ALCS monitor programs (CSECTs)
ALCS ECB-controlled programs
ALCS macros
ALCS offline programs.

You should not migrate these modifications unless they are absolutely necessary. Your new ALCS contains new functions and new installation-wide exits designed to reduce the need for user modifications.

You should carefully review every modification you applied to your old ALCS. If some require migration, take care in selecting where they should be applied in your new ALCS; during development of your new ALCS, we heavily modified some functional areas of ALCS. If you are in any doubt about the need to migrate a user modification or where to apply it in your new ALCS, please request assistance from IBM.

ECB-controlled system functions

You can migrate ECB-controlled programs without changing them, provided that they only use general-use programming interfaces. You may have added some ECB-controlled programs to your old ALCS to provide system functions, for example:

- Programs that add operator commands - called by the ECB-controlled exit programs ACM0 or ACM1.
- Other ECB-controlled exit programs.
- Utility programs that access and process system data - for example, the ALCS Resource Control Record (RCR).

These programs may use product-sensitive programming interfaces or interfaces that are not intended for customer use. If they use interfaces that are not general-use programming interfaces, you must carefully review the programs before migrating them. In particular:

- Your new ALCS contains many operational enhancements. These include new operator commands and new parameters on existing commands. They are described in *ALCS Operation and Maintenance*.

When you decide to migrate ECB-controlled programs that provide system functions, try assembling them against your new ALCS MAC1 macro library. If no assembly errors occur, the programs probably do not use product-sensitive interfaces.

If assemblies fail with unresolved macro calls, prepare a list of the missing macros and check them against your new ALCS MAC4 library directory. Although you should not use the MAC4 library for assembling application programs, ECB-controlled programs that perform system functions may require access to the MAC4 library.

ALCS application program interfaces

Your new ALCS is designed to run application programs from your old ALCS without change.

Provided that your application programs use only general-use programming interfaces of ALCS then you should not need to change any source code, or reassemble or recompile, or relink-edit your application programs when you move to your new ALCS. But you may need to be aware of the new and changed application interfaces described in this section.

If you change applications to exploit new facilities and interfaces of your new ALCS, the programs probably will **not** work with your old ALCS.

Even if you only reassemble or recompile application programs with your new ALCS libraries they may not work with your old ALCS.

If you do change, reassemble or recompile, or relink-edit application programs with your new ALCS libraries, you should use separate source, object, and load libraries. You need to retain your old members for use in case of fall back to your old ALCS.

AM0SG macro and <c$am0sg.h> header file

Allows you to send and receive large messages where the message length occupies a 4-byte field.

CM1CM macro and <c$cm1cm.h> header file

Allows you to send and receive large messages where the message length occupies a 4-byte field.

CO0RE macro

Allows you to detect abnormally high input message rates from a terminal, and to reduce the number of input messages which are passed to an application. The communication resource data includes fields RETDMTC, RETDMTD, and RETDMTT which installation-wide monitor exit USRCOM2 can refer to or modify for this purpose.

Your new ALCS changes the length of the system area of ALCS communication table entries for TCP/IP connections. This length is defined by the symbol REIPLEN in CO0RE DSECT. If you have written any installation-wide exits or other user code which refers to this symbol, then you must reassemble the exits or user code using the ALCS 2.4.1 macros.

DCLOG macro

Allows you to collect data about large messages where the message length occupies a 4-byte field.

RC0PL macro and <c$rc0pl.h> header file

Allows you to send and receive large messages where the message length occupies a 4-byte field.

SENDC macro

Allows you to send large messages using SENDC X.

THRTC macro

Allows you to control resources used by batch-like applications.

TPFDF special usermod

TPFDF PUT level 15 introduced Common Entry Point (CEP) that used data level independence (DLI) for TPFDF applications by saving and restoring the data levels.

TPFDF PUT level 18 introduced changes to TPFDF to use DECBs rather than data levels. With this there is no need for DLI which becomes an overhead, therefore your new ALCS removes the DLI code from the TPFDF CEP support.

If you use TPFDF, and your TPFDF is at the pre-DECB but post-CEP levels (PUT levels 15, 16, or 17), then you must install special usermod D500063 for ALCS 2.4.1.

You can request this usermod via Worldwide ALCS Technical Support.

ECB-controlled callable services

New callable services

Your new ALCS adds the following ECB-controlled callable-service programs:

CVEP, CVEQ
An application program can issue ENTRC to CVEP or CVEQ to check the availability of long-term or short-term pool.

Heap and stack storage

In your new ALCS, all heap storage is allocated either in type 1 storage units (if the heap storage fits in an entry storage block), or in type 3 storage units (if heap storage does not fit in an entry storage block). Stack storage for high level languages continues to be allocated in type 2 storage units. If you have any high-level language programs which allocate heap storage then you should review and adjust:

- the number of type 2 and type 3 storage units specified on the SCTGEN NBRSU parameter
- the size of type 2 and type 3 storage units specified on the SCTGEN SUSIZE parameter
- the entry storage limits specified on the SCTGEN ENTSTOR2 and ENTSTOR3 parameters

SLIMC monitor-request macros and `slimc` C functions in your application programs might also need to be modified to handle the change in heap storage allocation.

ALCS generation

Your new ALCS has a generation function that is very similar to the one your old ALCS has.

You can use the same *input* for your new ALCS generation as you are using for your old ALCS generation, but you must run a new generation for your new ALCS because:

- The *output* from the new ALCS generation is different.

  The formats of the control blocks that the generation creates are different even if you do not change your configuration. Normally you will want to avoid changing any details of your existing configuration until you are sure you no longer need to fall back to your old ALCS.

You may also want to change your generation input because:

- There are some new optional parameters that you may want to specify to exploit new features of ALCS.

  Many of the new optional parameters *add* information to your configuration. They do not change existing configuration information. This means you can still fall back to your old ALCS system.

The following sections describe the new optional parameters that you might want to use.

For each generation macro described below, refer to Chapter 4, “Generating ALCS,” on page 60 which gives full descriptions of the parameters for each macro.

SCTGEN generation macro

Your new ALCS adds or changes the following parameters of the SCTGEN macro:

AMODE64
Specify whether or not VFA buffers are above the bar.

DDTIME
Specify a time interval for suppressing duplicate system error messages.

TPFDF
Specify whether the application program name or the TPFDF program name is stored in the last-file control information for a record.

COMDFLT and COMDEF generation macros

Your new ALCS adds the following parameters to the COMDFLT and COMDEF macros:

IPMGSZ
Specify the maximum input or output message size for large messages on a TCP/IP connection.

VFA above the bar

Your new ALCS allows VFA buffers to be allocated in virtual storage above the bar [13], which can help to remove memory constraints and free memory for use by applications and other subsystems.

If you specify SCTGEN AMODE64=VFA in your system configuration, then ALCS will obtain page-fixed virtual storage for VFA buffers above the bar. If you do not specify SCTGEN AMODE64=VFA in your system configuration, then ALCS will obtain virtual storage for VFA buffers above the line but below the bar.

If your VFA buffers are above the bar, then you must:

- Ensure that enough virtual and real storage is available.
- Specify a MEMLIMIT for the ALCS job or started task.
- Review your installation-wide monitor exits USRGFS, USRLOG, USRPIDC, and USRVFA and modify them if necessary.

ALCS performance monitor

Your new ALCS includes the IBM performance monitor that was added to ALCS 2.3.1 by PTFs UK15115 and UK24579.

If you did not use the IBM performance monitor with your old ALCS but you used a predecessor version instead (also known as the JPM), and you plan to use the IBM performance monitor with your new ALCS, then you must:

- remove the predecessor version of the performance monitor
- apply PTFs UK15115 and UK24579
- run the sample program CPM0 to convert any performance history records to the format required by the IBM performance monitor

before you migrate to your new ALCS.

[13] In the MVS 64-bit address space, a virtual line called the bar separates storage below the 2-gigabyte address (below the bar) from storage above the 2-gigabyte address (above the bar). In the 31-bit address space, a virtual line separates storage below the 16-megabyte address (below the line) from storage above the 16-megabyte address (above the line).

ALCS throttle

Your new ALCS includes the IBM throttle to control resources used by batch-like applications.

If you did not use the IBM throttle with your old ALCS but you used a predecessor version instead (part of the JPM), and you plan to use the IBM throttle with your new ALCS, then you must:

- remove the predecessor version of the throttle
- reassemble all programs that contain a THRTC macro
- add the throttle applications to your system with the help of the ZCTHR command.

ALCS communications

Your new ALCS implements some enhancements and new facilities for communications.

Large messages

ALCS allows application programs to send and receive messages up to 2 megabytes in size using the **large message** facility. The large message facility is available for messages to and from ALC terminals connected through TCP/IP, and for messages to and from other TCP/IP communication resources, with the following restrictions:

- Unsolicited messages to terminals are not allowed.
- Messages to terminals owned by another system are not allowed.
- ALCS scroll logging is not allowed.
- ALCS message retrieval is not allowed.
- Messages must not exceed any maximum size imposed by the application layer protocol (for example, MATIP or host-to-host).

Application programs can send large output messages using the SENDC X or ROUTC (`routc`) service. The message must be in heap storage which the program has obtained using MALOC, CALOC, and RALOC monitor-request macros, or `malloc`, `calloc`, and `realloc` C functions.

ALCS passes large input messages to application programs in heap storage which the program must release using the FREEC monitor-request macro or `free` C function.

The following table summarizes the difference between the application programming interfaces for normal and large output messages.

Normal output messages	**Large output messages**
Use SENDC A|C|L|M	Use SENDC X
Use ROUTC (`routc`) with the RCPLOEXT indicator set off in the RCPL	Use ROUTC (`routc`) with the RCPLOEXT indicator set on in the RCPL
Message is in a storage block attached on ECB storage level	Message is in heap storage with address in ECB data level
Message is in standard AMSG or OMSG format (2-byte length field)	Message is in extended AMSG or OMSG format (4-byte length field)
ALCS releases the storage block containing the message	ALCS does not release the heap storage containing the message

The following table summaries the difference between the application programming interfaces for normal and large input messages.

Normal input messages	Large input messages
Message is in a storage block attached on ECB storage level	Message is in heap storage with address in ECB data level
Message is in standard AMSG or OMSG format (2-byte length field)	Message is in extended AMSG or OMSG format (4-byte length field)

When defining your ALCS system configuration, the system programmer specifies the number and size of type 3 storage units used for assembler and C/C++ heap storage (see the description of the SCTGEN macro in *ALCS Installation and Customization*).

When defining your TCP/IP connections to ALCS, the system programmer specifies whether or not the TCP/IP connection can support large messages (see the description of the COMDEF macro for a TCP/IP connection in *ALCS Installation and Customization*).

For details of the AMSG and OMSG message formats and the RCPL contents, see *ALCS Application Programming Guide*.

If you use the large message facility then you must review your installation-wide monitor exits USRDCR1, USRCOM2, USRCOM4, USRTCP2, and USRTCP3 and modify them if necessary.

Detect and throttle input messages

Your new ALCS allows you to detect abnormally high input message rates from a terminal, and reduce the number of input messages which are passed to an application. The communication resource data mapped by CO0RE DSECT includes new fields which installation-wide monitor exit USRCOM2 can refer to or modify for this purpose.

TCP/IP trace

In your old ALCS, TCP/IP trace records are automatically written to the diagnostic sequential file. Your new ALCS provides a choice of destinations for the TCP/IP trace data:

- 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

  By default, your new ALCS does not write TCP/IP trace records to the diagnostic file. Use the ZCTCP command to start and stop doing so.
- 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.

MQ Bridge

Your new ALCS optimizes the MQ Bridge by moving routines which were previously in ECB-controlled programs CMQC, CMQD, CMQQ, CMQR, CMQS, and CMQT into the ALCS monitor.

As a result, your new ALCS implements the MQ Bridge exits differently from your old ALCS.

Your old ALCS included installation-wide exit programs AMQ0, AMQ1, AMQ2, and AMQ3 for implementing custom processing for MQ Bridge messages. In your new ALCS, these have been replaced with new installation-wide monitor exits USRMQB0, USRMQB1, USRMQB2, and USRMQB3.

If you use the MQ Bridge function then you must migrate any functionality from the ECB-controlled exit programs into the new monitor exits.

You do not need to migrate any printer acknowledgement (ACK) processing from AMQ1 to USRMQB1, because your new ALCS handles ACKs automatically. You can optionally use installation-wide monitor exit USRCOM8 to take any additional actions required when ALCS receives a message from an MQ printer.

End users, operations, and programmed operators

Your new ALCS implements some changes and some new facilities that may affect:

- Human end users
- Programmable devices that simulate end users ("screen scrapers")
- Operators
- Programmed operators (for example, NetView).

You will probably want to tell your end users and operators about these changes. And you may want to investigate the impact (if any) on programmable devices and programmed operators. Whether or not you identify any impact on these "end users" you will probably want to test that they work as expected with your new ALCS.

New and enhanced ALCS commands

Your new ALCS introduces a new command and enhances several others. Your operations staff and end users may need an opportunity to familiarize themselves with these new and extended commands. They include:

ZCTCP
Enhanced to control and display TCP/IP trace information.

ZPCTL
Enhanced to display additional information.

ZDSER
Enhanced to display additional information.

ZCTHR
Implements the ALCS throttle.

ALCS commands, including new and enhanced commands, are described in *ALCS Operation and Maintenance*.

Messages and codes

If you have programmed devices that simulate end users or if you have programmed operators, you should check if they are impacted by changes to ALCS messages and codes. These are detailed in *ALCS Messages and Codes*.

Testing your new ALCS system

You will almost certainly want to do some testing before you cutover to your new ALCS. The amount of testing, and the testing methods, vary between our customers. Probably you already have established testing procedures that you used when you installed or migrated to your old ALCS, and testing procedures that you use when you install maintenance (APAR fixes and PTFs) on your existing system.

If you would like advice or help with your ALCS testing, please contact your IBM representative.

In general IBM recommends that you test everything that you change during the migration. In particular:

Customization

We recommend that you pay particular attention to testing installation-wide exits. Remember that installation-wide monitor exit code runs authorized; it can compromise the integrity and security of MVS itself.

User modifications

We strongly recommend that you avoid modifying ALCS components. If you must modify ALCS components, we recommend that you test your modifications thoroughly. Remember that parts of ALCS run authorized; changes can compromise the integrity and security of MVS itself.

Security

If you use ALCS's secure access control, you should test that the security characteristics of your end users, the resources that you protect, and so on, are correctly defined to your external security manager.

Network

If you have user modifications or installation-wide exits for communication or printer support, we recommend that you perform detailed network testing. We also recommend that you test any programmable devices that connect to your ALCS and simulate human end users (using "screen scraping" or similar methods).

Applications

You should not need to make significant changes to your existing application programs. But we recommend that you do at least some testing of your existing application before you cutover to your new ALCS.

Stress testing

Under high load, for example during high transaction rates or when utility functions are running, your application might behave differently under your new ALCS from the way it behaves under your old ALCS. We recommend that you try to reproduce these high load conditions during your testing. You might, for example, use TPNS for this.

Recovery and fallback

We recommend that you review, and if possible test, your existing recovery and fallback procedures. These might include procedures for:

- Fallback to your old ALCS system
- Database restore
- Manual database repair
- Off-site disaster recovery

Operations

We recommend that you allow your operations staff to verify that their existing procedures work as expected, and to learn any new or changed procedures. The testing period for your new ALCS can be an opportunity to do this.

End users

We recommend that you allow your end users to verify that their existing procedures work as expected, and to learn any new or changed procedures. The testing period for your new ALCS can be an opportunity to do this.

Trial cutover

You may want to perform a trial cutover of your production system to your new ALCS. You can switch to your new ALCS for a short period of time and then switch back to the old ALCS. This gives the opportunity to verify your new ALCS system against the production system environment. "Cutover and fallback" on page 515 discusses this in more detail.

Cutover and fallback

When your testing is complete, the cutover to your new ALCS system requires a very short outage. The typical cutover process is very similar to a normal planned restart of ALCS. That is:

1. While your old production ALCS system is running, start your new ALCS (using the same database, network, and so on) and bring it to STANDBY state.

   If you normally run with a standby ALCS, you now have both a standby new ALCS and a standby old ALCS.
2. Halt your old production ALCS.
3. Bring your standby new ALCS to NORM state.

You will probably want to keep your old ALCS as an alternate (standby) until you are confident that you will not need to fall back to your old ALCS. The fallback process is almost the same as the cutover process:

1. If you do not already have your old ALCS in STANDBY state, start your old ALCS (using the same database, network, and so on) and bring it to STANDBY state.
2. If your new production ALCS is still running, halt it.
3. Bring your standby old ALCS to NORM state.

Migration from ALCS 2.2.1

PrintView

Your old ALCS includes the OS/2 Presentation Manager function PrintView/2 which emulates a printer terminal (for example, an ALCS RO CRAS).

Your new ALCS does not include the PrintView/2 function and does not include the library with the final qualifier DXCWSTN (which contained PrintView/2). Instead, Appendix A, "Sample code for installation-wide exit program APR5," on page 543 shows sample code for ECB-controlled installation-wide APR5. You can use APR5 to copy RO CRAS output to an alternative destination, for example a sequential file or a TCP/IP connection.

Installation-wide exits

Installation-wide monitor exits - new

Your new ALCS implements the following new installation-wide monitor exits:

USRDCR2
Allows you to modify the length of the ECB user data collection area to be written. (This exit was added to ALCS 2.3.1 by PTF.)

USRDCR3
Allows you to select an alternative destination for writing data collection records, or to prevent the records from being written. (This exit was added to ALCS 2.3.1 by PTF.)

USRMQI3
Allows you to take action when the MQ interface fails to return control to ALCS. (This exit was added to ALCS 2.3.1 by PTF.)

USRSAF
Allows you to extract installation-defined data from your external security manager. (This exit was added to ALCS 2.3.1 by PTF.)

USRSQL3
Allows you to change the application plan name for SQL calls from this entry, if required. (This exit was added to ALCS 2.3.1 by PTF.)

USRWAIT
Allows you to detect and terminate an entry which exceeds one of the entry read thresholds. (This exit was added to ALCS 2.3.1 by PTF.)

Installation-wide monitor exits - changed

Your new ALCS changes the following installation-wide monitor exit:

USRLOG
In previous releases of ALCS, this exit is called for records retrieved by ECB-controlled programs but not for records retrieved by ALCS for its own purposes (system records - these are ALCS system fixed file records, and application pool records containing updated release file information). In ALCS 2.4.1 this exit is called for all record retrievals, including system records.

The entry conditions for system records are different from the entry conditions for records retrieved by ECB-controlled programs. If:

- You **have** implemented the USRL0G installation-wide exit, and
- You **have not** modified the ALCS 2.2.1 monitor CSECT DXCLOG so that it calls USRL0G for system records (as described in APAR AQ22386, PTF UQ38933)

then you must add code at the start of the exit to determine which routine activated it.

An installation-wide monitor exit shell is provided with ALCS 2.4.1 in the installation-wide exit library. This exit shell, which is called DXCUL0G, contains sample code for handling the different entry conditions.

If you have modified the ALCS 2.2.1 monitor CSECT DXCLOG so that it calls USRL0G for system records then you do not need to change your existing version of USRL0G and you do not need to modify the ALCS 2.4.1 monitor CSECT DXCLOG.

ALCS services for installation-wide monitor exits - new

Your new ALCS implements the following new service for installation-wide monitor exits:

UMLEVVAL
Validate an ECB monitor storage level. (This service was added to ALCS 2.3.1 by PTF.)

Installation-wide ECB-controlled exits - new

Your new ALCS implements the following new installation-wide ECB-controlled exits:

ACME
Allows you to process unknown parameters on the ZACOM command. (This exit was added to ALCS 2.3.1 by PTF.)

AMQR
Allows you to implement custom processing for the ZCMQI CONNECT command. (This exit was added to ALCS 2.3.1 by PTF.)

AOCM
Allows you to modify processing of the online communication table maintenance police facility.

APF1, APF2
Allows you to modify processing of the ALCS performance monitor. (These exits were added to ALCS 2.3.1 by PTF.)

APM2
Allows you to suppress messages from the long-term pool monitor.

APRB, APRC
Allows you to implement custom processing for the printer queue threshold. (These exits were added to ALCS 2.3.1 by PTF.)

ARO1
Allows you to restrict the use of ZROUT and ZACOM APPL commands. (This exit was added to ALCS 2.3.1 by PTF.)

ASD1
Allows you to format any installation-defined security data, before it is displayed by the ZDCOM command. (This exit was added to ALCS 2.3.1 by PTF.)

Installation-wide ECB-controlled exits - changed

Your new ALCS changes the following installation-wide ECB-controlled exits:

ACE1
New entry condition allows you to implement custom processing when an SNA printer is disconnected. (This exit was changed in ALCS 2.3.1 by PTF.)

AMQ0
New return condition allows you to adjust the length of an input message received on the MQ Bridge. (This exit was changed in ALCS 2.3.1 by PTF; in ALCS 2.4.1 it is replaced by installation-wide monitor exit USRMQB0.)

ATCP
Changes for multiple TCP/IP concurrent servers (Listeners). If your ALCS system uses the ALCS e-mail support, this exit will be routing inbound e-mail messages to ALCS. Migrate the functionality in ATCP to the new installation-wide exit shell provided with your new ALCS in the installation-wide exit library.

Trace

Conversational trace - SUBSTEP and RUNSTOP trace control commands

Your new ALCS includes new conversational trace control commands for loop and subroutine detection. (These commands were added to ALCS 2.3.1 by PTF.)

Conversational trace - GET, REL, and FLIP trace control commands

Your new ALCS includes conversational trace control commands to:

- Get a storage block and attach it to a storage level
- Release a storage block from a storage level

- Exchange the contents of two storage and data levels

(These commands were added to ALCS 2.3.1 by PTF.)

Conversational trace - AUT and SBx options

Your new ALCS includes new conversational trace options to display and set the contents of attached storage blocks and automatic storage blocks. (These options were added to ALCS 2.3.1 by PTF.)

Conversational trace - ECB prefix field options

Your new ALCS includes new conversational trace options to display and set the contents of fields in the ECB prefix. (These options were added to ALCS 2.3.1 by PTF.)

ALCS application program interfaces

APIDC macro

ALCS performance monitor interface request macro. (This macro was added to ALCS 2.3.1 by PTF.)

AUTHC macro

Allows you to access installation-defined security data. (This macro was changed in ALCS 2.3.1 by PTF.)

CALOC macro

Allows you to reserve and initialize a heap storage area. (This macro was added to ALCS 2.3.1 by PTF.)

DEQC macro

Allows you to specify conditional dequeue. (This macro was changed in ALCS 2.3.1 by PTF.)

FREEC macro

Allows you to release a heap storage area. (This macro was added to ALCS 2.3.1 by PTF.)

MALOC macro

Allows you to reserve a heap storage area. (This macro was added to ALCS 2.3.1 by PTF.)

MAP3270 macro

Your new ALCS includes the new UPPERCASE parameter on the MAP3270 macro. For 3270 screen maps, this allows you to choose whether or not to translate input alphabetic characters to uppercase.

RALOC macro

Allows you to change the size of a heap storage area. (This macro was added to ALCS 2.3.1 by PTF.)

RIDIC macro

Allows you to get information about fixed file as well as short-term pool file and long-term pool file records. (This macro was changed in ALCS 2.3.1 by PTF.)

SLIMC macro

Specify maximum entry storage according to storage type, and specify entry read threshold. (This macro was changed in ALCS 2.3.1 by PTF.)

WTOPC macro

Your new ALCS includes:

- The new MFORMAT parameter for AMSG, OMSG, and IMSG message formats.
- The new HEXBA substitution type for compatibility with TPF and TPFDF.

(This macro was changed in ALCS 2.3.1 by PTF.)

ECB-controlled callable services - changed

Your new ALCS changes the following ECB-controlled callable-service program:

CSMS
An application program can enter CSMS to send a message using the Simple Message Transfer Protocol (SMTP). Your new ALCS changes the entry and return conditions for this program.

High-level language program preparation

DXCBGSTR - Converting assembler DSECTs to C language structures

Your old ALCS includes the utility program DXCBGSTR. This program automates the conversion of assembler language DSECTs into corresponding C structures.

Your new ALCS does not include the DXCBGSTR program. Instead, the IBM z/OS XL C/C++ compiler provides a DSECT conversion utility which generates a structure to map an assembler DSECT. This utility is described in the *C/C++ User's Guide*. You might find this utility useful for creating structures for existing records in your installation.

ALCS generation

The following sections describe the new optional parameters that you might want to use.

For each generation macro described below, refer to Chapter 4, "Generating ALCS," on page 60 which gives full descriptions of the parameters for each macro.

SCTGEN generation macro

Your new ALCS adds or changes the following parameters of the SCTGEN macro:

DB2RECONNECT
Specify whether or not to automatically connect when DB2 starts. (This option was added to ALCS 2.3.1 by PTF.)

DYNTCB
Specify whether or not to enable the dynamic TCB facility. (This option was added to ALCS 2.3.1 by PTF.)

EMAILQCOUNT
Specify the threshold for the e-mail queue handler. (This option was added to ALCS 2.3.1 by PTF.)

EMAILQTIME
Specify the timeout for the e-mail queue handler. (This option was added to ALCS 2.3.1 by PTF.)

ENTREAD
Specify the entry read thresholds. (This option was added to ALCS 2.3.1 by PTF.)

ENTSTOR1
Specify the maximum entry storage limits (not including stack storage). (This option was added to ALCS 2.3.1 by PTF.)

ENTSTOR2
Specify the maximum entry storage limits (high-level language stack storage). (This option was added to ALCS 2.3.1 by PTF.)

ENTSTOR3
Specify the maximum entry storage limits (heap storage that does not fit in an entry storage block). (This option was added to ALCS 2.3.1 by PTF.)

NBRSU
Specify the number of type 3 storage units. (This option was changed in ALCS 2.3.1 by PTF.)

NOWAIT
Specify the maximum number of implied wait operations that an entry can request before it must delay. (This option was added to ALCS 2.3.1 by PTF.)

PERFMON
Specify whether or not to enable the performance monitor facility. (This option was added to ALCS 2.3.1 by PTF.)

RACF
Specify whether or not the external security manager for this ALCS is RACF. (This option was added to ALCS 2.3.1 by PTF.)

SUSIZE
Specify the size of type 3 storage units. (This option was changed in ALCS 2.3.1 by PTF.)

TCPPORT
Specify up to 8 port numbers for the TCP/IP concurrent servers (Listeners).

TCPVIPA
Specify a virtual IP address.

SEQGEN generation macro

Your new ALCS adds or changes the following parameters of the SEQGEN macro:

DEST
Specify the node and userid for a SYSOUT sequential file. (This option was added to ALCS 2.3.1 by PTF.)

Running the communication generation

The additional parameters which are provided by your new ALCS for the communication generation do not require you to change the sequence of the communication resources in your stage 1 input.

Caution:

It is important that you preserve the same communication resource ID (CRI) and communication ordinal for your existing communication resources. IBM recommends that you use the ALCS communication report file generator (see *ALCS Operation and Maintenance*) to check that you do not accidentally change these.

COMGEN generation macro

Your new ALCS adds the following parameters to the COMGEN macro:

OCTM, CRIRANGE, ORDRANGE
Specify details for online communication table maintenance (OCTM).

COMDFLT and COMDEF generation macros

Your new ALCS adds the following parameters to the COMDFLT and COMDEF macros:

FLOWID
Sepcify the MATIP flow ID. (This option was added to ALCS 2.3.1 by PTF.)

HDR
Specify the MATIP header type. (This option was added to ALCS 2.3.1 by PTF.)

MPX
Specify the MATIP multiplex type. (This option was added to ALCS 2.3.1 by PTF.)

MQCONVERT
Seocify whether or not to use MQ options for converting application message data on the MQ Bridge. (This option was added to ALCS 2.3.1 by PTF.)

NOP3270
Specify whether or not to allow the ALCS no-operation character to be part of the displayable character set in messages to IBM 3270 display terminals. (This option was added to ALCS 2.3.1 by PTF.)

QWARNING
Specify the printer queue threshold and frequency. (This option was added to ALCS 2.3.1 by PTF.)

SCRCOL
Specify the number of columns on a display terminal. (This option was added to ALCS 2.3.1 by PTF.)

RHOST
Specify up to four remote TCP/IP host addresses. (This option was changed in ALCS 2.3.1 by PTF.)

SPORT
Specify the TCP/IP local port number where your application provides services. (This option was added to ALCS 2.3.1 by PTF.)

SPORTMATCH
Specify whether or not to use the local port number to distinguish between services on the remote host which use the same protocol. (This option was added to ALCS 2.3.1 by PTF.)

VSAM data set access

Your new ALCS modifies the routines that ALCS uses to access VSAM data sets, in order to position ALCS for future changes in z/OS.

Before migration, you must do the following:

- Change the share options for your ALCS data sets using an IDCAMS job. See the NOTES file provided with PTF UK28709 for information about how to do this.
- If you use the DFSMSdss component of z/OS DFSMS to backup your ALCS data sets (including configuration data sets, real-time database data sets, and general file data sets) you must change the JCL for the backup jobs. See the NOTES file provided with PTF UK28709 for information about how to do this.
- If you have offline programs that process ALCS general files or general data sets, you may need to review them before migration. If they read from a general file or general data set which is online to ALCS, they must access the dataset as read only (DISP=SHR).

You can make these changes in advance of migration, since the routines that ALCS uses with your old ALCS to access VSAM data sets will work with either value of the VSAM share options.

Likewise, you do not need to undo these changes if you subsequently fall back.

(VSAM data set access was modified in ALCS 2.3.1 by PTF.)

ALCS long-term pool management

Your new ALCS implements an enhancement to long-term pool management.

Long-term pool activity monitor

This improves the way that ALCS assesses the dispense rate and time to depletion for long-term pool files. It applies to both Type 1 and Type 2 long-term pool.

Long-term pool monitoring in your old ALCS

For each long-term pool size, you can set a single dispense rate threshold value using the ZP00L command.

ALCS uses continuous exponential smoothing to calculate the average dispense rate for each pool size over the last 16 seconds.

If the dispense rate for any pool size exceeds the appropriate dispense rate threshold value, ALCS sends an **Attention** message to the RO CRAS:

DXC2758W PT-'*pool_type*' pool dispense rate is NR-'*number_of_records*' per second

ALCS also uses the dispense rate to predict when there will be no more records available in the pool. If this time is less than 12 hours, ALCS sends an Attention message to the RO CRAS stating the estimated time until pool depletion:

DXC2760W PT-'*pool_type*' pool will be depleted within HH-'*hh*' hours

While the dispense rate or the depletion time remain in this condition, ALCS continues to send Attention messages at 5 minute intervals.

Long-term pool monitoring in your new ALCS

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 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. It is therefore important that the pool activity monitor is sensitive to these fluctuations and does not issue unnecessary *high usage* warning messages (otherwise you are likely to just ignore the warnings and be unaware that there is a serious problem that requires your urgent attention).

The pool activity monitor therefore monitors pool dispense rates over short periods of time and longer periods of time. You tell it what the maximum dispense rate per second should be for periods of high system activity and when maintenance functions are run. You also tell it what the average dispense rate per second should be over a longer period of time (typically over one or more hours). To do this, you provide ALCS with short and long threshold values which are used by the long-term pool activity monitor for monitoring short and long periods of pool dispensing.

For each long-term pool size, ALCS accumulates the number of dispenses over 24 consecutive time intervals. At the end of each time interval, the long-term pool monitor examines the accumulated dispense counts and assesses them against several threshold values which you must set using the ZP00L command.

The ZP00L L*n*,DISPLAY command displays the threshold values, plus the counts of pool records dispensed during each of the last 24 pool intervals.

When you start ALCS 2.4.1 for the first time, the monitor time interval is initially 10 minutes and the threshold values are initially all zero. Warning messages are sent to RO CRAS stating that the threshold values are not set. ALCS does not start monitoring pool dispense rates until you have set the threshold values for each long-term pool.

You should immediately start monitoring the pool dispensed for each pool interval (via the ZP00L L*n*,DISPLAY command). Gather information on pool dispense rates during peak periods of system activity. As soon a possible, calculate and set the appropriate monitor time interval and threshold values for your installation using the ZP00L command. Any changes to the monitor time interval and threshold values are retained over an ALCS restart.

The following describes each threshold value that you must set:

- Minimum available records threshold

  For any pool size, if the number of available records goes below this threshold, ALCS sends an **Attention** message to the RO CRAS:

DXC2773W PT-'*pool_type*' Pool warning
Available records NR-'*number*' below minimum threshold

You can use the new ECB-controlled installation-wide exit program APM1 to suppress this message if required.

- Dispense rate thresholds

  Short rate threshold - Highest usage rate in records per second over a period of one monitor interval.
  Short time threshold - Time to exhaustion based on current short rate.
  Long rate threshold - Highest usage rate in records per second over a period of 24 monitor intervals.
  Long time threshold - Time to exhaustion based on current long rate.

For any pool size, if the dispense rate or time to exhaustion passes one of these thresholds, ALCS sends an **Attention** message to the RO CRAS:

DXC2774W PT-'*pool_type*' Pool warning
Available records NR-'*number*'
Dispense rate over last MM-'*mm*' minutes was NR-'*number_of_records*' per second
At this rate pool will be exhausted in HH-'*hh*' hours

You can use the new ECB-controlled installation-wide exit program APM2 to suppress this message if required.

Subsequently, you can use the ZP00L command to review the interval and threshold values and, if necessary, to alter them.

Please refer to the following topics for further information:

- ZP00L command in *ALCS Operation and Maintenance*.
- “Long-term pool activity monitor exit programs - APM1 and APM2” on page 382.

Initialize the long-term pool activity monitor thresholds

When you start an ALCS 2.4.1 system for the first time, you must initialize the monitor thresholds to values that are suitable for your installation, using the ZP00L command.

Performance monitor

Your new ALCS includes an online performance monitor tool that 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. In this way, the performance monitor overcomes some of the drawbacks of using data collection. It is described in section "Performance monitoring and control" in *ALCS Operation and Maintenance*.

The performance monitor is comprised of a number of separate elements, including

- an ALCS product-specific programming interface (APIDC macro),
- an ALCS operator interface (ZPERF command),
- a system generation option (SCTGEN PERFMON parameter),
- installation-wide ECB-controlled exits for customization (APF1 and APF2),
- a sample program CPM0 that converts history file records from an earlier non-IBM version of the performance monitor (also called the JPM) to the format required by the ALCS performance monitor.

(The performance monitor was added to ALCS 2.3.1 by PTF.)

Dynamic TCBs

Your new ALCS includes a dynamic TCB facility that allows you to increase or decrease the number of active ALCS CPU loops in real time.

The dynamic TCB facility is comprised of a number of separate elements, including

- an ALCS operator interface (ZCTCB command),
- a system generation option (SCTGEN DYNTCB parameter),
- an option on the startup parameters for the ALCS job or started task.

(The dynamic TCB facility was added to ALCS 2.3.1 by PTF.)

ALCS communications

Your new ALCS implements some enhancements and new facilities for communications.

Online communication table maintenance (OCTM)

Your new ALCS includes an online communication table maintenance facility which can help to manage changes in your communication network configuration.

OCTM is comprised of a number of separate elements, including an ALCS product-specific programming interface (COMTC macro), an ALCS operator interface (ZOCTM command), an OCTM database that contains communication resource definitions, and a communication end user system (CEUS). The CEUS is not part of the ALCS product, although an example CEUS front-end utility, based on 3270 screen maps, is available from the ALCS Web Site. You can use this, or you can develop your own CEUS utility.

The purpose of the CEUS is to provide a user-friendly way of specifying changes to the ALCS communication network and applying those changes directly in the ALCS communication table. The end-user submits communication change requests to your CEUS which checks if the change requests are valid and authorized. If they are, the CEUS presents them to ALCS using the COMTC programming interface. The end-user then requests your CEUS to load those change requests into the ALCS communication table, confirm and then commit them. The CEUS can also display changes to the ALCS online communication table and can manage change requests either singly or in batches. The COMTC programming interface enables the CEUS to perform all these functions.

OCTM also includes an offline support program called DXCCTMOL. You can use this program in conjunction with the offline installation-wide exit program DXCUTMOL to apply changes to the communication resources that are managed by OCTM. (DXCCTMOL and DXCUTMOL were added to ALCS 2.3.1 by PTF).

An OCTM User Guide which describes the COMTC programming interface and provides guidance on the design of the CEUS front-end utility is available from the ALCS Web Site.

We advise you to review the following information about the OCTM facility in case you want to exploit this new feature:

- ZOCTM command in *ALCS Operation and Maintenance*.
- “OCTM policing exit program - AOCM” on page 376.
- COMGEN generation macro in “COMGEN macro” on page 98.
- OCTM User Guide available on the ALCS Web Site.

Enhanced communication report file generator (DXCCOMOL)

The offline communication report file generator program (DXCCOMOL) provides a report of the communication resources defined in the offline communications generation. Many ALCS customers use DXCCOMOL to obtain this report as it provides a list of all the communication resources and includes information that is not available in the offline generation (for example, the CRI addresses that have been allocated by ALCS).

In your new ALCS, the DXCCOMOL communications report program has been enhanced so that it includes all the communication resources on the OCTM database as well as those defined in the offline communications generation. (DXCCOMOL only includes communication resources that are confirmed or committed; it does not include any unconfirmed changes.) For those ALCS customers who are not using the OCTM facility, DXCCOMOL can still be used for creating a communications report. 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.

DXCCOMOL produces a similar output file to that currently produced. It 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).

PARM parameter added to DXCCOMOL JCL

An additional parameter is now provided on the EXEC statement for the DXCCOMOL job (using the PARM parameter). This is an optional parameter, and is used to identify the OCTM sequential file data set name when OCTM is being used. The data set name can be any length (and it can therefore overflow onto a second line), for example:

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

This OCTM sequential file is created by the OCTM Database Backup function (ZOCTM BACKUP command). 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.

Enhancements for TCP/IP communications

Multiple concurrent servers (Listeners)

Your old ALCS supports a single TCP/IP concurrent server.

Your new ALCS allows up to eight concurrent servers to be started at the same time using different port numbers. If you have coded the TCP/IP concurrent server exit program ATCP for processing inbound connection requests, it will continue to work in the same way with a single concurrent server. If you plan to use the multiple concurrent server facility in your new ALCS, then you may need to re-work ATCP to distinguish between connections on the different concurrent servers.

Please refer to the following topics for further information:

- ZCTCP command in *ALCS Operation and Maintenance*.
- ECB-controlled installation-wide exit program ATCP in “TCP/IP concurrent server exit program - ATCP” on page 402.
- The TCPPORT parameter on the SCTGEN generation macro in “SCTGEN macro” on page 72.

Virtual IP addressing (VIPA)

Your new ALCS supports virtual IP addressing for TCP/IP connections.

Please refer to the following topics for further information:

- ZCTCP command in *ALCS Operation and Maintenance*.
- The TCPVIPA parameter on the SCTGEN generation macro in “SCTGEN macro” on page 72.

E-mail output queue

In your old ALCS, outbound e-mail messages were discarded if the remote mail transfer agent (MTA) was unavailable at the time messages were presented to ALCS for sending.

In your new ALCS, the e-mail messages are added to a queue before transmission. If the e-mail queue handler can not send the messages because the MTA is not available, they remain on the queue until the queue handler retries the transmission later.

If you are using the ECB-controlled callable service program CSMS to send messages from an application using SMTP, then you may need to re-work your application for the modified return conditions from CSMS.

Note: The first time your new ALCS sends an e-mail message, it initializes a keypoint record on the ALCS database and issues system error 000329. This is normal.

Please refer to the following topics for further information:

- ZMAIL command in *ALCS Operation and Maintenance*.
- ECB-controlled callable service program CSMS in "Send a message using SMTP - CSMS" on page 430.

E-mail HTML messages

If you are using the ECB-controlled callable service program CSMS to send messages from an application using SMTP, your new ALCS allows your program to prepare the message body using plain text or HTML (the default is plain text). (This feature was added to ALCS 2.3.1 by PTF.)

ALCS Web Server

Your new ALCS changes the behaviour of the ALCS Web Server by allowing HTTP 1.0 persistent connections.

You should consider implementing the ALCS idle connection timeout for the ALCS Web server if you have not already done so. You may also need to defined additional connections to accommodate persistent connections. See the NOTES file provided with PTF UK28654 for information about how to do this. (This feature was added to ALCS 2.3.1 by PTF.)

MATIP terminals

Your old ALCS allows you to define MATIP terminals that are connected through TCP/IP server resources defined with COMDEF LDTYPE=TCPIP,TERM=(SERVER,MATIPA).

Your new ALCS allows you to define MATIP terminals that are connected through TCP/IP client resources defined with COMDEF LDTYPE=TCPIP,TERM=(CLIENT,MATIPA) as well. (This feature was added to ALCS 2.3.1 by PTF.)

End users, operations, and programmed operators

New and enhanced ALCS commands

ZACOM
Enhanced for additional function.

ZAFIL
Enhanced for 8-byte file addresses.

ZASER
Enhanced for additional function.

ZASYS
Enhanced for additional function.

ZCTCB
Implements the dynamic TCB facility.

ZCTCP
Enhanced for additional function and to display additional information.

ZDASD
Enhanced to display additional information.

ZDCLR
Enhanced to allow the CPU collector to run on its own.

ZDCOM
Enhanced to display additional information.

ZDECB
Enhanced to display additional information.

ZDFIL
Enhanced for 8-byte file addresses.

ZDSEQ
Enhanced to display additional information.

ZMAIL
Enhanced to support the e-mail queue handler function.

ZOCTM
Implements the online communications table maintenance (OCTM) function.

ZPCTL
Enhanced to display additional information.

ZPERF
Implements the peformance monitor facility.

ZPOOL
Enhanced to support the long-term pool monitor function.

ZPURG
Enhanced to purge an entry immediately, regardless of its current activity and mark the storage unit as "quarantined".

ALCS commands, including new and enhanced commands, are described in *ALCS Operation and Maintenance*.

System takeover

Your new ALCS includes a system takeover facility which can help to minimize the delay when an alternate ALCS takes over from a prime ALCS.

You can enable ALCS for automatic system takeover by specifying the takeover option on the parameters for the ALCS job or started task.

When an alternate ALCS is enabled for automatic system takeover, it 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.

We advise you to review the information about the system takeover facility in *ALCS Operation and Maintenance* in case you want to exploit this new feature.

Migration from ALCS 2.1.3

Sequential files

Data collection output

The data collection output from your new ALCS is not the same as your old ALCS. We recommend that you use the statistical reports generator (DXCSRG) that comes with the ALCS which produced the data collection output. If you have offline programs that process ALCS data collection output, you may need to review them. They will probably still work with your new ALCS, provided they are not sensitive to the record length.

User sequential file

Your new ALCS includes a new type of system sequential file called the ALCS user file. ALCS itself does not write to this system sequential file. It is provided for you to write data from an installation-wide monitor exit, using the callable service UWSEQ. "Write to a system sequential file -- UWSEQ" on page 358 explains how to do this.

(This sequential file type was added to ALCS 2.2.1 by PTF.)

Installation-wide exits

Installation-wide monitor exits - new and changed exits

New exits

Your new ALCS implements the following new installation-wide monitor exits:

USRCDSA
Allows you to modify the processing of `ZPCTL LOAD,LIST` and `ZACOM LOAD,LIST` commands. (This exit was added to ALCS 2.2.1 by PTF.)

USRCDSB
Allows you to modify the processing of `ZPCTL BACKOUT,LIST` and `ZACOM BACKOUT,LIST` commands. (This exit was added to ALCS 2.2.1 by PTF.)

USRCOM0
Allows you to implement support for incoming LUSTAT, RTR, and SIGNAL requests from VTAM. (This exit was added to ALCS 2.1.3 by PTF.)

USRCOM6
Allows you to accept or reject a VTAM logon request. (This exit was added to ALCS 2.2.1 by PTF.)

USRDAS1
Allows you to authorize ALCS to accept a new DASD generation table when it does not match the currently loaded one.

USRDCR1
Allows you to modify the data collected by ALCS data collection. (This exit was added to ALCS 2.2.1 by PTF.)

USRDFMT
Allows you to implement custom date and time formats.

USREID
Allows you to allow or deny changes to the originator information for an entry using `GTFCC CONV,MODIFY`. (This exit was added to ALCS 2.2.1 by PTF.)

USRMQI2
Allows you to record information about an MQI call after the call has completed. (This exit was added to ALCS 2.2.1 by PTF.)

USRSEQ1
Allows you to request ALCS to wait for the recall by DFHSM of specific general sequential files (if they have been migrated). (This exit was added to ALCS 2.2.1 by PTF.)

USRSQL2
Allows you to implement custom authorization checking for SQL calls from applications. (This exit was added to ALCS 2.2.1 by PTF.)

USRTAB7-10
Allows you to implement custom translate tables for the ASCIC monitor-request macro. (These exits were added to ALCS 2.2.1 by PTF.)

USRTCP1
Allows you to implement custom authorization checking for TCP/IP calls from applications. (This exit was added to ALCS 2.1.3 by PTF.)

USRTCP2
Allows you to inform ALCS when a complete TCP/IP input message has been received. (This exit was added to ALCS 2.1.3 by PTF.)

USRTCP3
Allows you to validate TCP/IP output messages. (This exit was added to ALCS 2.2.1 by PTF.)

USRTCP4
Allows you to validate TCP/IP input messages. (This exit was added to ALCS 2.2.1 by PTF.)

USRTCP5
Allows you to implement custom processing when TCP/IP connections start. (This exit was added to ALCS 2.2.1 by PTF.)

USRTCP6
Allows you to implement custom processing when TCP/IP connections stop. (This exit was added to ALCS 2.2.1 by PTF.)

USRTCP7
Allows you to implement custom processing for input from an ACSA terminal emulator. (This exit was added to ALCS 2.2.1 by PTF.)

USRTCPA
Allows you to close a TCP/IP connection after the blocked send timeout expires. (This exit was added to ALCS 2.2.1 by PTF.)

Full details of the new exits are in "Implementing installation-wide monitor exits" on page 237.

Changed exits

Your new ALCS changes the following installation-wide monitor exits:

USRAPPC
New return conditions added to identify an input or output message. This causes the count of input or output APPC messages to be incremented. ZSTAT MSG or ZSTAT ALL displays these counts. When data collection is collecting statistics about input and output messages, this also causes the statistics for input or output APPC messages to be incremented. (This exit was changed in ALCS 2.2.1 by PTF.)

USRCOM5
Changed to distinguish between fields in the system and user area of the communication table entry. (This exit was changed in ALCS 2.2.1 by PTF.)

USRCOM8
Allows you to ignore input from an ALC printer. (This exit was changed in ALCS 2.2.1 by PTF.)

USRGFS
Changed to account for emergency pool recovery (PDU).

USRMQI1
New return conditions added to identify an input or output message. This causes the count of input or output MQ messages to be incremented. ZSTAT MSG or ZSTAT ALL displays these counts. When

data collection is collecting statistics about input and output messages, this also causes the statistics for input or output MQ messages to be incremented. (This exit was changed in ALCS 2.2.1 by PTF.)

USRSQL1
Changed to provide the DB2 subsystem name as a parameter. (This exit was changed in ALCS 2.2.1 by PTF.)

USRTCP1
New return conditions added to identify an input or output message. This causes the count of input or output TCP/IP messages to be incremented. ZSTAT MSG or ZSTAT ALL displays these counts. When data collection is collecting statistics about input and output messages, this also causes the statistics for input or output TCP/IP messages to be incremented. (This exit was changed in ALCS 2.2.1 by PTF.)

ALCS services for installation-wide monitor exits - new and changed services

Your new ALCS implements a number of new services. Review each of these new services to verify the functions they provide.

New services

Your new ALCS implements the following new ALCS services for installation-wide monitor exits:

UDLEVGET
Obtain a storage block and attach it to a DECB. (This service was added to ALCS 2.2.1 by PTF.)

UDLEVREL
Release a storage block attached to a DECB. (This service was added to ALCS 2.2.1 by PTF.)

UDLEVVAL
Validate a DECB storage level. (This service was added to ALCS 2.2.1 by PTF.)

UTAB7, UTAB8, UTAB9, UTAB10
Find a translate table for ASCIC. (These services were added to ALCS 2.2.1 by PTF.)

Changed services

Your new ALCS changes the following ALCS services for installation-wide monitor exits:

UCOMGET
Changed to return the address of the system or user part of the communication table entry. (This service was changed in ALCS 2.2.1 by PTF.)

USTRVAL
Changed to account for protected global areas. (This service was changed in ALCS 2.2.1 by PTF.)

UWSEQ
Changed to account for ALCS user sequential file. (This service was changed in ALCS 2.2.1 by PTF.)

Installation-wide ECB-controlled exits - new and changed exits

Your new ALCS implements a number of new exits. Review each of these new exits to verify the functions they provide.

New exits

Your new ALCS implements the following new installation-wide ECB-controlled exits:

APA1
Allows you to modify the minimum count of long-term pool addresses that must remain after a database restripe (ZDASD STRIPE) has been performed.

AMQ0, AMQ1, AMQ2, AMQ3
Allow you to customize the behaviour of the ALCS MQ Bridge. (These exits were added to ALCS 2.2.1 by PTF; in ALCS 2.4.1 they are replaced by installation-wide monitor exits USRMQB0, USRMQB1, USRMQB2, USRMQB3.)

APL1, APL2
Allows you to implement custom processing for applications during system state change. (These exits were added to ALCS 2.2.1 by PTF.)

APRA
Allows you to selectively purge messages on printer queues. (This exit was added to ALCS 2.2.1 by PTF.)

ARE1
Allows you to modify the output which is sent to the terminal by the ALCS retrieve function.

ASM0, ASM1, ASM2, ASM3, ASM5, ASM6
Allow you to customize the ALCS e-mail facility. (These exits were added to ALCS 2.2.1 by PTF.)

ATCP
This exit processes TCP/IP connections that have been received by the ALCS TCP/IP concurrent server (listener). (This exit was added to ALCS 2.1.3 by PTF.)

ATR9
You can implement program ATR9 to build and transmit (for example, across a communication link) a response message during remote terminal trace (as an alternative to message DXC8464I).

AWM1, AWM2
Allow you to customize the ALCS Web Server facility. (These exits were added to ALCS 2.2.1 by PTF.)

Changed exits

Your new ALCS changes the following installation-wide ECB-controlled exits:

AUS1
Allow space suppression in ALCS commands. (This exit was changed in ALCS 2.2.1 by PTF.)

AGT*n*
Changes to allow you to define global "tags" of up to 8 characters.

AUS1
Changes to allow multiple users who share a terminal to have their own input message stack for the ALCS retrieve function (in addition to the current support for scrolling).

AUS2
Changes to allow you to control use of the ALCS retrieve function (in addition to controlling use of scroll logging).

Installation-wide exits - new and changed examples

We have modified some of the sample exits provided with IPARS - ALCS V2, and added some new sample exits. These provide improved examples of how you can use these exits. If your application is based on, or similar to, IPARS - ALCS V2, then these examples may help you identify changes or additions that you need to make, or that you may want to make.

Application global area

Storage protection for the global area

You may request ALCS to provide storage protection for global areas 1 and 3 (refer to the GLBLPROT parameter on the ALCS generation SCTGEN macro). If your installation is using global area protection then application programs that modify data in global areas 1 and 3 must change their PSW protect key before modifying the data, and when they have completed their data modifications, restore their PSW protect key. Applications use the GLMOD, KEYCC, KEYRC and FILKW macros for this (refer to *ALCS Application Programming Reference - Assembler*).

(This facility was added to ALCS 2.2.1 by PTF.)

Trace

Workstation trace

Your new ALCS can use the ALCS trace facility to initiate source level debugging of C/C++ programs on a remote workstation.

(This facility was added to ALCS 2.2.1 by PTF.)

Diagnostic trace - RS and RSA options

Your new ALCS includes new diagnostic trace display options for showing 64 or 4096 bytes of storage addressed by the general registers and the PSW.

(This facility was added to ALCS 2.2.1 by PTF.)

Conversational trace - asynchronous trace

If you are running conversational trace, your new ALCS allows you to start or stop asynchronous trace. While 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.

(This facility was added to ALCS 2.2.1 by PTF.)

ALCS application program interfaces

Programs that you must reassemble

You must reassemble and relink-edit the global load control programs GOA0 through GOAE.

Extended date and time formats

Your new ALCS supports a much wider variety of formats for dates and times. Your application programs can request these new formats using extensions to the `TIMEC` assembler macro and the corresponding `timec` C language function.

See *ALCS Application Programming Reference - Assembler* for a full description of the `TIMEC` macro. See *ALCS Application Programming Reference - C Language* for a full description of the `timec` C function.

SAF authority checking

Your new ALCS allows you to control access to resources such as application functions, data, and so on using an MVS external security manager (for example, IBM's RACF). You decide on names for the resources you want to control and define them (and their security characteristics) to your external security manager.

Your application programs can check whether or not the originator of the message they are processing has authority to access the resource.

To do this, your applications use the new AUTHC macro. When an application program issues AUTHC, ALCS uses the MVS system authorization facility (SAF) to interface with your external security manager.

You can use SAF authority checking either as an addition to, or as a replacement for, existing controls such as:

- Restricting functions to designated CRASs

- Restricting functions based on the AAA duty code (this method is common in airline passenger-services applications).

For example, an airline might decide to restrict the schedule-change commands so that only explicitly authorized end users can change flight schedules. To do this, the airline defines a name for the schedule-change function. The schedule-change application programs use AUTHC to check that the originator is authorized to request the function.

See *ALCS Application Programming Reference - Assembler* for a full description of the AUTHC macro.

Local program work area and local program save stack

Your new ALCS implements a **local program work area** in the ECB. The local program work area is called CE1WKC. The format of CE1WKC is similar to the existing CE1WKA and CE1WKB work areas; CE1WKC contains fields with names EBL*nnn*.

ALCS automatically saves and restores the contents of CE1WKC across ENTER/BACK linkages. This means that you can use fields in CE1WKC without worrying that programs which you call might corrupt them, or that programs which call you might depend on the fields remaining unchanged. Of course, this also means that you cannot use CE1WKC to pass information between programs.

To avoid unnecessary saving and restoring of CE1WKC, programs that use it must say so by using a BEGIN macro parameter, LPW=YES. If and only if the BEGIN macro specifies LPW=YES, the EB0EB macro generates labels for CE1WKC itself, and the fields that it contains.

See *ALCS Application Programming Reference - Assembler* for a full description of the BEGIN and EB0EB macros.

Your new ALCS also implements a **local program save stack**. Programs that specify LPW=YES on the BEGIN macro can use the new SAVEC macro to save (push) information into the local save stack and later can use SAVEC to restore (pop) that information. SAVEC can save (push) and restore (pop) one or more of:

ECB work area 1 (CE1WKA)
ECB work area 2 (CE1WKB)
The local program work area (CE1WKC)
The general-purpose registers
The floating-point registers

SAVEC PUSH calls can be nested. If so, each SAVEC POP restores the information saved by the most recent SAVEC PUSH. Also, the entire local save stack is saved and restored across ENTER/BACK linkages. The following code fragment illustrates this:

```
SAVEC PUSH=WKA            Save work area 1 (CE1WKA)
...   (store information in CE1WKA for program XYZ1)

SAVEC PUSH=(GPRS,FPRS)    Save general and FP registers
ENTRC XYZ1                Call program XYZ1

SAVEC POP                 Restore general and FP registers
...   (examine information returned in CE1WKA by XYZ1)

SAVEC POP                 Restore work area 1 (CE1WKA)
```

See *ALCS Application Programming Reference - Assembler* for a full description of the SAVEC macro.

Context-sensitive help

Your new ALCS implements an extension to the existing help facility of your old ALCS. It is called **context-sensitive** help. It works like this.

When each message arrives at ALCS, ALCS clears the **help context** for that originator. The help context comprises a help topic and subtopic. While processing the message, an application program can (optionally) set a help context. If the end user then requests help (ZHELP command with no parameters), the ALCS help facility displays the help for the help context topic and subtopic.

Suppose an application provides a message that displays the symptoms of a specified disease. The application might provide help for that message (describing the message format, and so on) using the help topics 'DISEASE SYMPTOM'.

The application program that identifies an input message as a request to display disease symptoms can set the help context to 'DISEASE SYMPTOM'.

When an end user enters a message to display disease symptoms, they may get an unexpected response. If so, they probably assume that they do not understand exactly how to use the message. They enter ZHELP and ALCS displays the appropriate help information.

If no application sets the help context during the processing of a particular message then there is no help context. In this case, ALCS responds to ZHELP with an index of help topics.

See *ALCS Application Programming Reference - Assembler* for a full description of the HELPC macro. See *ALCS Application Programming Reference - C Language* for a full description of the helpc C function.

COMIC macro - applications verifying CRAS status

In addition to allowing you to define specific CRASs (Prime CRAS, alternate CRAS 1, and so on), your new ALCS allows you to assign CRAS authority to other terminals. For example, in addition to Prime CRAS itself, there can be several other terminals that have Prime CRAS authority. Your new C0OIC macro includes extra indicators (in the extended system data) that allow you to distinguish between an actual CRAS and another terminal that has the same authority.

You may have application programs that issue the COMIC macro to verify the CRAS status of a terminal. For example, a program might include the sequence:

```
         TM    ICESPRC,L'ICESPRC       Is this Prime CRAS?
         BO    IS_PRIME_CRAS           Yes, branch

         TM    ICESATN,L'ICESATN       Is this alternate CRAS?
         BO    IS_ALT_CRAS             Yes, branch
```

If the program is testing the originator's *authority* to perform some function then the above sequence is OK. But if the program is testing if the originator *is* Prime or alternate CRAS then you must change the sequence to something like:

```
         TM    ICESPRC,L'ICESPRC       Does this have PRC authority?
         BNO   NOT_PRC                 No, branch, not Prime CRAS
         TM    ICESFPR,L'ICESFPR       Is this really Prime CRAS?
         BNO   IS_PRIME_CRAS           Yes, branch

NOT_PRC  DC    0H'0'
         TM    ICESATN,L'ICESATN       Does this have ALT CRAS authority?
         BNO   NOT_ALT                 No, branch, not alternate CRAS
         TM    ICESFAL,L'ICESFAL       Is this really alternate CRAS?
         BNO   IS_ALT_CRAS             Yes, branch

NOT_ALT  DC    0H'0'
```

See *ALCS Application Programming Reference - Assembler* for a full description of the newly documented fields of the C0OIC macro.

GTFCC - Special interface to ALCS trace

Some ALCS applications handle input messages and responses for terminals that connect to ALCS through a communication link, for example an LU6.1 link. That is, the *application* implements a function similar to the ALCS message router facility.

In your old ALCS system, there is no way to "tell" ALCS that an entry from one of these terminals originated from the terminal rather than from the link (the ECB entry origin information contains the link CRI, not the terminal CRI).

Your new ALCS implements a special trace interface, using the GTFCC macro, which allows an application program to change the entry origin information from a communication link to an OSYS terminal owned by the system that the communication link connects. Changing the entry origin has several effects, including:

- Diagnostic or remote terminal trace commands can specify the OSYS terminal CRI or CRN. ALCS trace can recognize that the entry originated from the specified terminal.
- ZPCTL LOAD can specify the OSYS terminal CRI or CRN. ALCS program management can recognize that the entry originated from the specified terminal.
- ALCS SAF authority checking uses the default user ID of the terminal (instead of the default user ID of the communication link) to check the entry's access authority.

If you have an application that handles OSYS terminals in this way, you may want to use this new interface. See the *ALCS Application Programming Reference - Assembler* for details of the new GTFCC parameters.

You may also want to use the new trace exit program ATR9. "Trace exit programs - ATR1, ATR2, and ATR9" on page 404 describes how to do this.

Data event control blocks (DECBs)

A data event control block (DECB) contains a storage level and data level. An application program can use a DECB as an alternative to using a storage level or data level in the ECB. Although a DECB does not physically reside in an ECB, the DECB fields specify the same information as those in the ECB.

An application program can dynamically acquire a DECB by using the DECBC FUNC=CREATE monitor-request macro (`tpf_decb_create` C function).

For Assembler programs, the IDECB macro generates the IDECB DSECT that defines the labels for the fields in a DECB. Similarly for C-language programs, the c$decb header file generates a C data structure defined as type TPF_DECB that defines the labels.

Note: The $ character is the national currency symbol (X'B5').

Several ALCS general use programming macros and C functions include a new, optional parameter DECB. In addition, your new ALCS implements new ALCS services for installation-wide monitor exits for managing DECBs (UDLEVGET, UDLEVREL, and UDLEVVAL).

See *ALCS Application Programming Guide* for more information on the use of DECBs.

(This facility was added to ALCS 2.2.1 by PTF.)

cinfc, IPRSE_parse, tpf_STCK, wtopc C functions and <c$stdhd.h> C header

Your new ALCS supports the `cinfc`, `IPRSE_parse`, `tpf_STCK`, and `wtopc` C functions and the <c$stdhd.h> C header. These are provided for compatibility with TPF.

(Added to ALCS 2.2.1 by PTF.)

ISTD8 macro and <c$std8.h> C header

Your new ALCS supports the ISTD8 macro and <c$std8.h> C header. These are provided for compatibility with TPF.

(Added to ALCS 2.2.1 by PTF.)

LODIC macro

Your new ALCS supports the CONDITIONALDEFER parameter on the LODIC macro. LODIC CONDITIONALDEFER forces the entry to lose control if and only if the entry needs to lose control to avoid locking out other entries.

(Added to ALCS 2.2.1 by PTF.)

SWISC macro

Your new ALCS supports the SWISC macro. This macro is provided for compatibility with TPF.

(SWISC was added to ALCS 2.2.1 by PTF.)

WILDC macro

Your new ALCS supports the WILDC macro for pattern matching.

(WILDC was added to ALCS 2.2.1 by PTF.)

ECB-controlled callable services

New callable services

Your new ALCS implements the following new ECB-controlled callable-service programs:

CSMS
An application program can enter CSMS to send a message using the Simple Message Transfer Protocol (SMTP). (This program was added to ALCS 2.2.1 by PTF.)

CWW5, CWW6
These callable services are part of the ALCS Web Server support. (These programs were added to ALCS 2.2.1 by PTF.)

Changed callable services

Your new ALCS changes the following ECB-controlled callable-service program:

CAP1
Provides an extended interface for specifying records to be scanned. (This program was changed in ALCS 2.2.1 by PTF.)

ALCS generation

There are a few new mandatory parameters for your new ALCS generation.

The new mandatory parameters *add* information to your configuration. They do not change existing configuration information. This means you can still fall back to your old ALCS system.

Some generation parameters supported by your old ALCS are not used by your new ALCS. If you use generation stage 1 input that specifies these parameters, you may get attention (severity 4) MNOTEs.

The following sections describe the new mandatory parameters and the new optional parameters that you might want to use.

For each generation macro described below, refer to Chapter 4, “Generating ALCS,” on page 60 which gives full descriptions of the parameters for each macro.

ALCS generation macro

Your new ALCS adds or changes the following parameters of the ALCS macro:

PROC
Your new ALCS requires the high-level assembler and will not work with older assemblers. Your new ALCS defaults to use the high-level assembler. If you use non-default values for this parameter then you may need to change them.

ALCSGEN generation macro

Your new ALCS adds or changes the following parameters of the ALCSGEN macro:

STAGE2=NOJCL
Punches the STAGE 2 generation deck without punching any job control (JCL) statements.

SCTGEN generation macro

Your new ALCS adds or changes the following parameters of the SCTGEN macro:

DATEFORM, TIMEFORM, TIMEDATEFORM
Specify default formats for date and time information.

DCLOPTS
Specify the status of data collection when ALCS starts.

DIADUMP, SVCDUMP
Specify whether an ALCS diagnostic dump and/or SVC dump will be taken during termination. (This option was added to ALCS 2.2.1 by PTF.)

EMAILDOMAIN, EMAILMTA, EMAILPORT, EMAILTIMEOUTR, EMAILTIMEOUTS, EMAILPOSTMASTER
Specify details of your e-mail implementation. (E-mail support was added to ALCS 2.2.1 by PTF.)

ENTST, ENTLT
Specify individual short-term and long-term pool dispense limits. (This option was added to ALCS 2.2.1 by PTF.)

GLBLPROT
Specify whether or not the global area is protected. (Global area protection was added to ALCS 2.2.1 by PTF.)

MIGSEQ
Specify whether or not migrated sequential files are supported. (This option was added to ALCS 2.2.1 by PTF.)

MONTHNAME, MONTHABBR
Specify the month names and abbreviated month names that you use.

PDULOGSTREAM
Specifies the name of the MVS logger log stream for your emergency pool recovery (PDU) information, and activate the ALCS PDU function.

TCPIP, TCPLIST, TCPNAME, TCPPORT
Specify details of your TCP/IP implementation. (TCP/IP support was added to ALCS 2.1.3 by PTF.)

USRDTA generation macro

Your new ALCS adds or changes the following parameters of the USRDTA macro:

ACTION=BUILD
Your new ALCS supports this new option which you use in conjunction with the new restripe facility.

ID, IDSYM
Your new ALCS contains new functions that use new pool file record identifiers. You must add these new IDs to your database generation. To do this, you use the ID parameter of the USRDTA macroinstruction or macroinstructions for the corresponding pool type. The new record IDs are:

X'AC08'
Retrieve record.

We recommend that you define this record ID for the L3LT pool. If you use a larger (or smaller) record size then the retrieve function saves a correspondingly larger (or smaller) number of input messages.

DBHIST generation macro

Your new ALCS adds or changes the following parameters of the DBHIST macro:

BUILD_DIRECTORIES
Your new ALCS supports this new parameter which you use in conjunction with the new restripe facility.

COMGEN generation macro

Your new ALCS adds or changes the following parameters of the COMGEN macro:

ENTRIES
Your new ALCS adds a subparameter for TCP/IP resources. (TCP/IP support was added to ALCS 2.1.3 by PTF.)

LOGON
Specifies logon parameters for the ALCS secure access control function.

RCVSZE
Specifies size of buffer used for receiving input messages from 3270 displays (this support was added to ALCS 2.1.3 by PTF).

RCVSZEIP
Specifies size of buffer used for receiving input messages from TCP/IP commections (this support was added to ALCS 2.2.1 by PTF).

SAFOPTION
Specifies options for the ALCS secure access control function.

TIMEOUTV
Specifies timeout for VTAM SEND macro (this support was added to ALCS 2.2.1 by PTF).

COMDFLT and COMDEF generation macros

Your new ALCS adds or changes the following parameters of the COMDFLT and COMDEF macros:

CONV
Specify details of APPC resources. (This support was added to ALCS 2.2.1 by PTF.)

ISTATUS
Your old ALCS ignores the initial status for:

Prime CRAS
RO CRAS
Alternate CRASs AT1-AT16
Alternate CRAS printers AP1-AP16

For your new ALCS, you must specify ISTATUS=ACTIVE if you want ALCS to start the sessions automatically.

INQNAME, OUTQNAME, SQNAME
Specify details of MQ queues. (This support was added to ALCS 2.2.1 by PTF.)

LDTYPE
Your new ALCS supports the following additional LDTYPE values:

APPC
An ALCS-owned APPC conversation or pair of conversations. (This support was added to ALCS 2.1.3 by PTF.)

MQ
An MQ queue. (This support was added to ALCS 2.2.1 by PTF.)

MQTERM
An ALCS terminal accessed through an MQ queue. (This support was added to ALCS 2.2.1 by PTF.)

TCPIP
An ALCS-owned TCP/IP connection or set of connections. (TCP/IP support was added to ALCS 2.1.3 by PTF.)

TCPIPALC
An ALCS terminal (or compatible device) that connects to ALCS through TCP/IP. (This support was added to ALCS 2.2.1 by PTF.)

LLUNAME, MODE, PLUNAME, SYMDEST, TPNAME
Specify details of an ALCS-owned APPC conversation or pair of conversations. (This support was added to ALCS 2.1.3 by PTF.)

LOGON
Specifies logon parameters for the ALCS secure access control function.

LPORT, RHOST, RPORT, RPORTMATCH, TRANSLATE
Specify details of an ALCS-owned TCP/IP connection or set of connections. (TCP/IP support was added to ALCS 2.1.3 by PTF.)

RETRV
Specify whether the retrieve facility is activated automatically for the resource.

SFORM
Specifies mixed-case and FIS support for applications (this support was added to ALCS 2.2.1 by PTF).

TERM
Your new ALCS supports the following additional TERM values:

CLIENT
An ALCS-owned TCP/IP connection where ALCS acts as the client.

SERVER
An ALCS-owned TCP/IP connection or set of connections where ALCS acts as the server.

(TCP/IP support was added to ALCS 2.1.3 by PTF.)

SEQGEN generation macro

Your new ALCS adds or changes the following parameters of the SEQGEN macro:

BLKSIZE
This parameter is optional.

TYPE
A timestamp options is added on this parameter for RT sequential files.

ALCS communications

E-mail

Your new ALCS implements e-mail for end users and application programs. This allows you to send and receive messages over a TCP/IP network using the TCP/IP SMTP protocol.

We advise you to review the following information about the e-mail function in case you want to exploit this new feature:

- ZCTCP command in *ALCS Operation and Maintenance*.
- ZMAIL command in *ALCS Operation and Maintenance*.
- ECB-controlled callable service program CSMS in “Send a message using SMTP - CSMS” on page 430.
- ECB-controlled exit programs ASM0, ASM1, ASM2, ASM3, and ASM6 in “ALCS e-mail exit programs - ASM0 through ASM3, ASM5, ASM6, and ASMA” on page 393.
- SCTGEN generation macro in “SCTGEN macro” on page 72.

(E-mail support was added to ALCS 2.2.1 by PTF.)

MQ Bridge

Your new ALCS implements an MQ Bridge function.

The ALCS MQ Bridge allows MQ messages received on request queues to be formatted and passed on to legacy applications as if they came from ordinary terminal devices. The output from the applications is routed from the ALCS output routines to the MQ Bridge, in order to send it to corresponding response queues.

Minimal changes are required to ALCS. The MQ request queues are known to ALCS through communication definitions. *MQ terminals* are also known to ALCS through communication definitions, and they are associated with the queue definitions. In this way, applications have normal terminal records and addresses to deal with.

This support is intended to provide connectivity to current ALCS applications from remote systems (for example, web servers).

Exits are provided on both the input and output sides of the bridge to allow for user decoding and reformatting of messages.

We advise you to review the following information about the MQ Bridge function in case you want to exploit this new feature:

- Installation-wide monitor exits USRMQB0, USRMQB1, USRMQB2, and USRMQB3 in "MQ input bridge address exit - USRMQB0" on page 273.
- COMDEF generation macro in "COMDEF macro" on page 107.

(MQ Bridge support was added to ALCS 2.2.1 by PTF.)

ALCS data collection - CPU utilization

Your new ALCS can collect information about CPU utilization per ECB.

During the life of an entry, the ALCS online monitor can optionally accumulate statistics about CPU utilization for the entry. The accumulated CPU utilization is an estimate of the total elapsed CPU time while the entry has control in ECB-controlled program code or monitor-request macros or C functions. It does not include CPU time when the entry loses control (for example DASD I/O, sequential file I/O, or communication I/O).

If data collection is active, the statistics for CPU utilization are included in the data collection record written to the data-collection sequential file when the ECB exits. If a data-collection file is not defined, ALCS writes the information to the diagnostic file.

The ALCS statistical report generator (SRG) has not been enhanced to process the statistics for CPU utilization. Instead, ALCS includes a sample offline program (DXCCTR) which you can use to process the statistics for CPU utilization. Comments at the start of the program describe how it works.

We advise you to review the following information about data collection in case you want to exploit this new feature:

- ZDCLR command in *ALCS Operation and Maintenance*.
- SCTGEN generation macro in "SCTGEN macro" on page 72.
- DCLOG macro in *ALCS Application Programming Reference - Assembler*.

(This support was added to ALCS 2.2.1 by PTF.)

End users, operations, and programmed operators

Secure access control - logon dialogs

Your new ALCS implements secure access control. This allows you to enforce a logon for selected terminals. If you plan to do this, your end users may need an opportunity to familiarize themselves with the new logon dialogs for IBM 3270 and ALC terminal equipment.

Extended CRAS authorization

Your new ALCS allows terminals other than CRASs to obtain CRAS authority. For example, you can have several terminals authorized to enter Prime CRAS only commands. Your operations staff may need an opportunity to familiarize themselves with this new capability.

Acquiring CRAS terminals on ALCS startup

Your new ALCS allows the system to be started without any CRAS terminals. For example, the Prime CRAS, RO CRAS and alternate (1-16) CRAS terminals may not be acquired during ALCS startup because:

- They are not defined in the ALCS communication generation
- They are defined with `ISTATUS=INACTIVE` in the ALCS communication generation
- They are defined with `ISTATUS=ACTIVE` but are not available.

If so then ALCS will complete the startup without those terminals.

If RO CRAS is not defined in the ALCS communication generation, you will not be able to activate some of the ALCS maintenance functions. For example, you will not be able to:

- Start the ALCS recoup function (ZRECP)
- Start the ALCS system test vehicle (ZTEST)
- Drive ECB-controlled programs (ZDRIV)

When you do not have a physical printer available for RO CRAS, you should consider defining your RO CRAS in the communication generation as an STV printer or as a NetView operator ID.

PC file transfer

Your old ALCS allowed an end user to upload files from a 3270 display terminal to ALCS using the PC file transfer protocol. Your new ALCS also allows files to be downloaded from ALCS to a 3270 display terminal.

PC file transfer can be invoked using the PC SEND and RECEIVE commands, or by using the **file transfer** feature of many 3270 terminal emulators.

New and enhanced ALCS commands

ZACOM
Enhanced to support new parameters and to provide new functions.

ZASER, ZDSER
Enhanced for exception dump table.

ZCTCP
Implements the TCP/IP functions (added to ALCS 2.1.3 by PTF).

ZDASD
Enhanced for database restripe.

ZDATA
Enhanced to dump allocatable pool file addresses.

ZDCLR
Enhanced to support new parameters and to display additional information.

ZDCOM
Enhanced to support new parameters and to display additional information.

ZDECB
Enhanced to show program information for TPFDF application.

ZDRIV
Enhanced to support new parameters.

ZDPDU
Implements emergency pool recovery (PDU) function.

ZHELP
Enhanced for context-sensitive help.

ZMAIL
Implements the e-mail function.

ZPOOL
Enhanced for short-term pool event logging and record ID counting.

ZPURG
Enhanced for VFA purge function.

ZRETR
Implements the command retrieve function.

ZSSEQ
Enhanced for record logging.

ZSTAT
Enhanced to display peak statistics.

ZTRAC
Enhanced to support remote terminal trace, asynchronous trace, workstation trace, and REGSTOP/ANYSTOP trace commands.

ALCS commands, including new and enhanced commands, are described in *ALCS Operation and Maintenance*.

Appendix A. Sample code for installation-wide exit program APR5

When ALCS sends a message to a printer terminal, you can use ECB-controlled installation-wide exit program APR5 to copy the message to another destination. This can be useful when you are using STV printers defined on a test system, but you want to see the printer messages in real time rather than after post-processing the diagnostic sequential file.

This section shows a sample APR5 program which copies printer messages to two different destinations:

- A sequential file defined as a system output (SYSOUT) data set. Messages to this destination can be seen in real time in the ALCS job output.
- A TCP/IP connection defined in the ALCS communication tables. Messages to this destination can be seen in real time if the remote socket program is coded appropriately.

```
         BEGIN NAME=APR5,VERSION=04,TYPE='IPARS',AMODE=31,             -
               XCL=NONE,SHR=NONE
         SPACE 1
*=====================================================================*
*        ALCS PRINTER SPOCC EXIT FOR APPLICATION                      *
*=====================================================================*
*        APR5 ENTERED BY ALCS PRINTING ROUTINES                       *
*        JUST BEFORE EXECUTING THE SPOCC MACRO TO TRANSMIT THE DATA   *
*        TO THE PRINTER                                               *
*                                                                     *
*        ON ENTRY                                                     *
*        --------                                                     *
*                                                                     *
*        EBX000/000            TYPE OF SPOCC TO BE EXECUTED           *
*                 X'00'        SLMTC MACRO DIRECT TRANSMISSION        *
*                 X'01'        ROC DIRECT TRANSMISSION ON LOW POOL    *
*                 X'02'        NORMAL QUEUED TRANSMISSION             *
*                 X'03'        RETRY  TRANSMISSION (SENDCL OR CRASC)  *
*                 X'04'        TIMEOUT TRANSMISSION (TEST MESSAGE)    *
*                 X'05'        WELCOME MESSAGE FOR SHARED PRINTERS    *
*                                                                     *
*        EBX001/001            INDICATORS                             *
*                 X'80'        FIRST BLOCK OF MESSAGE                 *
*                 X'40'        LAST BLOCK OF MESSAGE                  *
*                 X'20'        RESERVED FOR FUTURE USE                *
*                 X'10'        RESERVED FOR FUTURE USE                *
*                 X'08'        RESERVED FOR FUTURE USE                *
*                 X'04'        RESERVED FOR FUTURE USE                *
*                 X'02'        RESERVED FOR FUTURE USE                *
*                 X'01'        RESERVED FOR FUTURE USE                *
*                                                                     *
*        EBX002/003            RETRY COUNT (FOR TYPE X'03')           *
*                              LONG TIMEOUT COUNT (FOR TYPE X'04')    *
*                                                                     *
*        EBX004/007            PRIORITY QUEUE NUMBER OF PRINTOUT      *
*                                                                     *
*        ECB STORAGE LEVEL 3 = MESSAGE TO BE TRANSMITTED IN           *
*                              CM1CM FORMAT                           *
*                                                                     *
*---------------------------------------------------------------------*
         EJECT ,
*---------------------------------------------------------------------*
*                                                                     *
*        NOTES:                                                       *
*        1.THE USER ROUTINE MAY ONLY USE FIELDS EBX000/103.           *
*          THE DATA SUPPLIED TO THE USER EXIT ON EBX000/103 MAY BE    *
*          OVERWRITTEN WITHOUT SAVING.                                *
*        2.THE USER ROUTINE MAY ALSO USE                              *
*          REGISTERS R00-R07,R14,R15.                                 *
*        3.THE USER MUST NOT RELEASE THE MESSAGE ON LEVEL 3.          *
*        4.THE USER MAY USE ANY LEVELS AS LONG AS THEY ARE            *
*          SAVED WITH A DETAC TYPE=STACK AND RESTORED WITH A          *
*          ATTAC TYPE=STACK BEFORE RETURNING TO THE CALLING PROGRAM.  *
*        5.THE USER MUST NOT MAKE USE OF ANY SERRC R/E MACRO.         *
*                                                                     *
```

© Copyright IBM Corp. 2003, 2019

```
*                                                                     *
*        ON RETURN                                                    *
*        ---------                                                    *
*                                                                     *
*        APR5 MUST RETURN TO THE CALLING PROGRAM (BACKC)              *
*                                                                     *
*---------------------------------------------------------------------*
         EJECT ,
*-------------------------------------------------------------**APR5**
* IMPLEMENTATION.                                            **APR5**
*                                                            **APR5**
* 1. SHADOW ROC TO SYSOUT                                    **APR5**
* ======================                                     **APR5**
* IF THE OUTPUT MESSAGE IS FOR RO CRAS, THIS EXIT WRITES A   **APR5**
* COPY TO THE REALTIME SEQUENTIAL FILE 'ROC', IF ONE EXISTS. **APR5**
*                                                            **APR5**
* IF A REALTIME SEQUENTIAL FILE 'ROC' IS NOT DEFINED IN THE  **APR5**
* SEQUENTIAL FILE TABLE, THIS EXIT RETURNS TO CALLER.        **APR5**
*                                                            **APR5**
* TO ACTIVATE THIS EXIT, LOAD IT (ZPCTL L S) OR ADD THE NAME **APR5**
* OF ITS LOAD MODULE TO THE PROGRAM LIST. YOU MUST ALSO DEFINE **APR5**
* THE FOLLOWING REALTIME SEQUENTIAL FILE:                    **APR5**
*                                                            **APR5**
*        SEQGEN NAME=ROC,                                    **APR5**
*             TYPE=REALTIME,                                 **APR5**
*             SYSOUT=X,         ANY CLASS - PREFERABLY HELD  **APR5**
*             BLKSIZE=132,                                   **APR5**
*             SPACE=(5000,5000),                             **APR5**
*             RECFM=F                                        **APR5**
*                                                            **APR5**
* NOTE: ALWAYS HAVE A LOAD MODULE WITH A "DUMMY" VERSION OF  **APR5**
*       APR5 READY FOR LOADING IN CASE THIS EXIT FAILS.      **APR5**
*       THE DUMMY EXIT SHOULD ONLY CONTAIN:                  **APR5**
*                                                            **APR5**
*        SR    R15,R15            SET ZERO RETURN CODE      **APR5**
*        BACKC ,                  RETURN TO CALLER          **APR5**
*                                                            **APR5**
*                                                            **APR5**
* 2. SHADOW ROC TO A TCP/IP CLIENT                           **APR5**
* ================================                           **APR5**
* THE TCP/IP CLIENT CAN BE A JAVA APPLICATION THAT DISPLAYS  **APR5**
* ROC OUTPUT ON A WINDOW.                                    **APR5**
*                                                            **APR5**
* THIS EXIT SENDS THE ROC MESSAGES (USING ROUTC) TO A TCP/IP **APR5**
* CLIENT CONNECTED TO AN ALCS SERVER DEFINED WITH CRN=ROSHADOW **APR5**
* AND LISTENING ON PORT 55766:                               **APR5**
*                                                            **APR5**
*      COMDFLT CLEAR                                         **APR5**
*      COMDFLT LDTYPE=TCPIP                                  **APR5**
*      COMDEF NAME=ROSHADOW,TERM=SERVER,APPL=SERC,LPORT=55766 **APR5**
*                                                            **APR5**
*-------------------------------------------------------------**APR5**
         EJECT ,
EB0EB    DSECT ,
******** THE DATA SETUP HERE DESCRIBES THE INTERFACE
         ORG   EBX000
APR5TYPE DS    XL1
APR5SLMT EQU   X'00'              SLMTC TYPE DIRECT TRANSMISSION
APR5ROC  EQU   X'01'              ROC LOW POOL TRANSMISSION
APR5NORM EQU   X'02'              NORMAL TRANSMISSION
APR5RETR EQU   X'03'              RETRY TRANSMISSION
APR5TIME EQU   X'04'              TIMEOUT RETRANSMISSION
APR5WELC EQU   X'05'              WELCOME TRANSMISSION (SHARED PRT)
APR5IND  DS    BL1
         ORG   APR5IND
APR5FRST DS    0XL(B'10000000')    FIRST PART OF THE MESSAGE
APR5LAST DS    0XL(B'01000000')    LAST PART OF THE MESSAGE
         DS    0XL(B'00100000')    RESERVED FOR FUTURE USE
         DS    0XL(B'00010000')    RESERVED FOR FUTURE USE
         DS    0XL(B'00001000')    RESERVED FOR FUTURE USE
         DS    0XL(B'00000100')    RESERVED FOR FUTURE USE
         DS    0XL(B'00000010')    RESERVED FOR FUTURE USE
         DS    0XL(B'00000001')    RESERVED FOR FUTURE USE
         ORG   APR5IND+L'APR5IND
APR5CNT  DS    XL2                 RETRY COUNT
APR5PRIO DS    XL4                 PRIORITY QUEUE NUMBER
APR5LEN  EQU   *-EBX000            LENGTH TO BE MOVED
         ORG   EBX000+APR5LEN                                 **APR5**
TCPCRI   DS    XL3                 CRI FOR COMIC              **APR5**
TCPRCPL  DS    XL(RCPLSIZE)        RCPL GOES IN HERE          **APR5**
TCPCOMIC DS    XL(ICELN2)          COMIC DATA AREA            **APR5**
         ORG   ,                                              **APR5**
```

```
         RSECT ,
         SPACE 1                                                **APR5**
         TRMEQ ,                                                **APR5**
         EJECT ,
APR5     EQU   *
         L     R03,CE1CR3      ADDRESS OUTPUT MESSAGE           **APR5**
         CM1CM REG=R03                                          **APR5**
         SPACE 1                                                **APR5**
         COMIC CRI=CM1CRI,AREA=(0,ICELN2) OBTAIN RESOURCE DATA **APR5**
         CO0IC REG=R14                    COMIC DATA AREA       **APR5**
         TM    ICESROC,L'ICESROC          IS IT RO CRAS?        **APR5**
         BZ    APR5990                    NO, RETURN TO CALLER **APR5**
         SPACE 1                                                **APR5**
         LEVTA LEVEL=D7,NOTUSED=APR5600  IF LEVEL D7 IN USE    **APR5**
         DETAC D7                        DETACH IT             **APR5**
         SPACE 1                                                **APR5**
         BAS   R06,APR5SEQ     COPY MESSAGE TO SEQ FILE         **APR5**
         BAS   R06,APR5TCP     COPY MESSAGE TO TCP/IP CLIENT    **APR5**
         ATTAC D7              RESTORE LEVEL D7                 **APR5**
         B     APR5990         RETURN TO CALLER                 **APR5**
         SPACE 1                                                **APR5**
APR5600  EQU   *                                                **APR5**
         BAS   R06,APR5SEQ     COPY MESSAGE TO SEQ FILE         **APR5**
         BAS   R06,APR5TCP     COPY MESSAGE TO TCP/IP CLIENT    **APR5**
         B     APR5990         RETURN TO CALLER                 **APR5**
         SPACE 1                                                **APR5**
APR5990  EQU   *                                                **APR5**
         SR    R15,R15         SET TO NORMAL COMPLETION         **APR5**
         BACKC ,               RETURN TO  CALLING PROGRAM       **APR5**
         DROP  R14             CO0IC                            **APR5**
         EJECT ,                                                **APR5**
*=============================================================**APR5**
*        SUBROUTINE TO COPY ROC MESSAGE TO SEQUENTIAL FILE      **APR5**
*=============================================================**APR5**
         SPACE 1                                                **APR5**
APR5SEQ  EQU   *                                                **APR5**
         TDSPC NAME=ROC,LEVEL=D7  CHECK SEQUENTIAL FILE         **APR5**
         OC    CE1FA7(8),CE1FA7   IS SEQ FILE DEFINED ?         **APR5**
         BZR   R06                NO, RETURN                    **APR5**
         SPACE 1                                                **APR5**
         TM    CE1FA7+1,B'00010000' IS IT A REALTIME SEQ FILE ?**APR5**
         BZR   R06                  NO, RETURN                  **APR5**
         SPACE 1                                                **APR5**
         XC    CE1FA7(8),CE1FA7    CLEAR DATA LEVEL             **APR5**
         LA    R14,CM1TXT          POINT TO START OF MSG TEXT   **APR5**
         LR    R04,R14             COPY START OF TEXT           **APR5**
         AH    R04,CM1CCT          POINT TO END OF MESSAGE      **APR5**
         SH    R04,=AL2(3+1+1+1)   LESS CRI, CMD-A, CMD-B, #EOM**APR5**
         SPACE 1                                                **APR5**
******** SCAN MESSAGE FOR #CAR                                  **APR5**
         SPACE 1                                                **APR5**
APR5S20  EQU   *                                                **APR5**
         ST    R14,CE1FA7          SAVE ADDRESS FOR TOUTC       **APR5**
         SPACE 1                                                **APR5**
APR5S30  EQU   *                                                **APR5**
         CLI   0(R14),#CAR         END OF LINE?                 **APR5**
         BE    APR5S60             YES, WRITE TO SEQ FILE       **APR5**
         SPACE 1                                                **APR5**
         CR    R14,R04             END OF MESSAGE?              **APR5**
         BNL   APR5S60             YES, WRITE TO SEQ FILE       **APR5**
         SPACE 1                                                **APR5**
         LA    R14,1(,R14)         NEXT CHARACTER               **APR5**
         B     APR5S30             GO CHECK IF #CAR             **APR5**
         SPACE 1                                                **APR5**
APR5S60  EQU   *                                                **APR5**
         LR    R00,R14             COPY LAST POSITION           **APR5**
         S     R00,CE1FA7          CALCULATE LINE LENGTH        **APR5**
         STH   R00,CE1FA7+6        SET UP FOR TOUTC             **APR5**
         SPACE 1                                                **APR5**
         TOUTC NAME=ROC,LEVEL=D7   WRITE THE RECORD             **APR5**
         WAITC APR5ERR             WAIT FOR WRITE TO COMPLETE   **APR5**
         SPACE 1                                                **APR5**
         L     R14,CE1FA7          POINT TO START OF LAST LINE **APR5**
         AH    R14,CE1FA7+6        ADD LINE LENGTH              **APR5**
         LA    R14,1(,R14)         PAST #CAR                    **APR5**
         CR    R14,R04             END OF MESSAGE?              **APR5**
         BL    APR5S20             NO, GO FIND NEXT LINE        **APR5**
         SPACE 1                                                **APR5**
         XC    CE1FA7(8),CE1FA7    CLEAR DATA LEVEL             **APR5**
         BR    R06                 RETURN TO CALLER             **APR5**
         EJECT ,                                                **APR5**
******** ERROR WRITING TO SEQ FILE                              **APR5**
```

```
         SPACE 1                                                **APR5**
APR5ERR  EQU   *                                                **APR5**
*******  NOTE THAT WE CANNOT SEND A MESSAGE TO ROC OR           **APR5**
*******  GENERATE A SYSTEM ERROR DUMP, BECAUSE THIS EXIT        **APR5**
*******  WILL BE ENTERED RECURSIVELY SO WE SIMPLY               **APR5**
*******  IGNORE THE ERROR AND RETURN TO CALLER                  **APR5**
         XC    CE1FA7(8),CE1FA7    CLEAR DATA LEVEL             **APR5**
         BR    R06                 RETURN TO CALLER             **APR5**
         EJECT ,                                                **APR5**
*================================================================**APR5**
*        SUBROUTINE TO COPY ROC MESSAGE TO TCP/IP CLIENT(S)     **APR5**
*================================================================**APR5**
         SPACE 1                                                **APR5**
APR5TCP  EQU   *                                                **APR5**
         COMIC CRN='ROSHADOW',    FIND THE COMMS TABLE ENTRY    **APR5**-
               DATA=SYS,             FOR THE TCPIP SERVER      **APR5**-
               AREA=(TCPCOMIC,0)                               **APR5**
         BCR   B'0110',R06        NOT FOUND, RETURN TO CALLER  **APR5**
         SPACE 1                                                **APR5**
         LR    R07,R14            USE R07 FOR CO0RE            **APR5**
         CO0RE REG=R07            COMMS TABLE ENTRY            **APR5**
         TM    REC1CST,L'REC1CST  IS RESOURCE ACTIVE           **APR5**
         BZR   R06                NO, RETURN TO CALLER         **APR5**
         SPACE 1                                                **APR5**
         CLI   RECODV1,TPTCPIP    IS THIS A TCP/IP RESOURCE    **APR5**
         BNER  R06                NO, RETURN TO CALLER         **APR5**
         SPACE 1                                                **APR5**
         CLC   REIPLPO,ROCPORT    LISTENING ON CORRECT PORT    **APR5**
         BNER  R06                NO, RETURN TO CALLER         **APR5**
         SPACE 1                                                **APR5**
APR5T01  EQU   *                                                **APR5**
         ICM   R07,B'1111',REIPNSC GET RESOURCE ENTRY OF CLIENT**APR5**
         BZR   R06                NO (MORE) CLIENTS, RETURN    **APR5**
         SPACE 1                                                **APR5**
         TM    REC1CST,L'REC1CST  IS RESOURCE ACTIVE           **APR5**
         BZ    APR5T01            NO - SEE IF ANY MORE         **APR5**
         EJECT ,                                                **APR5**
******** WE FOUND AN ACTIVE CONNECTION ON THE RIGHT PORT       **APR5**
******** SEND A COPY OF THE MESSAGE USING ROUTC                **APR5**
         SPACE 1                                                **APR5**
         LH    R04,CM1CCT          LOAD CHARACTER COUNT       **APR5**
         LA    R04,16+2(,R04)      ADD HEADER AND COUNT ITSELF **APR5**
*        THE FOLLOWING ENSURES THAT THE SIZE PASSED TO GETCC    **APR5**
*        DOES NOT END WITH X'1' WHICH COULD BE CONFUSED WITH    **APR5**
*        A SIZE CODE.                                           **APR5**
         LA    R04,1(,R04)         ADD ONE                    **APR5**
         SRL   R04,1               SHIFT OUT LAST BIT         **APR5**
         SLL   R04,1               AND SHIFT BACK             **APR5**
         SPACE 1                                                **APR5**
         GETCC LEVEL=D7,SIZE=(R04)                              **APR5**
         LR    R04,R03             USE ORIGIN ADDRESS FOR MVCL **APR5**
         LH    R05,CM1CCT          LOAD ORIGIN LENGTH FOR MVCL **APR5**
         LA    R05,16+2(,R05)      ADD HEADER AND COUNT        **APR5**
         LR    R15,R05             DESTINATION SIZE FOR MVCL   **APR5**
         MVCL  R14,R04             COPY MESSAGE                **APR5**
         SPACE 1                                                **APR5**
         L     R04,CE1CR7          RELOAD MESSAGE BLOCK ADDRESS**APR5**
         CM1CM REG=R04,SUFFIX=R    COPY OF OMSG FOR ROUTC      **APR5**
         MVC   CM1CRIR,RECOCRI+1   COPY CRI OF CONNECTION      **APR5**
         SPACE 1                                                **APR5**
******** BUILD RCPL                                            **APR5**
         SPACE 1                                                **APR5**
         RC0PL REG=R05             RCPL                       **APR5**
         LA    R05,TCPRCPL         POINT TO RCPL AREA IN EBX  **APR5**
         XC    RC0PL(RCPLSIZE),RC0PL CLEAR RCPL               **APR5**
         MVC   RCPLDESX,RECOCRI+1  MOVE IN DESTINATION CRI    **APR5**
         MVC   RCPLORGX,EBROUT     MOVE IN ORIGIN CRI         **APR5**
         OI    RCPLCTL2,RCPL2MFT   MESSAGE FORMAT = NOT AM0SG **APR5**
         OI    RCPLCTL2,RCPL2REL   RELEASE MESSAGE BLOCK      **APR5**
         SPACE 1                                                **APR5**
******** SEND MESSAGE TO DEST                                  **APR5**
         SPACE 1                                                **APR5**
         ROUTC LEV=D7,RCPL=RC0PL   ROUTE MESSAGE              **APR5**
         SPACE 1                                                **APR5**
         DROP  R04                 CM1CM                      **APR5**
         DROP  R05                 RC0PL                      **APR5**
         DROP  R07                 CO0RE                      **APR5**
         SPACE 1                                                **APR5**
         B     APR5T01             GO TRY NEXT SOCKET CONNECTION**APR5**
         BR    R06                 RETURN TO CALLER           **APR5**
         EJECT ,
*-----------------------------------------------------------------*
```

```
*        PROGRAM CONSTANTS                                             *
*--------------------------------------------------------------------*
         SPACE 1
******** TCP/IP PORT NUMBER FOR ROC SHADOW SOCKET             **APR5**
         SPACE 1                                             **APR5**
ROCPORT  DC    AL2(55766)                                    **APR5**
         LTORG ,
         SPACE 1
         FINIS ,
         SPACE 1
         END   ,
```

Appendix B. Register numbers and symbolic names

General-use Programming Interface

The following tables list the registers and their corresponding symbolic names.

In the ALCS books, general registers are usually referred to by number, with the older register name in parentheses, for example:

general register 0 (RAC)

When you are writing a new program, IBM recommends that you use the new form, R00 and so on. (This form, as against R0, ensures that registers are in sequence in the cross-reference listing produced by the assembler.)

The alternative symbol is shown merely as a reminder to programmers who are working with existing TPF or ALCS programs that use the older symbols RAC and so on. These symbols are perfectly valid, and if you are modifying such a program, it is better to maintain consistency within the program or set of programs. This makes your programs easier to understand and makes the cross-reference listing produced by the assembler more useful.

Table 37. Notation convention for general registers

General register	Recommended notation	Alternative notation conventions			Notes
0	R00	R0	RAC	RG0	
1	R01	R1	RG1	RG1	
2	R02	R2	RGA	RG2	
3	R03	R3	RGB	RG3	
4	R04	R4	RGC	RG4	
5	R05	R5	RGD	RG5	
6	R06	R6	RGE	RG6	
7	R07	R7	RGF	RG7	
8	R08	R8	RAP	RG8	Application program base
9	R09	R9	REB	RG9	Entry control block base
10	R10	R10	RLA	RG10	Reserved
11	R11	R11	RLB	RG11	Reserved
12	R12	R12	RLC	RG12	Reserved
13	R13	R13	RLD	RG13	Reserved
14	R14	R14	RDA	RG14	
15	R15	R15	RDB	RG15	

TPF compatibility:

If your program must be compatible with TPF, do not use the register symbols RG10 - RG15. TPF does not support these symbols.

Floating-point registers

Table 38 on page 549 shows notation conventions for the floating-point registers.

Table 38. Notation conventions for floating-point registers

Floating-point register	Notation
0	FP0
2	FP2
4	FP4
6	FP6

TPF compatibility:

If your program must be compatible with TPF, do not use the register symbols RG10 - RG15. TPF does not support these symbols.

End of General-use Programming Interface

Appendix C. Sample symbolic line number conversion

General-use Programming Interface

Figure 123 on page 550 shows how you can convert the CRI of an X.25 PVC to the SLN of a virtual SLC link.

```
         CO0IC REG=R04
         LA    R04,EBX000              EBX000 = COMIC DATA AREA
         MVC   EBW000(3),X.25_pvc_cri  MOVE IN CRI OF X25 PVC
         COMIC CRI=EBW000,             GET X25 PVC DATA            -
               DATA=SYS,                                           -
               AREA=((R04),ICELEN)
         MVC   EBW000(1),ICEALC+3      MOVE IN SLN OF SLC LINK
```

Figure 123. Convert the CRI of an X.25 PVC to the SLN of a virtual SLC link

Figure 124 on page 550 shows how you can convert the SLN of a virtual SLC link to the CRI of an X.25 PVC.

```
         CO0IC REG=R04
         LA    R04,EBX000              EBX000 = COMIC DATA AREA
         COMIC CRN='SLCLINK',          GET SLC LINK HEADER DATA    -
               DATA=SYS,                                           -
               AREA=((R04),ICELEN)
         XC    EBW000(3),EBW000        CLEAR CRI BUILD AREA
         MVC   EBW000(1),ICELDI        MOVE IN LDI OF SLC LINK
         MVC   EBW002(1),SLC_link_sln  MOVE IN SLN OF SLC LINK
         COMIC CRI=EBW000,             GET VIRTUAL SLC LINK DATA   -
               DATA=SYS,                                           -
               AREA=((R04),ICELEN)
         MVC   EBW000(3),ICEALC+1      MOVE IN CRI OF X25 PVC
```

Figure 124. Convert the SLN of a virtual SLC link to the CRI of an X.25 PVC

End of General-use Programming Interface

Appendix D. Sample logon mode table

"Sample logon mode entries" on page 551 shows an example VTAM logon mode table for ALCS.

IBM supplies a default logon mode table (ISTINCLM) with VTAM, that provides generally accepted session protocols for a basic list of IBM device types. These include display and printer terminals of different model types, connected through SNA and non-SNA control units.

"Sample logon mode entries" on page 551 provides some additional sample logon mode entries that can be used with ALCS V2. The general form used in these examples is:

```
MODEENT LOGMODE=logmode
```

where *logmode* is a generic name for a device type. You can use any suitable name for *logmode*. In this particular example the names are of the general form:

```
MODEENT LOGMODE=DXCLOGtype
```

Where *type* can be:

S*n*
Screen definition

P*n*
Printer definition

W
World Trade Teletype

A
ALCI

X
X.25 PVC

See the *VTAM Customization Manual* for more information about creating or modifying logon mode tables.

See the *VTAM Resource Definition Reference* for more information about the PSERVIC parameter.

Sample logon mode entries

```
*------------------------------------------------------------------*
*        Sample logon mode entries that can be used with ALCS V2.  *
*------------------------------------------------------------------*
*
*
DXCLGTB  MODETAB ,
*********************************************************************
*                                                                   *
* 3270-type display (SNA LU Type 2)                                 *
*                                                                   *
* Primary display size 12 x 40 (480)                                *
* Alternate display size 12 x 80 (960)                              *
*                                                                   *
*********************************************************************
         MODEENT LOGMODE=DXCLOGS1,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'3080',
               RUSIZES=X'87F8',
               PSERVIC=X'02000000000000C280C507F00'
*********************************************************************
*
```

© Copyright IBM Corp. 2003, 2019

```
* 3270-type display (SNA LU Type 2)                                    *
*                                                                      *
* Primary display size 24 x 80 (1920)                                  *
* No alternate display size                                            *
*                                                                      *
************************************************************************
         MODEENT LOGMODE=DXCLOGS2,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'3080',
               RUSIZES=X'87F8',
               PSERVIC=X'020000000000185000007E00'
************************************************************************
*                                                                      *
* 3270-type display (SNA LU Type 2)                                    *
*                                                                      *
* Primary display size 24 x 80 (1920)                                  *
* Alternate display size 32 x 80 (2560)                                *
*                                                                      *
************************************************************************
         MODEENT LOGMODE=DXCLOGS3,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'3080',
               RUSIZES=X'87F8',
               PSERVIC=X'020000000000185020507F00'
************************************************************************
*                                                                      *
* 3270-type display (SNA LU Type 2)                                    *
*                                                                      *
* Primary display size 24 x 80 (1920)                                  *
* Alternate display size 43 x 80 (3440)                                *
*                                                                      *
************************************************************************
         MODEENT LOGMODE=DXCLOGS4,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'3080',
               RUSIZES=X'87F8',
               PSERVIC=X'02000000000018502B507F00'
************************************************************************
*                                                                      *
* 3270-type display (SNA LU Type 2)                                    *
*                                                                      *
* Display size 24 x 80 (1920)                                          *
*                                                                      *
************************************************************************
         MODEENT LOGMODE=DXCLOGS5,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'3080',
               RUSIZES=X'87F8',
               PSERVIC=X'020000000000000000000200'
         ************************************************************
         *                                                          *
         * 3270-type display (SNA LU Type 2)                        *
         *                                                          *
         * Dynamic logon mode                                       *
         * Default display size is 24 x 80 (1920)                   *
         * ALCS obtains the alternate display size from the         *
         * Query Reply (implicit partition size) structured field.  *
         *                                                          *
         ************************************************************
         MODEENT LOGMODE=DXCLOGS6,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'3080',
               RUSIZES=X'87F8',
               PSERVIC=X'028000000000000000000300'
************************************************************************
*                                                                      *
* 3270-type display (non-SNA)                                          *
```

```
*                                                                     *
* Display size 24 x 80 (1920)                                         *
*                                                                     *
***********************************************************************
         MODEENT LOGMODE=DXCLOGS7,
               FMPROF=X'02',
               TSPROF=X'02',
               PRIPROT=X'71',
               SECPROT=X'40',
               COMPROT=X'2000',
               RUSIZES=X'0000',
               PSERVIC=X'000000000000000000000202'
         ************************************************************
         *                                                           *
         * 3270-type display (non-SNA)                               *
         *                                                           *
         * Presentation Services usage field undefined               *
         * ALCS uses one of:                                         *
         *                                                           *
         * 3270-type display with display size 24 x 80 (1920)        *
         *   If COMDEF TERM=3270DSP is specified or TERM             *
         *   is omitted in ALCS communication generation             *
         *   for the device using this logon mode entry.             *
         *                                                           *
         * 3270-type printer with buffer size 480                    *
         *   If COMDEF TERM=3270PRT is specified in                  *
         *   ALCS communication generation for the                   *
         *   device using this logon mode entry.                     *
         *                                                           *
         ************************************************************
                  MODEENT LOGMODE=DXCLOGS8,
                        FMPROF=X'02',
                        TSPROF=X'02',
                        PRIPROT=X'71',
                        SECPROT=X'40',
                        COMPROT=X'2000',
                        RUSIZES=X'0000',
                        PSERVIC=X'000000000000000000000300'
***********************************************************************
*                                                                     *
* 3270-type printer (SNA LU Type 3)                                   *
*                                                                     *
* Primary buffer size 24 x 80 (1920)                                  *
* Alternate buffer size 24 x 80 (1920)                                *
*                                                                     *
***********************************************************************
         MODEENT LOGMODE=DXCLOGP1,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'3080',
               RUSIZES=X'87F8',
               PSERVIC=X'030000000000185018507F00'
***********************************************************************
*                                                                     *
* 3270-type printer (SNA LU Type 3)                                   *
*                                                                     *
* Primary buffer size 24 x 80 (1920)                                  *
* Alternate buffer size 43 x 80 (3440)                                *
*                                                                     *
***********************************************************************
         MODEENT LOGMODE=DXCLOGP2,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'90',
               COMPROT=X'3080',
               RUSIZES=X'87F8',
               PSERVIC=X'038000000000185028507F00'
***********************************************************************
*                                                                     *
* 3270-type printer (SNA LU Type 1)                                   *
*                                                                     *
* Primary buffer size 24 x 80 (1920)                                  *
* Alternate buffer size 43 x 80 (3440)                                *
*                                                                     *
***********************************************************************
         MODEENT LOGMODE=DXCLOGP3,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
```

```
               SECPROT=X'90',
               COMPROT=X'3080',
               RUSIZES=X'87F8',
               PSERVIC=X'018000000000018502B507F00'
*********************************************************************
*                                                                   *
* 3270-type printer (non-SNA)                                       *
*                                                                   *
* Buffer size undefined; ALCS uses 24 x 80 (1920)                   *
*                                                                   *
*********************************************************************
         MODEENT LOGMODE=DXCLOGP4,
               FMPROF=X'02',
               TSPROF=X'02',
               PRIPROT=X'71',
               SECPROT=X'40',
               COMPROT=X'2000',
               RUSIZES=X'0000',
               PSERVIC=X'000000000000000000000003'
*********************************************************************
*                                                                   *
* World Trade Teletype device (WTTY)                                *
*                                                                   *
*********************************************************************
         MODEENT LOGMODE=DXCLOGW,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'91',
               COMPROT=X'3040',
               RUSIZES=X'8585',
               TYPE=1,
               PSERVIC=X'010000000000000000000000'
*********************************************************************
*                                                                   *
* ALCI                                                              *
*                                                                   *
*********************************************************************
         MODEENT LOGMODE=DXCLOGA,
               FMPROF=X'02',
               TSPROF=X'03',
               PRIPROT=X'F0',
               SECPROT=X'90',
               COMPROT=X'0800',
               RUSIZES=X'F8F8',
               TYPE=0,
               PSERVIC=X'000000000000000000000000'
*********************************************************************
*                                                                   *
* X.25 PVC                                                          *
*                                                                   *
*********************************************************************
         MODEENT LOGMODE=DXCLOGX,
               FMPROF=X'03',
               TSPROF=X'03',
               PRIPROT=X'B1',
               SECPROT=X'91',
               COMPROT=X'3040',
               RUSIZES=X'F8F8',
               TYPE=1,
               PSERVIC=X'010000000000000000000000'
*********************************************************************
*                                                                   *
* ALCS application program node (capable of using LU6.1)            *
*                                                                   *
*********************************************************************
         MODEENT LOGMODE=ALCSPARS,
               FMPROF=X'12',
               TSPROF=X'04',
               PRIPROT=X'B1',
               SECPROT=X'B1',
               COMPROT=X'70A0',
               RUSIZES=X'F8F8',
               TYPE=0,
               PSERVIC=X'060038000000380000000000'
*
         MODEEND ,
         END   ,
```

Appendix E. LU 6.1 Communication generation

Generating an LU 6.1 link between ALCS and IMS/VS

Use the sample data in Figure 125 on page 555 and Figure 126 on page 555, in conjunction with the appropriate program product manuals, for generating an LU 6.1 link between ALCS and IMS/VS.

```
         COMDEF NAME=DY1ZIMS1,TERM=LU61,COMID=I
*
         COMDFLT TERM=PARSESS,LINK=DY1ZIMS1
*
         COMDEF NAME=LCCIMS01,RHSID=ALCSP201
         COMDEF NAME=LCCIMS02,RHSID=ALCSP202
         COMDEF NAME=LCCIMS03,RHSID=ALCSP203
         COMDEF NAME=LCCIMS04,RHSID=ALCSP204
         COMDEF NAME=LCCIMS05,RHSID=ALCSP205
         COMDEF NAME=LCCIMS06,RHSID=ALCSP206
         COMDEF NAME=LCCIMS07,RHSID=ALCSP207
         COMDEF NAME=LCCIMS08,RHSID=ALCSP208
         COMDEF NAME=LCCIMS09,RHSID=ALCSP209
         COMDEF NAME=LCCIMS10,RHSID=ALCSP210
```

Figure 125. Example of ALCS communication generation instructions

```
         COMM RECANY=(10,4096),                                X
               APPLID=DY1ZIMS1,                                X
               OPTIONS=(PAGING,TIMESTAMP,8000,FMTMAST,         X
               USERMSGS,VTAMAUTH,NOBLANK),                     X
               COPYLOG=MASTER,                                 X
               EDTNAME=ISCE
*
         TYPE  UNITYPE=LUTYPE6
         TERMINAL NAME=DY1ZALCY,                               X
               OPTIONS=(FORCESESS,NORELRQ),                    X
               SESSION=10,                                     X
               COMPT1=(SINGLE1,DPM-B1),                        X
               OUTBUF=3840,                                    X
               SEGSIZE=6000
*
         VTAMPOOL ,
         SUBPOOL NAME=ALCSP201
         NAME ALCSL201,COMPT=1
         SUBPOOL NAME=ALCSP202
         NAME ALCSL202,COMPT=1
         SUBPOOL NAME=ALCSP203
         NAME ALCSL203,COMPT=1
         SUBPOOL NAME=ALCSP204
         NAME ALCSL204,COMPT=1
         SUBPOOL NAME=ALCSP205
         NAME ALCSL205,COMPT=1
         SUBPOOL NAME=ALCSP206
         NAME ALCSL206,COMPT=1
         SUBPOOL NAME=ALCSP207
         NAME ALCSL207,COMPT=1
         SUBPOOL NAME=ALCSP208
         NAME ALCSL208,COMPT=1
         SUBPOOL NAME=ALCSP209
         NAME ALCSL209,COMPT=1
         SUBPOOL NAME=ALCSP210
         NAME ALCSL210,COMPT=1
```

Figure 126. Example of IMS/VS generation statements

© Copyright IBM Corp. 2003, 2019

Generating an LU 6.1 link between ALCS and CICS

Use the sample data in Figure 127 on page 556 and Figure 128 on page 556 in conjunction with the appropriate program product manuals, to generate an LU 6.1 link between ALCS and CICS.

In this example, two parallel sessions are used on the LU 6.1 link to avoid possible contention due to the half-duplex protocol. One session is used to receive messages and the other is used to send messages.

The RHSID name of the receive session for ALCS must be the name of the send session for CICS.

The RHSID of the send session for ALCS must be the name of the receive session for CICS.

These instructions define an LU 6.1 link and its parallel sessions.

```
         COMDFLT LDTYPE=ALCSLINK
         COMDEF NAME=PYEZ1CIC,TERM=LU61,COMID=I
         COMDFLT LDTYPE=ALCSLINK,TERM=PARSESS,LINK=PYEZ1CIC
         COMDEF NAME=PYEZ1CIS,RHSID=AL1R,SESSION=SEND
         COMDEF NAME=PYEZ1CIR,RHSID=AL1S,SESSION=RECEIVE
```

Figure 127. Example of ALCS communication generation instructions.

CICS requires one CONNECTION session and two SESSIONS definitions to describe the LU 6.1 link to ALCS.

In this example:

- The `CONNECTION(ALC1)` has a `NETNAME(PYEZ1AL)` that specifies the VTAM application name for ALCS.
- The receive session in CICS, `SESSNAME(AL1R)`, is associated with the send session in ALCS, `NETNAMEQ(PYEZ1CIS)`.
- The send session in CICS, `SESSNAME(AL1S)`, is associated with the receive session in ALCS, `NETNAMEQ(PYEZ1CIR)`.

```
DEFINE CONNECTION(ALC1) GROUP(ALCSLU61)
DESCRIPTION(LU61 CONNECTION TO ALCS - PYEZ1AL)
      NETNAME(PYEZ1AL) ACCESSMETHOD(VTAM) PROTOCOL(LU61)
      SINGLESESS(NO) DATASTREAM(USER) RECORDFORMAT(U) AUTOCONNECT(NO)
      INSERVICE(YES) ATTACHSEC(LOCAL) BINDSECURITY(NO)

DEFINE SESSIONS(PYEZ1CIR) GROUP(ALCSLU61)
DESCRIPTION(SEND PARALLEL SESSION TO ALCS)
      CONNECTION(ALC1) SESSNAME(AL1S) NETNAMEQ(PYEZ1CIR)
      PROTOCOL(LU61) MAXIMUM(0,0) SENDCOUNT(1) SENDSIZE(4096)
      RECEIVESIZE(4096) SESSPRIORITY(0) USERID(IALVS) AUTOCONNECT(NO)
      BUILDCHAIN(YES) USERAREALEN(0) IOAREALEN(4096,4096) RELREQ(NO)
      DISCREQ(NO) NEPCLASS(0) RECOVOPTION(SYSDEFAULT)
      RECOVNOTIFY(NONE)

DEFINE SESSIONS(PYEZ1CIS) GROUP(ALCSLU61)
DESCRIPTION(RECEIVE PARALLEL SESSION TO ALCS)
      CONNECTION(ALC1) SESSNAME(AL1R) NETNAMEQ(PYEZ1CIS)
      PROTOCOL(LU61) MAXIMUM(0,0) RECEIVECOUNT(1) SENDSIZE(4096)
      RECEIVESIZE(4096) SESSPRIORITY(0) USERID(IALVS) AUTOCONNECT(NO)
      BUILDCHAIN(YES) USERAREALEN(0) IOAREALEN(4096,4096) RELREQ(NO)
      DISCREQ(NO) NEPCLASS(0) RECOVOPTION(SYSDEFAULT)
      RECOVNOTIFY(NONE)
```

Figure 128. Example of CICS generation statements for LU 6.1

Appendix F. Network Control Program sample definition

This section shows a sample definition for a test network.

```
         OPTIONS USERGEN=(CXNNTO,X25NPSI,FNMNDFGN),                    C
               NEWDEFN=(YES,ECHO,NOSUPP)
*********************************************************************
* NCP/ALCI/X25/WTTY GENERATION FOR 3745-170 FOR PYE NETWORK          *
*********************************************************************
* USING:  NCP 5.2.1 LIBRARIES                                       *
*         SSP 3.4.1 LIBRARIES                                       *
*         ALCI      LIBRARIES                                       *
*         NTO       LIBRARIES                                       *
*         NPSI      LIBRARIES                                       *
*********************************************************************
* LIC1 LINES:                                                       *
*            000 - WTTY A                                           *
*            001 - WTTY B                                           *
*            008 - INN LINK TO SUBAREA 8                            *
*            012 - INN LINK TO SUBAREA 30                           *
*            016 - ALCI A2CS                                        *
*            017 - ALCI A2CS                                        *
*            018 - ALCI A2CS                                        *
*            019 - ALCI A2CS                                        *
* LIC5 LINES:                                                       *
*            080 - A2CS                                             *
*            081 - 3274                                             *
*            082 - RS/6000                                          *
*            083 - RS/6000                                          *
*            084 - X.25                                             *
*            085 - SPARE                                            *
*            086 - RS/6000                                          *
*            087 - RS/6000                                          *
*            088 - TOKEN RING                                       *
*            089 - TOKEN RING                                       *
*********************************************************************
*********************************************************************
* PCCU SUBAREA 29 - PYE MVS VTAM                                    *
*********************************************************************
         PCCU AUTODMP=NO,              OPERATION DESIGN               C
               AUTOIPL=NO,             OPERATION DESIGN               C
               AUTOSYN=YES,            OPERATION DESIGN               C
               CDUMPDS=SCANDUMP,       PYE                            C
               DUMPDS=COMDUMP,         PYE                            C
               DUMPSTA=PYEAP32,        37XX DUMP STATION              C
               LOADSTA=PYEAP32,        37XX LOAD STATION              C
               MAXDATA=6198,           PYE                            C
               MDUMPDS=MOSSDUMP,       PYE                            C
               NETID=GBIBMPYE,         PYE                            C
               OWNER=PYE,              PYE                            C
               RNAME=(PYEAP32,PYEDP32), PYEN3 & PYEND RNAMES          C
               SUBAREA=31,             PYE                            C
               VFYLM=YES               PERFORMANCE DESIGN
*********************************************************************
*********************************************************************
*  GENERATION FOR 3745                                              *
*********************************************************************
         BUILD ADDSESS=500,            BOUNDARY SESSION CONTROL BLKS  C
               AUXADDR=50,             DEFAULT                        C
               BFRS=240,               DEFAULT                        C
               BRANCH=100,             PROBLEM DETERMINATION          C
               CSMHDR=27F5C8,          OPERATION DESIGN               C
               CSMSG=5A5A40D9C5E3D9E840C1C6E3C5D940F340D4C9D5E4E3C5E2, C
               CWALL=26,               PERFORMANCE DESIGN             C
               DR3270=YES,             OPERATION DESIGN               C
               DSABLTO=3,              3865                           C
               ENABLTO=3,              3865                           C
               LOADLIB=NCPLIB,         PYE                            C
               LTRACE=8,               PROBLEM DETERMINATION          C
               MAXSESS=500,            MAX LU-LU SESSIONS             C
               MAXSSCP=8,              OPERATION DESIGN               C
               MAXSUBA=31,             PYE                            C
```

© Copyright IBM Corp. 2003, 2019

```
               MXRLINE=2,              I TIC                           C
               MXVLINE=200,            I TIC                           C
               MEMSIZE=4096,           HARDWARE                        C
               MODEL=3745-170,         HARDWARE                        C
               NAMTAB=30,              DEFAULT                         C
               NETID=GBIBMPYE,                                         C
               NEWNAME=PYENC,          PYE                             C
               NPA=(YES,DR),           PERF. ANALYSIS + D/RECONFIG     C
               NUMHSAS=6,              OPERATION DESIGN                C
               OLT=YES,                PYE                             C
               SESSACC=(YES,ALL,100,5000,1000,300,0,,,,),              C
               SLODOWN=12,             PYE                             C
               SUBAREA=29,             PYE                             C
               TGBXTRA=200,            DYN PATH UPDATE                 C
               TRACE=(YES,64),         PYE                             C
               TRANSFR=129,            PERFORMANCE DESIGN              C
               TYPGEN=NCP,             NCP                             C
               TYPSYS=MVS,             NCP                             C
               USGTIER=4,              NCP                             C
               VERSION=V5R2.1,                                         C
               VRPOOL=48,              OPERATION DESIGN                C
               X25.MAXPIU=5K,                                          C
               X25.MCHCNT=1,                                           C
               X25.PREFIX=A,                                           C
               X25.SNAP=YES,                                           C
               X25.USGTIER=4
*********************************************************************
* DYNAMIC CONTROL REQUIREMENTS                                      *
*********************************************************************
         SYSCNTRL OPTIONS=(MODE,RIMM,NAKLIM,SESSION,SSPAUSE,LNSTAT,RCNTC
               RL,RCOND,RECMD,ENDCALL,XMTLMT,BHSASSC,STORDSP) VTAM BSC
*********************************************************************
* USER-DEFINED NETWORK ADDRESSABLE UNIT - NTO                       *
*********************************************************************
PYENNTO  NCPNAU VIROWNER=CXNNTO,                                       C
               TYPE=SSCP,                                              C
               NAUCB=NTOSSCP,                                          C
               NAUFVT=CXNFVTN
*********************************************************************
* PATH DEFINITIONS                                                  *
*********************************************************************
         PATH  DESTSA=1,                                               C
               ER0=(8,1),ER1=(8,1),                                    C
               VR0=0,                                                  C
               VR1=1
         PATH  DESTSA=2,                                               C
               ER0=(30,1),ER1=(8,1),ER2=(8,1),                         C
               VR0=0,                                                  C
               VR1=2,                                                  C
               VR2=1
         PATH  DESTSA=3,                                               C
               ER0=(8,1),ER1=(8,1),                                    C
               VR0=0,                                                  C
               VR1=1
         PATH  DESTSA=8,                                               C
               ER0=(8,1),ER1=(8,1),ER3=(8,1),                          C
               VR0=0
         PATH  DESTSA=23,                                              C
               ER0=(8,1),                                              C
               VR0=0
         PATH  DESTSA=24,                                              C
               ER0=(8,1),ER1=(8,1),ER2=(8,1),                          C
               ER4=(30,1),ER6=(30,1),                                  C
               VR0=0,                                                  C
               VR1=2,                                                  C
               VR2=4,                                                  C
               VR3=6
         PATH  DESTSA=30,                                              C
               ER0=(30,1),ER1=(8,1),ER2=(8,1),                         C
               VR0=0,                                                  C
               VR1=1,                                                  C
               VR2=2
         PATH  DESTSA=31,                                              C
               ER0=(8,1),ER1=(8,1),ER2=(8,1),                          C
               ER4=(30,1),ER6=(30,1),                                  C
               VR0=0,                                                  C
               VR1=1,                                                  C
               VR2=4,                                                  C
               VR3=6
*********************************************************************
* SDLCST                                                            *
*********************************************************************
```

```
SDLC0    SDLCST MODE=PRI,MAXOUT=127,RETRIES=(3,10,5),                  C
               GROUP=PYEGPRI,SERVLIM=254,PASSLIM=127
SDLC1    SDLCST MODE=SEC,MAXOUT=127,                                   C
               GROUP=PYEGSEC,PASSLIM=127
*********************************************************************
* DYNAMIC RECONFIGURATION                                           *
*********************************************************************
         PUDRPOOL NUMBER=200
         LUDRPOOL NUMILU=100,NUMTYP1=100,NUMTYP2=200
*********************************************************************
* GROUP FOR WTTY LEASED LINES (REAL)                                *
*********************************************************************
PYECWTTY GROUP CRETRY=0,               T                              C
               DIAL=NO,                LEASED                         C
               ISTATUS=INACTIVE,       V SYSTEM DESIGN                C
               LNCTL=SS,               WTTTY                          C
               PADCNT=10,                                             C
               TEXTTO=28,              PYE                            C
               WTTYEOB=1B1E,           EOB                            C
               WTTYEOT=0C0C0C0C        EOT (NNNN)
*********************************************************************
* NTO REAL LINE ADDRESS 00  FDX                                     *
*********************************************************************
PYEL00   LINE ADDRESS=(000,HALF),      ADDRESS                        C
               CLOCKNG=INT,            SCANNER CLOCKING               C
               CODE=ITA2,              WTTTY                          C
               CRRATE=69,              WTTTY                          C
               DUPLEX=HALF,            NTO                            C
               FEATURE=IMEND,          PYE                            C
               ISTATUS=INACTIVE,       VTAM - REQUIRED FOR NTO        C
               LINESIZ=69,             WTTTY                          C
               MONITOR=YES,            WTTTY                          C
               POLLED=NO,              TWX                            C
               TRANSFR=47,             TWX                            C
               SPEED=300               TWX DESIGN
*              STATOPT=('TTY/A REAL LNE',NOMONIT)
*********************************************************************
* NTO REAL TERMINAL WTTY                                            *
*********************************************************************
PYEY00   TERMINAL DIRECTN=INOUT,       WTTTY                          C
               ENDTRNS=EOT,              WTTTY FEATURE                C
               FEATURE=(NOATTN,NOBREAK), WTTTY FEATURE                C
               INHIBIT=(TIMEFILL,ERPR,ERPW), WTTTY                    C
               LGRAPHS=(REJECT,REJECT),  WTTTY FEATURE                C
               TERM=WTTY               WTTTY
*              STATOPT=('TTY/A REAL TRM',NOMONIT)
*********************************************************************
* NTO REAL LINE ADDRESS 01  FDX                                     *
*********************************************************************
PYEL01   LINE ADDRESS=(001,HALF),      ADDRESS                        C
               CLOCKNG=INT,            SCANNER CLOCKING               C
               CODE=ITA2,              WTTTY                          C
               CRRATE=69,              WTTTY                          C
               DUPLEX=HALF,            NTO                            C
               FEATURE=IMEND,          PYE                            C
               ISTATUS=INACTIVE,       VTAM - REQUIRED FOR NTO        C
               LINESIZ=69,             WTTTY                          C
               MONITOR=YES,            WTTTY                          C
               POLLED=NO,              TWX                            C
               TRANSFR=47,             TWX                            C
               SPEED=300               TWX DESIGN
*              STATOPT=('TTY/B REAL LNE',NOMONIT)
*********************************************************************
* NTO REAL TERMINAL WTTY                                            *
*********************************************************************
PYEY01   TERMINAL DIRECTN=INOUT,       WTTTY                          C
               ENDTRNS=EOT,              WTTTY FEATURE                C
               FEATURE=(NOATTN,NOBREAK), WTTTY FEATURE                C
               INHIBIT=(TIMEFILL,ERPR,ERPW), WTTTY                    C
               LGRAPHS=(REJECT,REJECT),  WTTTY FEATURE                C
               TERM=WTTY               WTTTY
*              STATOPT=('TTY/B REAL TRM',NOMONIT)
*********************************************************************
* GROUP FOR WTTY LEASED LINES (VIRTUAL)                             *
*********************************************************************
PYECVTY  GROUP DIAL=NO,                LEASED                         C
               LNCTL=SDLC,             SDLC                           C
               SSCPFM=USSNTO,          NTO                            C
               USSTAB=USSNTO,          NTO                            C
               DLOGMOD=INTERACT,       PYE                            C
               MODETAB=CWTTY,          PYE                            C
               VIRTUAL=YES,            NTO                            C
```

```
               VIROWNER=CXNNTO         NTO
*********************************************************************
* NTO VIRTUAL LINE 00                                               *
*********************************************************************
PYELV00  LINE NTO.SSLINE=PYEL00,NTO.TRALL=YES,ISTATUS=ACTIVE,          C
               LINECB=CXNL1,                                          C
               LINEFVT=CXNFVTV
*              STATOPT=('TTY/A VIRT LNE',NOMONIT)
*
PYEC00   PU NTO.SSTERM=PYEY00,                                         C
               NTO.EDIT=00,           ATTENTION CHARACTER             C
               NTO.WTTYTO=YES,                                        C
               MAXOUT=1,ANS=STOP,PASSLIM=1,PUTYPE=1,                  C
               PUCB=CXNP1,                                            C
               LUCB=(CXNU1,CXNU1,CXNU1),                              C
               PUFVT=CXNFVTP,                                         C
               LUFVT=(CXNFVTL,CXNFVT1,CXNFVT2)
*              STATOPT=('TTY/A VIRT PU ',NOMONIT)
PYEXAB   LU LOCADDR=0,TERM=WTTY,PACING=1
*              STATOPT=('TTY/A VIRT LU ',NOMONIT)
*********************************************************************
* NTO VIRTUAL LINE 01                                               *
*********************************************************************
PYELV01  LINE NTO.SSLINE=PYEL01,NTO.TRALL=YES,ISTATUS=INACTIVE,        C
               LINECB=CXNL2,                                          C
               LINEFVT=CXNFVTV
*              STATOPT=('TTY/B VIRT LNE',NOMONIT)
*
PYEC01   PU NTO.SSTERM=PYEY01,                                         C
               NTO.EDIT=00,           ATTENTION CHARACTER             C
               NTO.WTTYTO=YES,                                        C
               MAXOUT=1,ANS=STOP,PASSLIM=1,PUTYPE=1,                  C
               PUCB=CXNP2,                                            C
               LUCB=(CXNU2,CXNU2,CXNU2),                              C
               PUFVT=CXNFVTP,                                         C
               LUFVT=(CXNFVTL,CXNFVT1,CXNFVT2)
*              STATOPT=('TTY/B VIRT PU ',NOMONIT)
PYEXAC   LU LOCADDR=0,TERM=WTTY,PACING=1
*              STATOPT=('TTY/B VIRT LU ',NOMONIT)
*********************************************************************
* NTO SELECTIVE FID0 TRACE DEFINITION                               *
*********************************************************************
PYECNTO  GROUP LNCTL=SDLC,MAXLU=1,PUTYPE=1,                            C
               VIRTUAL=YES,                                           C
               VIROWNER=CXNNTO
*
PYELCNTO LINE NTO.SSLINE=TRLINK,ISTATUS=INACTIVE,                      C
               LINECB=CXNL3,                                          C
               LINEFVT=CXNFVTTV
*              STATOPT=('NTO DIAG LINE ',NOMONIT)
*
PYECCNTO PU NTO.SSTERM=TRLINK,                                         C
               PUCB=CXNP3,                                            C
               LUCB=(CXNU3,CXNU3,CXNU3),                              C
               PUFVT=CXNFVTTP,                                        C
               LUFVT=(CXNFVTTP,CXNFVTTP,CXNFVTTP)
*              STATOPT=('NTO DIAG PU   ',NOMONIT)
*
PYECUNTO LU LOCADDR=0
*              STATOPT=('NTO DIAG LU   ',NOMONIT)
*********************************************************************
*    A L C   R E A L   L I N E   G R O U P :  PYEGCALC              *
*********************************************************************
PYEGCALC GROUP COMPTAD=YES,                                            C
               COMPSWP=YES,COMPOWN=YES,COMPACB=YES,                   C
               LEVEL2=DTMLVL2,LEVEL3=DTMLVL3,LEVEL5=NCP,              C
               LNCTL=SDLC,USERID=(ALCI---,DTMBDTNT),                  C
               TIMER=(DTMTIMER,,DTMTIMST,DTMTIMLG),                   C
               XIO=(DTMXIOLN,DTMXIOST,DTMXIOIM,DTMXIOLK),             C
               USE=NCP
*
*********************************************************************
PYEL16   LINE  ADDRESS=(16,FULL),                                      C
               SPEED=9600,                                            C
               UACB=(PYEL16X,PYEL16R),                                C
               MAXPU=4,                                               C
               ETRATIO=1,                                             C
               ISTATUS=ACTIVE,                                        C
               LPDATS=LPDA2
*
PYE16SOT SERVICE ORDER=(PYEC161),                                      C
               MAXLIST=254
```

```
*
*
PYEC161  PU    ADDR=10,                                                C
               ANS=CONT,                                               C
               XID=NO,                                                 C
               PUTYPE=2,                                               C
               PUDR=YES,                                               C
               AVGPB=240,                                              C
               MAXDATA=1200,                                           C
               SRT=(1000,50)
*
************************************************************************
PYEL17   LINE  ADDRESS=(17,FULL),                                      C
               SPEED=9600,                                             C
               UACB=(PYEL17X,PYEL17R),                                 C
               MAXPU=4,                                                C
               ETRATIO=10,                                             C
               ISTATUS=ACTIVE,                                         C
               LPDATS=LPDA2
*
PYE17SOT SERVICE ORDER=(PYEC171),                                      C
               MAXLIST=254
*
*
PYEC171  PU    ADDR=11,                                                C
               ANS=CONT,                                               C
               XID=NO,                                                 C
               PUTYPE=2,                                               C
               PUDR=YES,                                               C
               AVGPB=240,                                              C
               MAXDATA=1200,                                           C
               SRT=(1000,50)
*
************************************************************************
PYEL18   LINE  ADDRESS=(18,FULL),                                      C
               SPEED=9600,                                             C
               UACB=(PYEL18X,PYEL18R),                                 C
               MAXPU=4,                                                C
               ETRATIO=10,                                             C
               ISTATUS=ACTIVE,                                         C
               LPDATS=LPDA2
*
PYE18SOT SERVICE ORDER=(PYEC181),                                      C
               MAXLIST=254
*
*
PYEC181  PU    ADDR=12,                                                C
               ANS=CONT,                                               C
               XID=NO,                                                 C
               PUTYPE=2,                                               C
               PUDR=YES,                                               C
               AVGPB=240,                                              C
               MAXDATA=1200,                                           C
               SRT=(1000,50)
*
************************************************************************
PYEL19   LINE  ADDRESS=(19,FULL),                                      C
               SPEED=9600,                                             C
               UACB=(PYEL19X,PYEL19R),                                 C
               MAXPU=4,                                                C
               ETRATIO=10,                                             C
               ISTATUS=ACTIVE,                                         C
               LPDATS=LPDA2
*
PYE19SOT SERVICE ORDER=(PYEC191,PYEC192),                              C
               MAXLIST=254
*
*
PYEC191  PU    ADDR=13,                                                C
               ANS=CONT,                                               C
               XID=NO,                                                 C
               PUTYPE=2,                                               C
               PUDR=YES,                                               C
               AVGPB=240,                                              C
               MAXDATA=1200,                                           C
               SRT=(1000,50)
*
*
PYEC192  PU    ADDR=14,                                                C
               ANS=CONT,                                               C
               XID=NO,                                                 C
               PUTYPE=2,                                               C
```

```
               PUDR=YES,                                                C
               AVGPB=240,                                               C
               MAXDATA=1200,                                            C
               SRT=(1000,50)
*
*************************************************************************
*    A L C  D U M M Y  L I N E   G R O U P :  PYEGCV                    *
*************************************************************************
PYEGCV   GROUP LEVEL2=DTMLVL2,                                          C
               LEVEL3=DTMLVL3,LEVEL5=NCP,                               C
               LINEADD=NONE,                                            C
               LNCTL=SDLC,USERID=(ALCI---,DTMBDTAP),                    C
               TIMER=(DTMTIMER,,DTMTIMST,DTMTIMLG),                     C
               XIO=(DTMXIOLN,DTMXIOST,DTMXIOIM,DTMXIOLK),               C
               VPACING=0,                                               C
               USE=NCP
*
*************************************************************************
PYELALCI LINE  ADDRESS=(NONE),                                          C
               UACB=(DTMUACBA),                                         C
               ISTATUS=ACTIVE,                                          C
               DLOGMOD=MNEF2,                                           C
               MODETAB=CWTTY
*
PYECVSOT SERVICE ORDER=(PYEIALCI)
*
*
PYEIALCI PU    ADDR=01,                                                 C
               ANS=CONT,                                                C
               XID=NO,                                                  C
               PUTYPE=2,                                                C
               ISTATUS=ACTIVE,                                          C
               MAXDATA=4096
*
PYEUA01  LU    LOCADDR=1,                                               C
               PACING=0,                                                C
               ISTATUS=ACTIVE,                                          C
               LOGAPPL=PYEZ1KT
*
PYEUA02  LU    LOCADDR=2,                                               C
               PACING=0,                                                C
               ISTATUS=ACTIVE,                                          C
               LOGAPPL=PYEZ1ISW
*
*************************************************************************
*            E N D   O F   A L C   L I N E   G R O U P                  *
*************************************************************************
*************************************************************************
* X25NPSI STAGE1 INPUT DECK FOR MVS GENERATION                          *
* NETWORK TYPE 2 DEFINITION FOR CONNECTION TO X.25 EQUIPMENT            *
*************************************************************************
         X25.NET DM=NO,                                                 C
               NETTYPE=1,            DIAG FIELDS NOT USED               C
               CPHINDX=1
*
         X25.VCCPT INDEX=1,                                             C
               MAXPKTL=240,                                             C
               VWINDOW=2
*************************************************************************
* X.25 LINK (9600 BPS) -- 3745 LINE 084                                 *
*************************************************************************
PYEL84   X25.MCH ADDRESS=084,                                           C
               FRMLGTH=243,        FOR MAX PACKET DATA 240              C
               LCN0=NOTUSED,       LCN0 RESERVED FOR NETWORK FUNCTION   C
               LCGDEF=(1,7),                                            C
               MWINDOW=7,          FRAME WINDOW                         C
               LUNAME=PYEX84,                                           C
               PUNAME=PYEC84,                                           C
               NDRETRY=2,          NO. OF TIMES WHOLE SEQUENCE REPEATEDC
               NPRETRY=5,          NO. OF TIMES PACKETS RETRIED         C
               TPTIMER=2,          T1 TIMER ELAPSED BETWEEN PKT RETRIESC
               STATION=DCE,        HOST IS NETWORK                      C
               OWNER=PYE,                                               C
               ISTATUS=ACTIVE,                                          C
               ANS=CONTINUE
*              STATOPT=(' X.25 ',NOMONIT)
*************************************************************************
* LOGICAL CHANNEL GROUP 1 -- X.25 REMOTE LINE 1 (8 IA)                  *
*************************************************************************
         X25.LCG LCGN=1
*
PYEL841  X25.LINE LCN=0,                                                C
```

```
               TYPE=PERMANENT,                                          C
               LLC=LLC0,                                                C
               OWNER=PYE,                                               C
               ISTATUS=ACTIVE,                                          C
               VCCINDX=1
*              STATOPT=('X25 LCN0      ',NOMONIT)
*
PYEC841  X25.PU PUTYPE=1,                                               C
               ADDR=01,                                                 C
               VPACING=(2,1),                                           C
               PACING=1,                                                C
               MAXDATA=3845,                                            C
               MODETAB=LOGIASC,                                         C
               DLOGMOD=X25LU1
*              STATOPT=('X25 LCN0 PU=1 ',NOMONIT)
*
PYEU841  X25.LU LOCADDR=0,                                              C
               USSTAB=USS1024B
*              STATOPT=('X25 LCN0 LU=0 ',NOMONIT)
*
PYEL842  X25.LINE LCN=1,                                                C
               TYPE=PERMANENT,                                          C
               LLC=LLC0,                                                C
               OWNER=PYE,                                               C
               ISTATUS=ACTIVE,                                          C
               VCCINDX=1
*              STATOPT=('X25 LCN1      ',NOMONIT)
*
PYEC842  X25.PU PUTYPE=1,                                               C
               ADDR=01,                                                 C
               VPACING=(2,1),                                           C
               PACING=1,                                                C
               MAXDATA=3845,                                            C
               MODETAB=LOGIASC,                                         C
               DLOGMOD=X25LU1
*              STATOPT=('X25 LCN1 PU=1 ',NOMONIT)
*
PYEU842  X25.LU LOCADDR=0,                                              C
               USSTAB=USS1024B
*              STATOPT=('X25 LCN1 LU=0 ',NOMONIT)
*
PYEL843  X25.LINE LCN=2,                                                C
               TYPE=PERMANENT,                                          C
               LLC=LLC0,                                                C
               OWNER=PYE,                                               C
               ISTATUS=ACTIVE,                                          C
               VCCINDX=1
*              STATOPT=('X25 LCN2      ',NOMONIT)
*
PYEC843  X25.PU PUTYPE=1,                                               C
               ADDR=01,                                                 C
               VPACING=(2,1),                                           C
               PACING=1,                                                C
               MAXDATA=3845,                                            C
               MODETAB=LOGIASC,                                         C
               DLOGMOD=X25LU1
*              STATOPT=('X25 LCN2 PU=1 ',NOMONIT)
*
PYEU843  X25.LU LOCADDR=0,                                              C
               USSTAB=USS1024B
*              STATOPT=('X25 LCN2 LU=0 ',NOMONIT)
*
PYEL844  X25.LINE LCN=3,                                                C
               TYPE=PERMANENT,                                          C
               LLC=LLC0,                                                C
               OWNER=PYE,                                               C
               ISTATUS=ACTIVE,                                          C
               VCCINDX=1
*              STATOPT=('X25 LCN3      ',NOMONIT)
*
PYEC844  X25.PU PUTYPE=1,                                               C
               ADDR=01,                                                 C
               VPACING=(2,1),                                           C
               PACING=1,                                                C
               MAXDATA=3845,                                            C
               MODETAB=LOGIASC,                                         C
               DLOGMOD=X25LU1
*              STATOPT=('X25 LCN3 PU=1 ',NOMONIT)
*
PYEU844  X25.LU LOCADDR=0,                                              C
               USSTAB=USS1024B
*              STATOPT=('X25 LCN3 LU=0 ',NOMONIT)
```

```
*
PYEL845  X25.LINE LCN=4,                                              C
               TYPE=PERMANENT,                                        C
               LLC=LLC0,                                              C
               OWNER=PYE,                                             C
               ISTATUS=ACTIVE,                                        C
               VCCINDX=1
*              STATOPT=('X25 LCN4      ',NOMONIT)
*
PYEC845  X25.PU PUTYPE=1,                                             C
               ADDR=01,                                               C
               VPACING=(2,1),                                         C
               PACING=1,                                              C
               MAXDATA=3845,                                          C
               MODETAB=LOGIASC,                                       C
               DLOGMOD=X25LU1
*              STATOPT=('X25 LCN4 PU=1 ',NOMONIT)
*
PYEU845  X25.LU LOCADDR=0,                                            C
               USSTAB=USS1024B
*              STATOPT=('X25 LCN4 LU=0 ',NOMONIT)
*
PYEL846  X25.LINE LCN=5,                                              C
               TYPE=PERMANENT,                                        C
               LLC=LLC0,                                              C
               OWNER=PYE,                                             C
               ISTATUS=ACTIVE,                                        C
               VCCINDX=1
*              STATOPT=('X25 LCN5      ',NOMONIT)
*
PYEC846  X25.PU PUTYPE=1,                                             C
               ADDR=01,                                               C
               VPACING=(2,1),                                         C
               PACING=1,                                              C
               MAXDATA=3845,                                          C
               MODETAB=LOGIASC,                                       C
               DLOGMOD=X25LU1
*              STATOPT=('X25 LCN5 PU=1 ',NOMONIT)
*
PYEU846  X25.LU LOCADDR=0,                                            C
               USSTAB=USS1024B
*              STATOPT=('X25 LCN5 LU=0 ',NOMONIT)
*
PYEL847  X25.LINE LCN=6,                                              C
               TYPE=PERMANENT,                                        C
               LLC=LLC0,                                              C
               OWNER=PYE,                                             C
               ISTATUS=ACTIVE,                                        C
               VCCINDX=1
*              STATOPT=('X25 LCN6      ',NOMONIT)
*
PYEC847  X25.PU PUTYPE=1,                                             C
               ADDR=01,                                               C
               VPACING=(2,1),                                         C
               PACING=1,                                              C
               MAXDATA=3845,                                          C
               MODETAB=LOGIASC,                                       C
               DLOGMOD=X25LU1
*              STATOPT=('X25 LCN6 PU=1 ',NOMONIT)
*
PYEU847  X25.LU LOCADDR=0,                                            C
               USSTAB=USS1024B
*              STATOPT=('X25 LCN6 LU=0 ',NOMONIT)
*
PYEL848  X25.LINE LCN=7,                                              C
               TYPE=PERMANENT,                                        C
               LLC=LLC0,                                              C
               OWNER=PYE,                                             C
               ISTATUS=ACTIVE,                                        C
               VCCINDX=1
*              STATOPT=('X25 LCN7      ',NOMONIT)
*
PYEC848  X25.PU PUTYPE=1,                                             C
               ADDR=01,                                               C
               VPACING=(2,1),                                         C
               PACING=1,                                              C
               MAXDATA=3845,                                          C
               MODETAB=LOGIASC,                                       C
               DLOGMOD=X25LU1
*              STATOPT=('X25 LCN7 PU=1 ',NOMONIT)
*
PYEU848  X25.LU LOCADDR=0,                                            C
```

```
              USSTAB=USS1024B
*             STATOPT=('X25 LCN7 LU=0',NOMONIT)
         X25.END
*********************************************************************
* NPA DEFINITIONS                                                   *
*********************************************************************
PYEGCNPA GROUP LNCTL=SDLC,NPARSC=YES,VIRTUAL=YES
PYELCNPA LINE
PYECCNPA PU
PYEUCNPA LU MAXCOLL=100
*********************************************************************
* GROUP FOR SDLC LINES                                              *
*********************************************************************
PYEGPU2  GROUP LNCTL=SDLC,                                           C
               CLOCKNG=EXT,                CLOCKING BY MODEM         C
               DATRATE=HIGH,                                         C
               DIAL=NO,                    NON-SWITCHED              C
               MODETAB=DY1D37,                                       C
               NPACOLL=YES,                                          C
               NRZI=YES,                   DEFAULT                   C
               REPLYTO=1.0,                REPLY TIMEOUT VALUE       C
               TEXTTO=3.0,                                           C
               TYPE=NCP,                                             C
               USSTAB=USSIASC,                                       C
               VIRTUAL=NO
*              ANS=CONT,
*              RETRIES=(5,10,3),           3 * (5 RETRIES @ 10 SECS)
*********************************************************************
* SDLC LINE                                                         *
*********************************************************************
PYEL80   LINE ADDRESS=(080,FULL),SPEED=9600,                         C
               AVGPB=84,                                             C
               DUPLEX=FULL,                                          C
               ETRATIO=30,NPACOLL=YES,                               C
               ISTATUS=ACTIVE,                                       C
               LPDATS=LPDA2,                                         C
               NEWSYNC=NO,                                           C
               NRZI=NO,                    DEFAULT                   C
               PAUSE=(0.2,2.8),                                      C
               RETRIES=(5,1,4),TRANSFR=74,                           C
               SERVLIM=4,                                            C
               SPDSEL=NO
*              STATOPT=('A2CS 1ST FLOOR',NOMONIT)
         SERVICE ORDER=(PYEC80)
*
*********************************************************************
*********************************************************************
* 3274 CONTROL UNIT                                                 *
*********************************************************************
PYEC80   PU ADDR=C1,                                                 C
               ISTATUS=ACTIVE,                                       C
               PUTYPE=2,                                             C
               PASSLIM=1,                  MAX CONSECUTIVE PIU'S     C
               MAXDATA=265,                MAX DATA BYTES IN 1 PIU   C
               NPACOLL=YES,                                          C
               PACING=1,                                             C
               PUDR=NO,                                              C
               SSCPFM=USSSCS
*              STATOPT=('A2CS 1ST FLOOR',NOMONIT)
*
PYEC8000 LU LOCADDR=2,DLOGMOD=M32782
PYEC8001 LU LOCADDR=3,DLOGMOD=M3287
PYEC8002 LU LOCADDR=4,DLOGMOD=M32782
PYEC8003 LU LOCADDR=5,DLOGMOD=M32782
PYEC8004 LU LOCADDR=6,DLOGMOD=M32872
PYEC8005 LU LOCADDR=7,DLOGMOD=M32782
PYEC8006 LU LOCADDR=8,DLOGMOD=M32872
PYEC8007 LU LOCADDR=9,DLOGMOD=M32872
PYEC8008 LU LOCADDR=10,DLOGMOD=M32872
PYEC8009 LU LOCADDR=11,DLOGMOD=M32782
PYEC800A LU LOCADDR=12,MODETAB=DY1DALCI,DLOGMOD=MNEF2,LOGAPPL=PYEZ1KBT
PYEC800B LU LOCADDR=13,MODETAB=CWTTY,DLOGMOD=MNEF2
PYEC800C LU LOCADDR=14,DLOGMOD=M32872
PYEC800D LU LOCADDR=15,DLOGMOD=M32782
PYEC800E LU LOCADDR=16,DLOGMOD=M32782
PYEC800F LU LOCADDR=17,DLOGMOD=M32782
*
PYEC8010 LU LOCADDR=18,DLOGMOD=M32782
PYEC8011 LU LOCADDR=19,DLOGMOD=M32782
PYEC8012 LU LOCADDR=20,DLOGMOD=M32782
PYEC8013 LU LOCADDR=21,DLOGMOD=M32782
PYEC8014 LU LOCADDR=22,DLOGMOD=M32782
```

```
PYEC8015 LU LOCADDR=23,DLOGMOD=M32782
PYEC8016 LU LOCADDR=24,DLOGMOD=M32782
PYEC8017 LU LOCADDR=25,DLOGMOD=M32782
PYEC8018 LU LOCADDR=26,DLOGMOD=M32782
PYEC8019 LU LOCADDR=27,DLOGMOD=M32782
PYEC801A LU LOCADDR=28,DLOGMOD=M32782
PYEC801B LU LOCADDR=29,DLOGMOD=M32782
PYEC801C LU LOCADDR=30,DLOGMOD=M32782
PYEC801D LU LOCADDR=31,DLOGMOD=M32782
PYEC801E LU LOCADDR=32,DLOGMOD=M32782
PYEC801F LU LOCADDR=33,DLOGMOD=M32782
*
PYEC8020 LU LOCADDR=34,DLOGMOD=M32782
*              STATOPT=('ASCII 3153',NOMONIT)
PYEC8021 LU LOCADDR=35,DLOGMOD=M32782
*              STATOPT=('ASCII 3153',NOMONIT)
PYEC8022 LU LOCADDR=36,DLOGMOD=M32782
*              STATOPT=('ASCII 3153',NOMONIT)
PYEC8023 LU LOCADDR=37,DLOGMOD=M32782
*              STATOPT=('ASCII 3153',NOMONIT)
PYEC8024 LU LOCADDR=38,DLOGMOD=M32872
*              STATOPT=('IER PRINTER',NOMONIT)
PYEC8025 LU LOCADDR=39,DLOGMOD=M32872
*              STATOPT=('IER PRINTER',NOMONIT)
PYEP8026 LU LOCADDR=40,DLOGMOD=M32872
*              STATOPT=('IER PRINTER',NOMONIT)
PYES8027 LU LOCADDR=41,DLOGMOD=M32872
*              STATOPT=('IER PRINTER',NOMONIT)
PYES8028 LU LOCADDR=42,DLOGMOD=M32782
PYES8029 LU LOCADDR=43,DLOGMOD=M32782
PYES802A LU LOCADDR=44,DLOGMOD=M32782
PYES802B LU LOCADDR=45,DLOGMOD=M32782
PYES802C LU LOCADDR=46,DLOGMOD=M32782
PYES802D LU LOCADDR=47,DLOGMOD=M32782
PYES802E LU LOCADDR=48,DLOGMOD=M32782
PYES802F LU LOCADDR=49,DLOGMOD=M32782
*
PYEP8030 LU LOCADDR=50,DLOGMOD=M32872
PYES8031 LU LOCADDR=51,DLOGMOD=M32782
PYES8032 LU LOCADDR=52,DLOGMOD=M32782
PYES8033 LU LOCADDR=53,DLOGMOD=M32782
PYES8034 LU LOCADDR=54,DLOGMOD=M32782
PYES8035 LU LOCADDR=55,DLOGMOD=M32782
PYES8036 LU LOCADDR=56,DLOGMOD=M32782
PYES8037 LU LOCADDR=57,DLOGMOD=M32782
PYES8038 LU LOCADDR=58,DLOGMOD=M32782
PYES8039 LU LOCADDR=59,DLOGMOD=M32782
PYEP803A LU LOCADDR=60,DLOGMOD=M32872
PYES803B LU LOCADDR=61,DLOGMOD=M32782
PYES803C LU LOCADDR=62,DLOGMOD=M32782
PYES803D LU LOCADDR=63,DLOGMOD=M32782
PYES803E LU LOCADDR=64,DLOGMOD=M32782
PYES803F LU LOCADDR=65,DLOGMOD=M32782
*
PYES8040 LU LOCADDR=66,DLOGMOD=M32782
PYES8041 LU LOCADDR=67,DLOGMOD=M32782
PYES8042 LU LOCADDR=68,DLOGMOD=M32782
PYES8043 LU LOCADDR=69,DLOGMOD=M32782
PYEP8044 LU LOCADDR=70,DLOGMOD=M32872
PYES8045 LU LOCADDR=71,DLOGMOD=M32782
PYES8046 LU LOCADDR=72,DLOGMOD=M32782
PYES8047 LU LOCADDR=73,DLOGMOD=M32782
PYES8048 LU LOCADDR=74,DLOGMOD=M32782
PYES8049 LU LOCADDR=75,DLOGMOD=M32782
PYES804A LU LOCADDR=76,DLOGMOD=M32782
PYES804B LU LOCADDR=77,DLOGMOD=M32782
PYES804C LU LOCADDR=78,DLOGMOD=M32782
PYES804D LU LOCADDR=79,DLOGMOD=M32782
PYEP804E LU LOCADDR=80,DLOGMOD=M32872
PYES804F LU LOCADDR=81,DLOGMOD=M32782
*
PYES8050 LU LOCADDR=82,DLOGMOD=M32782
PYES8051 LU LOCADDR=83,DLOGMOD=M32782
PYES8052 LU LOCADDR=84,DLOGMOD=M32782
PYES8053 LU LOCADDR=85,DLOGMOD=M32782
PYES8054 LU LOCADDR=86,DLOGMOD=M32782
PYES8055 LU LOCADDR=87,DLOGMOD=M32782
PYES8056 LU LOCADDR=88,DLOGMOD=M32782
PYES8057 LU LOCADDR=89,DLOGMOD=M32782
PYEP8058 LU LOCADDR=90,DLOGMOD=M32872
PYES8059 LU LOCADDR=91,DLOGMOD=M32782
PYES805A LU LOCADDR=92,DLOGMOD=M32782
```

```
PYES805B LU LOCADDR=93,DLOGMOD=M32782
PYES805C LU LOCADDR=94,DLOGMOD=M32782
PYES805D LU LOCADDR=95,DLOGMOD=M32782
PYES805E LU LOCADDR=96,DLOGMOD=M32782
PYES805F LU LOCADDR=97,DLOGMOD=M32782
*
PYES8060 LU LOCADDR=98,DLOGMOD=M32782
PYES8061 LU LOCADDR=99,DLOGMOD=M32782
PYEP8062 LU LOCADDR=100,DLOGMOD=M32872
PYES8063 LU LOCADDR=101,DLOGMOD=M32782
PYES8064 LU LOCADDR=102,DLOGMOD=M32782
PYES8065 LU LOCADDR=103,DLOGMOD=M32782
PYES8066 LU LOCADDR=104,DLOGMOD=M32782
PYES8067 LU LOCADDR=105,DLOGMOD=M32782
PYES8068 LU LOCADDR=106,DLOGMOD=M32782
PYES8069 LU LOCADDR=107,DLOGMOD=M32782
PYES806A LU LOCADDR=108,DLOGMOD=M32782
PYES806B LU LOCADDR=109,DLOGMOD=M32782
PYEP806C LU LOCADDR=110,DLOGMOD=M32872
PYES806D LU LOCADDR=111,DLOGMOD=M32782
PYES806E LU LOCADDR=112,DLOGMOD=M32782
PYES806F LU LOCADDR=113,DLOGMOD=M32782
*
PYES8070 LU LOCADDR=114,DLOGMOD=M32782
PYES8071 LU LOCADDR=115,DLOGMOD=M32782
PYES8072 LU LOCADDR=116,DLOGMOD=M32782
PYES8073 LU LOCADDR=117,DLOGMOD=M32782
PYES8074 LU LOCADDR=118,DLOGMOD=M32782
PYES8075 LU LOCADDR=119,DLOGMOD=M32782
PYEP8076 LU LOCADDR=120,DLOGMOD=M32872
PYES8077 LU LOCADDR=121,DLOGMOD=M32782
PYES8078 LU LOCADDR=122,DLOGMOD=M32782
PYES8079 LU LOCADDR=123,DLOGMOD=M32782
PYES807A LU LOCADDR=124,DLOGMOD=M32782
PYES807B LU LOCADDR=125,DLOGMOD=M32782
PYES807C LU LOCADDR=126,DLOGMOD=M32782
PYES807D LU LOCADDR=127,DLOGMOD=M32782
PYES807E LU LOCADDR=128,DLOGMOD=M32782
PYES807F LU LOCADDR=129,DLOGMOD=M32782
*********************************************************************
* SDLC LINE                                                         *
*********************************************************************
PYEL81   LINE ADDRESS=(081,FULL),SPEED=9600,                         C
               AVGPB=84,                                             C
               DUPLEX=FULL,                                          C
               LPDATS=LPDA2,                                         C
               ETRATIO=30,NPACOLL=YES,                               C
               ISTATUS=ACTIVE,                                       C
               NEWSYNC=NO,                                           C
               PAUSE=(0.2,2.8),                                      C
               RETRIES=(5,1,4),TRANSFR=74,                           C
               SERVLIM=4,                                            C
               SPDSEL=NO
*              STATOPT=('3274          ',NOMONIT)
*
         SERVICE ORDER=(PYEC81)
*
*********************************************************************
* 3274 CONTROL UNIT                                                 *
*********************************************************************
PYEC81   PU ADDR=C4,                                                 C
               ISTATUS=ACTIVE,                                       C
               PUTYPE=2,                                             C
               PASSLIM=1,                  MAX CONSECUTIVE PIU'S     C
               MAXDATA=512,                MAX DATA BYTES IN 1 PIU   C
               NPACOLL=YES,                                          C
               PACING=1,                                             C
               PUDR=NO,                                              C
               SSCPFM=USSSCS
*              STATOPT=('3274 PU=C4    ',NOMONIT)
*
PYES0702 LU    LOCADDR=2,DLOGMOD=M32782
PYES0703 LU    LOCADDR=3,DLOGMOD=M32782
PYES0704 LU    LOCADDR=4,DLOGMOD=M32782
PYES0705 LU    LOCADDR=5,DLOGMOD=M32782
PYES0706 LU    LOCADDR=6,DLOGMOD=M32782
PYES0707 LU    LOCADDR=7,DLOGMOD=M32782
PYES0708 LU    LOCADDR=8,DLOGMOD=M32782
PYEP0709 LU    LOCADDR=9,DLOGMOD=M32872
*
PYES0710 LU    LOCADDR=10,DLOGMOD=M32782
PYES0711 LU    LOCADDR=11,DLOGMOD=M32785
```

```
PYES0712 LU    LOCADDR=12,DLOGMOD=M32872
PYES0713 LU    LOCADDR=13,DLOGMOD=M32782
PYES0714 LU    LOCADDR=14,DLOGMOD=M32782
PYES0715 LU    LOCADDR=15,DLOGMOD=M32782
PYES0716 LU    LOCADDR=16,DLOGMOD=M32782
PYES0717 LU    LOCADDR=17,DLOGMOD=M32793
PYES0718 LU    LOCADDR=18,DLOGMOD=M32782
PYES0719 LU    LOCADDR=19,DLOGMOD=M32793
*
PYES0720 LU    LOCADDR=20,DLOGMOD=M32782
PYES0721 LU    LOCADDR=21,DLOGMOD=M32782
PYES0722 LU    LOCADDR=22,DLOGMOD=M32782
PYES0723 LU    LOCADDR=23,DLOGMOD=M32782
PYES0724 LU    LOCADDR=24,DLOGMOD=M32782
PYES0725 LU    LOCADDR=25,DLOGMOD=M32782
PYES0726 LU    LOCADDR=26,DLOGMOD=M32782
PYES0727 LU    LOCADDR=27,DLOGMOD=M32782
PYES0728 LU    LOCADDR=28,DLOGMOD=M32782
PYES0729 LU    LOCADDR=29,DLOGMOD=M32782
*
PYES0730 LU    LOCADDR=30,DLOGMOD=M32782
PYES0731 LU    LOCADDR=31,DLOGMOD=M32782
PYES0732 LU    LOCADDR=32,DLOGMOD=M32782
PYES0733 LU    LOCADDR=33,DLOGMOD=M32782
PYEP0734 LU    LOCADDR=34,DLOGMOD=M32872
PYES0735 LU    LOCADDR=35,DLOGMOD=M32782
PYES0736 LU    LOCADDR=36,DLOGMOD=M32782
PYEP0737 LU    LOCADDR=37,DLOGMOD=M32872
PYES0738 LU    LOCADDR=38,DLOGMOD=M32782
PYES0739 LU    LOCADDR=39,DLOGMOD=M32782
*
PYEP0740 LU    LOCADDR=40,DLOGMOD=M32872
PYES0741 LU    LOCADDR=41,DLOGMOD=M32782
PYES0742 LU    LOCADDR=42,DLOGMOD=M32782
PYEP0743 LU    LOCADDR=43,DLOGMOD=M32872
PYES0744 LU    LOCADDR=44,DLOGMOD=M32782
PYES0745 LU    LOCADDR=45,DLOGMOD=M32782
PYEP0746 LU    LOCADDR=46,DLOGMOD=M32872
PYES0747 LU    LOCADDR=47,DLOGMOD=M32782
PYES0748 LU    LOCADDR=48,DLOGMOD=M32782
PYEP0749 LU    LOCADDR=49,DLOGMOD=M32872
*
PYES0750 LU    LOCADDR=50,DLOGMOD=M32782
PYES0751 LU    LOCADDR=51,DLOGMOD=M32782
PYES0752 LU    LOCADDR=52,DLOGMOD=M32782
PYES0753 LU    LOCADDR=53,DLOGMOD=M32782
PYES0754 LU    LOCADDR=54,DLOGMOD=M32782
PYES0755 LU    LOCADDR=55,DLOGMOD=M32872
PYES0756 LU    LOCADDR=56,DLOGMOD=M32782
PYES0757 LU    LOCADDR=57,DLOGMOD=M32782
PYES0758 LU    LOCADDR=58,DLOGMOD=M32782
PYES0759 LU    LOCADDR=59,DLOGMOD=M32782
*
PYES0760 LU    LOCADDR=60,DLOGMOD=M32782
PYES0761 LU    LOCADDR=61,DLOGMOD=M32782
PYES0762 LU    LOCADDR=62,DLOGMOD=M32782
PYES0763 LU    LOCADDR=63,DLOGMOD=M32782
PYES0764 LU    LOCADDR=64,DLOGMOD=M32782
PYES0765 LU    LOCADDR=65,DLOGMOD=M32782
PYES0766 LU    LOCADDR=66,DLOGMOD=M32782
PYES0767 LU    LOCADDR=67,DLOGMOD=M32782
PYES0768 LU    LOCADDR=68,DLOGMOD=M32782
PYES0769 LU    LOCADDR=69,DLOGMOD=M32782
*
PYES0770 LU    LOCADDR=70,DLOGMOD=M32782
PYES0771 LU    LOCADDR=71,DLOGMOD=M32782
PYES0772 LU    LOCADDR=72,DLOGMOD=M32782
PYES0773 LU    LOCADDR=73,DLOGMOD=M32782
PYES0774 LU    LOCADDR=74,DLOGMOD=M32782
PYES0775 LU    LOCADDR=75,DLOGMOD=M32782
PYES0776 LU    LOCADDR=76,DLOGMOD=M32782
PYES0777 LU    LOCADDR=77,DLOGMOD=M32782
PYES0778 LU    LOCADDR=78,DLOGMOD=M32782
PYES0779 LU    LOCADDR=79,DLOGMOD=M32782
*
PYES0780 LU    LOCADDR=80,DLOGMOD=M32782
PYES0781 LU    LOCADDR=81,DLOGMOD=M32782
PYES0782 LU    LOCADDR=82,DLOGMOD=M32782
PYES0783 LU    LOCADDR=83,DLOGMOD=M32782
PYES0784 LU    LOCADDR=84,DLOGMOD=M32782
PYES0785 LU    LOCADDR=85,DLOGMOD=M32782
PYES0786 LU    LOCADDR=86,DLOGMOD=M32782
```

```
PYES0787 LU    LOCADDR=87,DLOGMOD=M32782
PYES0788 LU    LOCADDR=88,DLOGMOD=M32782
PYES0789 LU    LOCADDR=89,DLOGMOD=M32782
*
PYES0790 LU    LOCADDR=90,DLOGMOD=M32782
PYES0791 LU    LOCADDR=91,DLOGMOD=M32782
PYES0792 LU    LOCADDR=92,DLOGMOD=M32782
PYES0793 LU    LOCADDR=93,DLOGMOD=M32782
PYES0794 LU    LOCADDR=94,DLOGMOD=M32782
PYES0795 LU    LOCADDR=95,DLOGMOD=M32782
PYES0796 LU    LOCADDR=96,DLOGMOD=M32782
PYES0797 LU    LOCADDR=97,DLOGMOD=M32782
PYES0798 LU    LOCADDR=98,DLOGMOD=M32782
PYES0799 LU    LOCADDR=99,DLOGMOD=M32782
*
PYES07A0 LU    LOCADDR=100,DLOGMOD=M32782
PYES07A1 LU    LOCADDR=101,DLOGMOD=M32782
PYES07A2 LU    LOCADDR=102,DLOGMOD=M32782
PYES07A3 LU    LOCADDR=103,DLOGMOD=M32872
PYES07A4 LU    LOCADDR=104,DLOGMOD=M32782
PYES07A5 LU    LOCADDR=105,DLOGMOD=M32782
PYES07A6 LU    LOCADDR=106,DLOGMOD=M32782
PYES07A7 LU    LOCADDR=107,DLOGMOD=M32782
PYES07A8 LU    LOCADDR=108,DLOGMOD=M32782
PYES07A9 LU    LOCADDR=109,DLOGMOD=M32872
*
PYES07B0 LU    LOCADDR=110,DLOGMOD=M32782
PYES07B1 LU    LOCADDR=111,DLOGMOD=M32782
PYES07B2 LU    LOCADDR=112,DLOGMOD=M32782
PYES07B3 LU    LOCADDR=113,DLOGMOD=M32782
PYES07B4 LU    LOCADDR=114,DLOGMOD=M32782
PYES07B5 LU    LOCADDR=115,DLOGMOD=M32782
PYES07B6 LU    LOCADDR=116,DLOGMOD=M32782
PYES07B7 LU    LOCADDR=117,DLOGMOD=M32782
PYES07B8 LU    LOCADDR=118,DLOGMOD=M32782
PYES07B9 LU    LOCADDR=119,DLOGMOD=M32782
*
PYES07C0 LU    LOCADDR=120,DLOGMOD=M32782
PYES07C1 LU    LOCADDR=121,DLOGMOD=M32782
PYES07C2 LU    LOCADDR=122,DLOGMOD=M32782
PYES07C3 LU    LOCADDR=123,DLOGMOD=M32782
PYES07C4 LU    LOCADDR=124,DLOGMOD=M32782
PYES07C5 LU    LOCADDR=125,DLOGMOD=M32782
PYES07C6 LU    LOCADDR=126,DLOGMOD=M32782
PYES07C7 LU    LOCADDR=127,DLOGMOD=M32782
PYES07C8 LU    LOCADDR=128,DLOGMOD=M32782
PYES07C9 LU    LOCADDR=129,DLOGMOD=M32782
**********************************************************************
* DEMO DEFINITION FOR RS/6000                                        *
**********************************************************************
PYEL82   LINE ADDRESS=(082,FULL),SPEED=9600,                          C
              AVGPB=84,                                               C
              DUPLEX=FULL,                                            C
              LPDATS=LPDA2,                                           C
              ETRATIO=30,NPACOLL=YES,                                 C
              ISTATUS=ACTIVE,                                         C
              NEWSYNC=NO,                                             C
              PAUSE=(0.2,2.8),                                        C
              RETRIES=(5,1,4),TRANSFR=74,                             C
              SERVLIM=4,                                              C
              SPDSEL=NO
*             STATOPT=('RS/6000       ',NOMONIT)
*
         SERVICE ORDER=(PYEC82)
*
PYEC82   PU ADDR=C1,                                                  C
              ISTATUS=INACTIVE,                                       C
              PUTYPE=2,                                               C
              PASSLIM=1,                 MAX CONSECUTIVE PIU'S        C
              MAXDATA=265,               MAX DATA BYTES IN 1 PIU      C
              MODETAB=IASC62,                                         C
              NPACOLL=YES,                                            C
              PACING=1,                                               C
              PUDR=NO,                                                C
              SSCPFM=USSSCS
*             STATOPT=('RS/6000       ',NOMONIT)
PYEC8200 LU LOCADDR=0,DLOGMOD=CICSOS2
*             STATOPT=('LU6.2',NOMONIT)
**********************************************************************
PYEL83   LINE ADDRESS=(083,FULL),SPEED=9600,                          C
              AVGPB=84,                                               C
              DUPLEX=FULL,                                            C
```

```
               LPDATS=LPDA2,                                           C
               ETRATIO=30,NPACOLL=YES,                                 C
               ISTATUS=ACTIVE,                                         C
               NEWSYNC=NO,                                             C
               PAUSE=(0.2,2.8),                                        C
               RETRIES=(5,1,4),TRANSFR=74,                             C
               SERVLIM=4,                                              C
               SPDSEL=NO
*              STATOPT=('RS/6000       ',NOMONIT)
*
         SERVICE ORDER=(PYEC83)
PYEC83   PU ADDR=C1,                                                   C
               ISTATUS=INACTIVE,                                       C
               PUTYPE=2,                                               C
               PASSLIM=1,                  MAX CONSECUTIVE PIU'S       C
               MAXDATA=265,                MAX DATA BYTES IN 1 PIU     C
               MODETAB=IASC62,                                         C
               NPACOLL=YES,                                            C
               PACING=1,                                               C
               PUDR=NO,                                                C
               SSCPFM=USSSCS
*              STATOPT=('RS/6000       ',NOMONIT)
PYEC8300 LU LOCADDR=1,DLOGMOD=CLU62
*              STATOPT=('LU6.2',NOMONIT)
*****************************************************************
PYEL86   LINE ADDRESS=(086,FULL),SPEED=9600,                           C
               AVGPB=84,                                               C
               DUPLEX=FULL,                                            C
               LPDATS=LPDA2,                                           C
               ETRATIO=30,NPACOLL=YES,                                 C
               ISTATUS=ACTIVE,                                         C
               NEWSYNC=NO,                                             C
               PAUSE=(0.2,2.8),                                        C
               RETRIES=(5,1,4),TRANSFR=74,                             C
               SERVLIM=4,                                              C
               SPDSEL=NO
*              STATOPT=('RS/6000       ',NOMONIT)
*
         SERVICE ORDER=(PYEC86)
PYEC86   PU ADDR=C1,                                                   C
               ISTATUS=INACTIVE,                                       C
               PUTYPE=2,                                               C
               PASSLIM=1,                  MAX CONSECUTIVE PIU'S       C
               MAXDATA=265,                MAX DATA BYTES IN 1 PIU     C
               NPACOLL=YES,                                            C
               PACING=1,                                               C
               PUDR=NO,                                                C
               SSCPFM=USSSCS
*              STATOPT=('RS/6000       ',NOMONIT)
PYEC8600 LU LOCADDR=1,DLOGMOD=SNASVCMG
*              STATOPT=('LU6.2',NOMONIT)
*****************************************************************
PYEL87   LINE ADDRESS=(087,FULL),SPEED=9600,                           C
               AVGPB=84,                                               C
               DUPLEX=FULL,                                            C
               LPDATS=LPDA2,                                           C
               ETRATIO=30,NPACOLL=YES,                                 C
               ISTATUS=ACTIVE,                                         C
               NEWSYNC=NO,                                             C
               PAUSE=(0.2,2.8),                                        C
               RETRIES=(5,1,4),TRANSFR=74,                             C
               SERVLIM=4,                                              C
               SPDSEL=NO
*              STATOPT=('RS/6000       ',NOMONIT)
*
         SERVICE ORDER=(PYEC87)
PYEC87   PU ADDR=C1,                                                   C
               ISTATUS=INACTIVE,                                       C
               PUTYPE=2,                                               C
               PASSLIM=1,                  MAX CONSECUTIVE PIU'S       C
               MAXDATA=265,                MAX DATA BYTES IN 1 PIU     C
               NPACOLL=YES,                                            C
               PACING=1,                                               C
               PUDR=NO,                                                C
               SSCPFM=USSSCS
*              STATOPT=('RS/6000       ',NOMONIT)
PYEC8700 LU LOCADDR=1,DLOGMOD=SNASVCMG
*              STATOPT=('LU6.2',NOMONIT)
         EJECT
*********************************************************************
* INN LINKS TO 37XX SUBAREA 8                                       *
*********************************************************************
```

```
PYEGPRI  GROUP TYPE=NCP,LNCTL=SDLC,MODE=PRIMARY,                        C
               NPACOLL=YES,REPLYTO=6,DIAL=NO
PYEGSEC  GROUP TYPE=NCP,LNCTL=SDLC,MODE=SECONDARY,                      C
               TEXTTO=1,NPACOLL=YES,REPLYTO=6,VIRTUAL=NO,DIAL=NO
PYEGINN  GROUP LNCTL=SDLC,ANS=CONTINUE,CLOCKNG=EXT,COMPTAD=YES,IPL=YES
PYEL08   LINE ADDRESS=(008,FULL),SPEED=64000,HISPEED=NO,                C
               MODULO=128,CONFIG=NONSW,IPL=YES,MONLINK=YES,MAXPU=1,     C
               NRZI=YES,SDLCST=(SDLC0,SDLC1),LPDATS=NO,ETRATIO=30,      C
               PAUSE=(0.2,5.0),SERVLIM=254, RETRIES=(3,10,5),           C
               TRANSFR=49,SPDSEL=NO,ISTATUS=ACTIVE,NEWSYNC=NO,          C
               CLOCKNG=EXT,DUPLEX=FULL,AVGPB=2046,DATRATE=HIGH,         C
               LIC=3
*              STATOPT=('INN TO PYEAL32',NOMONIT)
*              STATOPT=('SA29 TO SA8')
PYEC08   PU PUTYPE=4,DATMODE=FULL,MAXOUT=127,IRETRY=NO,PASSLIM=127,     C
               MODULO=128,                                              C
               TGN=(1),RETRIES=(,10,5),SRT=(32768,50),ISTATUS=ACTIVE
*              STATOPT=('STN TO PYEAP32',NOMONIT)
*************************************************************************
* INN LINKS TO 37XX SUBAREA 30                                         *
*************************************************************************
PYEL0C   LINE ADDRESS=(012,FULL),SPEED=64000,HISPEED=NO,                C
               MODULO=128,CONFIG=NONSW,IPL=YES,MONLINK=YES,MAXPU=1,     C
               NRZI=YES,SDLCST=(SDLC0,SDLC1),LPDATS=NO,ETRATIO=30,      C
               PAUSE=(0.2,5.0),SERVLIM=254, RETRIES=(3,10,5),           C
               TRANSFR=49,SPDSEL=NO,ISTATUS=ACTIVE,NEWSYNC=NO,          C
               CLOCKNG=EXT,DUPLEX=FULL,AVGPB=2046,DATRATE=HIGH,         C
               LIC=3
*              STATOPT=('INN TO PYELD032',NOMONIT)
PYEC0C   PU PUTYPE=4,DATMODE=FULL,MAXOUT=127,IRETRY=NO,PASSLIM=127,     C
               MODULO=128,                                              C
               TGN=(1),RETRIES=(,10,5),SRT=(32768,50),ISTATUS=ACTIVE
*              STATOPT=('STN TO PYECD032',NOMONIT)
*-----------------------------------------------------------------------*
* TOKEN-RING GENERATION FOR 3745 - SWITCHED LINES / NTRI TIC1          *
*                                                                       *
* DO NOT FORGET TO ADD MXVLINE AND MXRLINE PARAMETERS IN BUILD MACRO   *
*-----------------------------------------------------------------------*
*-----------------------------------------------------------------------*
* PHYSICAL GROUP FOR TOKEN RING LINES - NTRI TIC1                       *
*-----------------------------------------------------------------------*
*
PYECTICP GROUP ECLTYPE=PHYSICAL        PHYSICAL GROUP
*
*-----------------------------------------------------------------------*
* LINE SPECIFICATIONS                                                   *
*-----------------------------------------------------------------------*
PYECLPT1 LINE  ADDRESS=(1088,FULL),    TIC LOCATION                     C
               PORTADD=0,              REF FOR LOGICAL LINE PHYPORT     C
               LOCADD=400078839899,    LOCALLY ADMINISTERED ADDRESS     C
               RCVBUFC=4095,           RECOMMENDED BUFFER CAPACITY      C
               MAXTSL=521              TRANSMIT FRAME SIZE
*              STATOPT=('TOK RING TIC1 ',NOMONIT)
*-----------------------------------------------------------------------*
* PU - LU SPECIFICATIONS                                                *
*-----------------------------------------------------------------------*
PYECPPT1 PU
PYECUPT1 LU    ISTATUS=INACTIVE
*-----------------------------------------------------------------------*
* LINE SPECIFICATIONS                                                   *
*-----------------------------------------------------------------------*
PYECLPT2 LINE  ADDRESS=(1089,FULL),    TIC LOCATION                     C
               PORTADD=1,              REF FOR LOGICAL LINE PHYPORT     C
               LOCADD=400078839898,    LOCALLY ADMINISTERED ADDRESS     C
               RCVBUFC=4095,           RECOMMENDED BUFFER CAPACITY      C
               MAXTSL=521              TRANSMIT FRAME SIZE
*              STATOPT=('TOK RING TIC2 ',NOMONIT)
*-----------------------------------------------------------------------*
* PU - LU SPECIFICATIONS                                                *
*-----------------------------------------------------------------------*
PYECPPT2 PU
PYECUPT2 LU    ISTATUS=INACTIVE
*-----------------------------------------------------------------------*
* LOGICAL GROUP FOR TOKEN RING LINES - NTRI TIC1                        *
*-----------------------------------------------------------------------*
PYECTL1  GROUP ECLTYPE=LOGICAL,        LOGICAL GROUP                    C
               AUTOGEN=100,            100 LOGICAL LINES/PUS            C
               CALL=INOUT,             ALLOW DIAL-IN & DIAL-OUT         C
               PHYPORT=0               TIC PORTADDR
*              STATOPT=('TIC1 LOGICAL',NOMONIT)
*-----------------------------------------------------------------------*
* LOGICAL GROUP FOR TOKEN RING LINES - NTRI TIC1                        *
```

```
*--------------------------------------------------------------------*
PYECTL2  GROUP ECLTYPE=LOGICAL,        LOGICAL GROUP                  C
               AUTOGEN=100,            100 LOGICAL LINES/PUS          C
               CALL=INOUT,             ALLOW DIAL-IN & DIAL-OUT       C
               PHYPORT=1               TIC PORTADDR
*              STATOPT=('TIC2 LOGICAL',NOMONIT)
**********************************************************************
* CHANNEL ADAPTER GROUP                                              *
**********************************************************************
PYEGCA   GROUP LNCTL=CA,NCPCA=ACTIVE
PYELCA   LINE ADDRESS=0,ISTATUS=INACTIVE
PYELPU   PU
**********************************************************************
* GENERATION DELIMITER MACRO INSTRUCTION                             *
**********************************************************************
         GENEND INCHI=DTM45IR,                                       C
               INCINIT=DTM45IIN,                                     C
               INCL2HI=DTM45IHI,                                     C
               INCL2LO=DTM45ILO,                                     C
               ORDHI=DTM45OR,                                        C
               ORDINIT=DTM45OIN,                                     C
               ORDL2LO=DTM45OLO,                                     C
               ORDL2HI=DTM45OHI,                                     C
               TMRTICK=(DTMTIMTK,BALTICK,ECLTICK),                   C
               UGLOBAL=(DTMNEOR,BALNMGOP,ECLUGBL),                   C
               SRCLO=CBLOCK,                                         C
               INIT=(DTMINIT,BALINIMD,ECLINIT,CXNNINI)
```

Appendix G. ALCI sample definition

This section shows a sample definition for an ALCI test network.

```
*********************************************************************
* ALCI MACRO ASSEMBLY DECK                                          *
*********************************************************************
         DTMCHECK REPORT=FULL
PYECALCI DTMHOSTS GROUPNM=PYEGCALC,     PHYSICAL LINK GROUP NAME       C
               MODEL=3745-170,          CONTROLLER                     C
               LINENME=PYELALCI,        VIRTUAL LINK (ALCI)            C
               SOTNAME=PYECVSOT,        VIRT LINK SERVICE ORDER        C
               APPL=(PYEIALCI,PYEUA01,PYEUA02), ALCI SLU               C
               BFRS=240,                NCP BUFFER SIZE                C
               DLOGMOD=MNEF2,           ALCI TYPE LOGMODE              C
               GRPDMNM=PYEGCV,          VIRTUAL LINK GROUP NAME        C
               ISTATUS=ACTIVE,          ALCI INITIAL STATUS            C
               LEID=NONE,               LEID GENERATION                C
               LPDATS=LPDA2,            7861 MODEM SUPPORT             C
               LOGAPPL=(PYEZ1KT,PYEZ1ISW), ESTABLISH HOST ASSOCIATION  C
               MAXDATA=4096,            MUST BE GT LARGEST PIU         C
               MODETAB=CWTTY,           LOGON MODE TABLE               C
               ORDER=(PYEIALCI),        VIRT LINK PU                   C
               SPEED=9600,              LINE SPEED                     C
               TABEXP=50,               DYN. RECONFIG.                 C
               TIDRP=20                 DYN. RECONFIG.
*
* DEFINITION FOR PC/TRAVEL
*
PYEL16   DTMLINE ADDRESS=16,            REAL ALC LINE                  C
               LINENME=PYEL16,          UACB LINE NAME                 C
               SOTNAME=PYE16SOT,        SOT NAME                       C
               DUPLEX=FULL,             DEFAULT                        C
               EOT=0,                   QUIESCE DELAY (CHARS)          C
               ETRATIO=1,               FOR NETVIEW                    C
               ISTATUS=ACTIVE,          VTAM STATUS                    C
               MAXLIST=3,               ADDITIONAL ENTRIES IN SOT      C
               MAXPU=4,                 TOTAL OF STATIC & DR PUS       C
               NEWSYNC=NO,              MULTIPOINT CONNECTIONS ONLY    C
               ORDER=(PYEC161),                                        C
               PAUSE=0.5,               MEDIUM LINE LOAD               C
               REPLYTO=2.0,             DEFAULT                        C
               RETRIES=(1,1,1),         FOR N/W MAN. TEST              C
               SERVLIM=4,               DEFAULT                        C
               TRANSFR=15               DEFAULT
PYEC161  DTMPOLIA ADDR=10,              PC/TRAVEL TI                   C
               ISTATUS=ACTIVE,          VTAM STATUS                    C
               LELIST=(R01,04),         LEIDS                          C
               SESS=(PYEUA01),                                         C
               TALIST=(02,10,12,14)     TAS
*
* DEFINITION FOR A2CS
*
PYEL17   DTMLINE ADDRESS=17,            REAL ALC LINE                  C
               LINENME=PYEL17,          UACB LINE NAME                 C
               SOTNAME=PYE17SOT,        SOT NAME                       C
               DUPLEX=FULL,             DEFAULT                        C
               EOT=0,                   QUIESCE DELAY (CHARS)          C
               ETRATIO=10,              =1.0% ERROR TO TRAFFIC         C
               ISTATUS=ACTIVE,          VTAM STATUS                    C
               MAXLIST=3,               ADDITIONAL ENTRIES IN SOT      C
               MAXPU=4,                 TOTAL OF STATIC & DR PUS       C
               NEWSYNC=NO,              MULTIPOINT CONNECTIONS ONLY    C
               ORDER=(PYEC171),                                        C
               PAUSE=0.2,               HEAVILY LOADED LINE            C
               REPLYTO=2.0,             DEFAULT                        C
               RETRIES=(1,1,1),                                        C
               SERVLIM=4,               DEFAULT                        C
               TRANSFR=15               DEFAULT
PYEC171  DTMPOLIA ADDR=11,              A2CS TI                        C
               ISTATUS=ACTIVE,          VTAM STATUS                    C
               LELIST=(I10,36),         DEVELOPMENT GROUP              C
               SESS=(PYEUA01,PYEUA02),  ALCI/ALCS HOSTS                C
               TALIST=(I01,36)          TAS
*
```

© Copyright IBM Corp. 2003, 2019

```
* DEFINITION FOR A2CS
*
PYEL18   DTMLINE ADDRESS=18,            REAL ALC LINE                  C
               LINENME=PYEL18,          UACB LINE NAME                 C
               SOTNAME=PYE18SOT,        SOT NAME                       C
               DUPLEX=FULL,             DEFAULT                        C
               EOT=0,                   QUIESCE DELAY (CHARS)          C
               ETRATIO=10,              =1.0% ERROR TO TRAFFIC         C
               ISTATUS=ACTIVE,          VTAM STATUS                    C
               MAXLIST=3,               ADDITIONAL ENTRIES IN SOT      C
               MAXPU=4,                 TOTAL OF STATIC & DR PUS       C
               NEWSYNC=NO,              MULTIPOINT CONNECTIONS ONLY    C
               ORDER=(PYEC181),                                        C
               PAUSE=0.5,               MEDIUM LINE LOAD               C
               REPLYTO=2.0,             DEFAULT                        C
               RETRIES=(1,1,1),         N/W MAN. TEST.                 C
               SERVLIM=4,               DEFAULT                        C
               TRANSFR=15               DEFAULT
PYEC181  DTMPOLIA ADDR=12,              A2CS                           C
               ISTATUS=ACTIVE,          VTAM STATUS                    C
               LELIST=(I50,12),         LEIDS                          C
               SESS=(PYEUA02,PYEUA01),  ALCI/ALCS HOSTS                C
               TALIST=(I01,12)          TAS
PYEC182  DTMCASIA ADDR=16,              A2CS 2ND TI                    C
               LELIST=(I70,06),                                        C
               TALIST=(I01,6),                                         C
               SESS=(PYEUA02,PYEUA01)   ALCI/ALCS HOSTS
*
* DEFINITION FOR A2CS
*
PYEL19   DTMLINE ADDRESS=19,            REAL ALC LINE                  C
               LINENME=PYEL19,          UACB LINE NAME                 C
               SOTNAME=PYE19SOT,        SOT NAME                       C
               DUPLEX=FULL,             DEFAULT                        C
               EOT=0,                   QUIESCE DELAY (CHARS)          C
               ETRATIO=10,              =1.0% ERROR TO TRAFFIC         C
               ISTATUS=ACTIVE,          VTAM STATUS                    C
               MAXLIST=3,               ADDITIONAL ENTRIES IN SOT      C
               MAXPU=4,                 TOTAL OF STATIC & DR PUS       C
               NEWSYNC=NO,              MULTIPOINT CONNECTIONS ONLY    C
               ORDER=(PYEC191,PYEC192),                                C
               PAUSE=0.2,               HEAVILY LOAD LINE              C
               REPLYTO=2.0,             DEFAULT                        C
               RETRIES=(1,1,1),         N/W MAN. TEST                  C
               SERVLIM=4,               DEFAULT                        C
               TRANSFR=15               DEFAULT
PYEC191  DTMPOLIA ADDR=13,              GATEWAY AND 1ST TI             C
               LELIST=(I60,06),                                        C
               TALIST=(I01,6),                                         C
               SESS=(PYEUA01,PYEUA02),  ALCI/ALCS HOSTS                C
               ISTATUS=ACTIVE
PYEC192  DTMPOLIA ADDR=14,              A2CS 2ND TI                    C
               LELIST=(I67,06),                                        C
               TALIST=(I01,6),                                         C
               SESS=(PYEUA01,PYEUA02),  ALCI/ALCS HOSTS                C
               ISTATUS=ACTIVE
         DTMEND
         DTMCHECK
```

Appendix H. Generating a test ALC terminal

You can define test ALC terminals in the ALCS communication generation, in any of the following ways.

Note: Only STV can input data from a test ALC terminal. When you send a message to a test ALC printer terminal, ALCS automatically simulates an acknowledgment.

Method 1

Specify TEST=YES on the COMDEF macroinstruction that defines one or more AX.25 ALC terminals (LDTYPE=X25ALC). Also specify TEST=YES on the COMDEF macroinstruction that defines the owning X.25 PVC for these terminals (LDTYPE=X25PVC).

Method 2

Specify TEST=YES,NEFLU=*name* on the COMDEF macroinstruction that defines a NEF or ALCI terminal (LDTYPE=VTAMALC), where *name* is the CRN of the owning NEF or ALCI LU for this terminal. Also specify TEST=YES on the COMDEF macroinstruction that defines the owning NEF or ALCI LU for this terminal (LDTYPE=VTAMALC,TERM=NEFLU).

Method 3

"COMDEF parameters for a test ALC terminal" on page 575 describes another way to define a test ALC terminal ALCS supports this for compatibility with previous versions and releases of ALCS.

COMDEF parameters for a test ALC terminal

Use this COMDEF format to define a test ALC terminal. Only STV can input data from this resource. You must define a test ALC LU as well. A test ALC LU can be associated with many test ALC terminals.

```
[label]  COMDEF LDTYPE=VTAMALC
                        ,TEST=YES
                        ,NAME=base_name
                        ,APPL=application_name
                        ,TERM={1977|1980|2915|4505|user_device}
                        ,IA=ia
                        ,TA=ta
                        ,NEFLU=lu_name
                        [,ASSDEV=name]
                        [,BUFSZE={size}]
                        [,CRAS=ATnnn]
                        [,CSID=terminal_id]
                        [,ISTATUS={ACTIVE|INACTIVE}]
                        [,SCRSZE=rows]
                        [,TAB={NO|YES}]
                        [,TIMEOUT=({50|time1],[300|time2],[3|countl)]
                        [,UNSLITE={NO|YES}]
                        [,UPDATE={ADD|REPLACE|DELETE}]
```

Where:

NAME=*base_name*
Base CRN for the terminal. An alphanumeric string of 1 to 4 characters; the first character must be alphabetic. The stage 1 generation builds the full CRN for the terminal. The terminal CRN is made up of:

- The base name (NAME)
- The 2-character terminal address (TA)

© Copyright IBM Corp. 2003, 2019

- The 2-character interchange address (IA)

The full CRN for the terminal must be unique within ALCS.

TERM={1977|1980|2915|4505|*user_device*}
Terminal device type.

1977
ALC keyboard printer

1980
ALC receive only printer

2915
ALC display (12 lines)

4505
ALC display (30 lines)

Use SCRSZE to override the display size.

NEFLU=*lu_name*
The LU name of the test ALC LU associated with the terminal.

For other parameters, see Table 13 on page 110; and see also "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

COMDEF parameters for a test ALC LU

Use this COMDEF format to define a test ALC LU.

```
[label]  COMDEF LDTYPE=VTAMALC
                          ,TEST=YES
                          ,NAME=lu_name
                          ,TERM=NEFLU,
                          [,ISTATUS={ACTIVE|INACTIVE}]
                          [,UPDATE={ADD|REPLACE|DELETE}]
                          [,USERDAT=user_data]
```

Where:

NAME=*name*
CRN of the test ALC LU. An alphanumeric string of 1 to 8 characters; the first character must be alphanumeric. It must be unique within ALCS.

For other parameters, see Table 13 on page 110; and see also "Setting defaults for COMDEF parameters - COMDFLT macro" on page 105.

Appendix I. Sample definition for GDG sequential file

An ALCS sequential file can be a generation data group (GDG) member. There are different requirements for defining GDG data sets depending whether Storage Management Subsystem (SMS) is used in your installation or not. For more information about using generation data groups, see the IBM publication *z/OS DFSMS Using Data Sets*, SC26-7410. You can access this publication at http://publibz.boulder.ibm.com/epubs/pdf/dgt2d430.pdf.

Example for an SMS-managed environment

1. Add a definition in the ALCS sequential file generation.

```
      SEQGEN NAME=GDG,       OUTPUT TO DISK                    -
          TYPE=GENERAL,                                        -
          UNIT=(SYSDA,1),                                      -
          DISP=(NEW,CATLG,CATLG),                              -
          DSNAME=ALCS.SMS.GDG(+1),                             -
          LABEL=(,,,OUT,RETPD=32),                             -
          RECFM=U,                                             -
          VOLCNT=1,                                            -
          BLKSIZE=2048,                                        -
          SPACE=(20000,1)
```

2. Create a catalog entry for the GDG base.

```
//DEFGDG1 EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
    DEFINE GENERATIONDATAGROUP (NAME(ALCS.SMS.GDG) -
    SCRATCH NOEMPTY LIMIT(9))
/*
```

The GDG members will be allocated data set names like this:

```
ALCS.SMS.GDG.G0001V00
ALCS.SMS.GDG.G0002V00
⋮
```

Example for a non-SMS-managed environment

1. Add a definition in the ALCS sequential file generation.

```
         SEQGEN NAME=GDG,          OUTPUT TO DISK                  -
               TYPE=GENERAL,                                       -
               UNIT=(SYSDA,1),                                     -
               DISP=(NEW,CATLG,CATLG),                             -
               DSNAME=ALCS.NONSMS.GDG(+1),                         -
               LABEL=(,,,OUT,RETPD=32),                            -
               RECFM=U,                                            -
               VOLCNT=1,                                           -
               BLKSIZE=2048,                                       -
               SPACE=(20000,1)
```

2. Create a catalog entry for the GDG base.

```
//DEFGDG1 EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
     DEFINE GENERATIONDATAGROUP (NAME(ALCS.NONSMS.GDG) -
     SCRATCH NOEMPTY LIMIT(9))
/*
```

© Copyright IBM Corp. 2003, 2019

3. Create a pattern DSCB for the GDG generation.

   This DSCB must be allocated on the same disk where the catalog entry for the GDG base resides.

```
//DEFPATT1 EXEC PGM=IEFBR14
//DD1      DD DSNAME=ALCS.NONSMS.GDG,SPACE=(TRK,(0)),UNIT=SYSDA,
//            VOL=SER=UCAT01,DISP=(NEW,KEEP)
```

   Notes:

   a. DSNAME specifies the GDG data set name without the generation and version number. It is the same DSNAME used in your ALCS sequential file generation.
   b. VOL=SER= specifies the serial number of the volume where the User Catalog for the alias 'ALCS' resides.
   c. DISP=(NEW,KEEP) ensures the pattern DSCB is not cataloged.
   d. You will need a separate pattern DSCB for each different GDG data set defined in your ALCS sequential file generation.

The GDG members will be allocated data set names like this:

```
ALCS.NONSMS.GDG.G0001V00
ALCS.NONSMS.GDG.G0002V00
⋮
```

Appendix J. Sample definition for OCTM sequential file

The following is an example of the SEQGEN sequential file definitions that could be used to define the output sequential file that is required by the ZOCTM BACKUP (OCTM backup) function and the input sequential file that is required by the ZOCTM RESTORE (OCTM restore) 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

SEQGEN NAME=CMR,
      TYPE=GEN,
      UNIT=(3380,1),
      DISP=(OLD,KEEP,KEEP),
      DSNAME=.............CMB,
      LABEL=(,,,IN,RETPD=0),
      BUFNO=4
```

© Copyright IBM Corp. 2003, 2019

Appendix K. Communications End User System (CEUS)

The CEUS provides the user interface for the OCTM facility. This section contains an overview of the CEUS, provides detailed information on the sample CEUS application, and also provides guidance on the type of functionality that you may wish to include in your own CEUS application.

CEUS Overview

The CEUS is comprised of ECB-controlled programs which receive communication change requests from end users and issue COMTC macros for those change requests. At most ALCS installations, the OCTM end users will be system administrators who are responsible for accepting communications change requests, checking them and then entering them into the OCTM facility via the CEUS.

The CEUS will be unique for each ALCS system because it must provide functionality that meets the specific operational and network requirements of the ALCS system. Each ALCS system will have its own unique requirements regarding the procedures which should be followed when applying changes via OCTM to the online communication table. Each ALCS system will also have its own unique layout and format for the communication table user data area.

Although the CEUS must include some ALCS ECB-controlled programs (for issuing the COMTC macro) the complete CEUS software could be split over multiple platforms. In particular, the software that provides the end user interface (for example, the screen formats used for entering details of new terminals) does not need to reside in ALCS and could be developed on another platform. It could be a GUI interface on a workstation, using a TCP/IP link to ALCS. It could also be developed using the ALCS 3270 screen mapping package.

Sample CEUS Application

A sample CEUS application is provided by IBM on the ALCS web site. A further sample CEUS application can be obtained from one of the ALCS users (contact IBM for more details on this). Both of these sample CEUS applications are available to the ALCS users on an ***as is*** basis. Each ALCS user can develop a CEUS using one of the supplied CEUS applications as the primary building block, or alternatively they can develop their own.

The sample CEUS developed by IBM is a 3270 map application which provides the basic functions that are necessary for interaction with the ALCS OCTM facility. Those basic functions include:

- Utilizing ALCS 3270 screen mapping support, providing formatted screens for inputting communications data
- Displaying details of communication resources managed by OCTM
- Providing screen maps that allow every type of communication data to be input to OCTM
- Utilizing the COMTC macro to submit change requests to OCTM (add, replace and delete communication resources) and to activate those change requests
- Reporting error conditions

Although the sample CEUS application provides only the basic functions, it does exercise and demonstrate the use of the COMTC macro. Each entry in the sample application is atomic. Input is received, an action performed, and another map is displayed. Nothing is held by ALCS relating to a session and no information is stored in the end user's terminal related records. The sample CEUS operates as a stateless application.

When each ALCS user develops their own CEUS application, it may need be more sophisticated than the sample CEUS developed by IBM. The additional functionality that ALCS users might require in their CEUS could include the following:

- Help information for each of the screen maps
- A security interface for ensuring that only authorized system administrators use the CEUS application
- Status information on each communications group, maintained in database records
- Limitations on the usage of some COMTC macro parameters

The section called "Limitations of Sample CEUS" on page 589 provides information on the limitations of the sample CEUS and includes suggestions on how you may wish to remove those limitations when you develop your own CEUS application. The section called "Additional CEUS Functionality" on page 590 contains suggestions on the additional functionality that could be included in your CEUS application.

Components of Sample CEUS

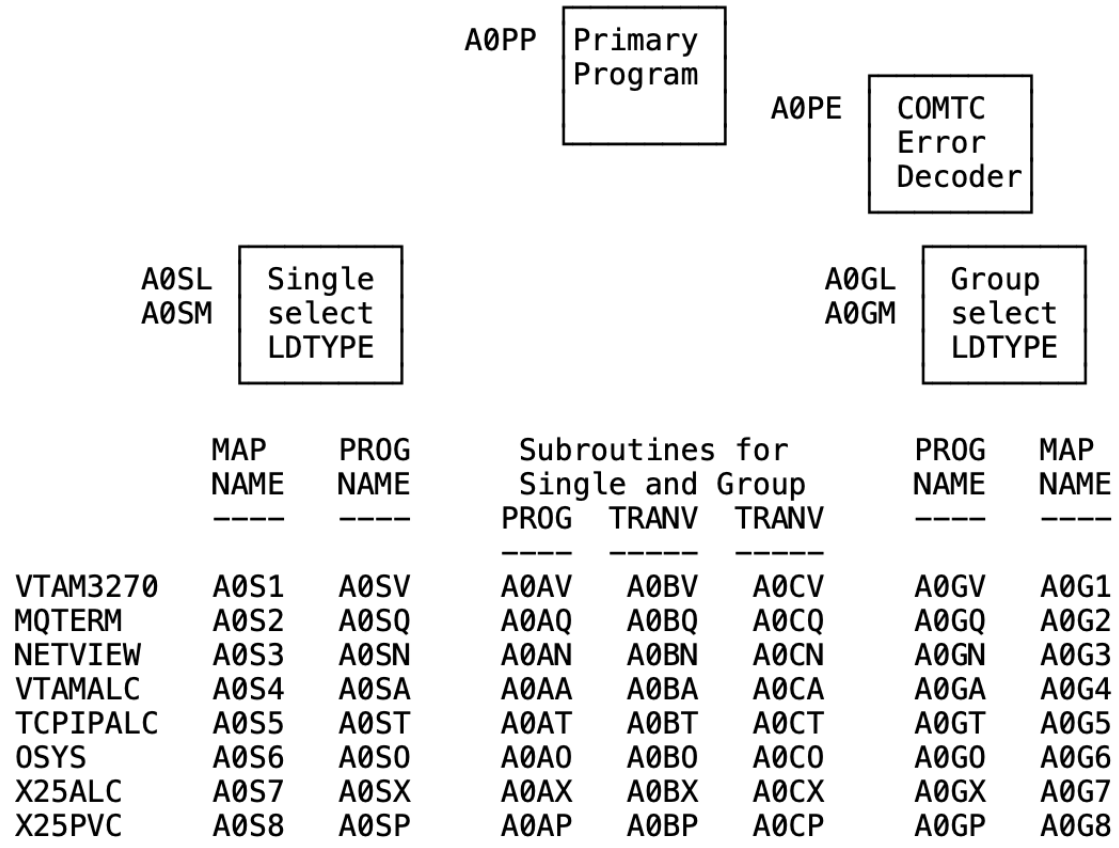
The primary ECB-controlled program for the sample CEUS application (the input editor program) is called A0PP. This program determines if the input is from one of the sample CEUS screen maps, and if it is, may enter the ECB-controlled program A0SL or A0GL based on the input map name. Similarly, programs A0SL and A0GL may route to subsequent programs based on the map used in the input message. Each map has a displayed field (called XMAP) at the same position in the screen map, and this field contains the map name. The primary screen map for the sample CEUS is called A0P0. All other screen maps are called A0Gx or A0Sx (where the last digit of the map name is numeric).

If the input message contains one of the Communications Group screen maps (the A0Gx maps) then the message will be processed by one of the A0Gx programs. If the input message is for a single communication resource (and is not associated with a communications group) it will contain an A0Sx screen map and will be processed by one of the A0Sx programs. As most of the information handled is common for the A0Gx and the A0Sx programs, they share the same subroutine program A0Ax. The A0Ax subroutine program contains two transfer vectors, A0Bx and A0Cx.

There are programs and maps for each ALCS communication device type. For example, the VTAM 3270 device type is managed by programs A0GV, A0SV and A0AV, and by screen maps A0G1 and A0S1. The screen map members contain ALCS MAP3270 macroinstructions. These macroinstructions create map DSECTs which define the input and output fields on the 3270 screen. They specify attributes of the screen (size, and so on) and the start position and length of each map field and its initial contents.

When you are altering or extending the sample CEUS application, the map fields need to be maintained at the same offset (for example, maps A0S1 and A0G1 have all modifiable fields at the same offset). All the screen maps contain fields called XOPTION, XNAME, XCRI, XPOPUP, MCRI, MCRIBASE, MORD and MORDBASE at the same offset so that thay can share the same routines in the ECB-controlled programs.

The sample CEUS application is made up of nineteen 3270 screen maps and twenty-eight ECB-controlled programs. The names used are illustrated in the following diagrams.



	MAP NAME	PROG NAME	Subroutines for Single and Group PROG	TRANV	TRANV	PROG NAME	MAP NAME
VTAM3270	A0S1	A0SV	A0AV	A0BV	A0CV	A0GV	A0G1
MQTERM	A0S2	A0SQ	A0AQ	A0BQ	A0CQ	A0GQ	A0G2
NETVIEW	A0S3	A0SN	A0AN	A0BN	A0CN	A0GN	A0G3
VTAMALC	A0S4	A0SA	A0AA	A0BA	A0CA	A0GA	A0G4
TCPIPALC	A0S5	A0ST	A0AT	A0BT	A0CT	A0GT	A0G5
OSYS	A0S6	A0SO	A0AO	A0BO	A0CO	A0GO	A0G6
X25ALC	A0S7	A0SX	A0AX	A0BX	A0CX	A0GX	A0G7
X25PVC	A0S8	A0SP	A0AP	A0BP	A0CP	A0GP	A0G8

Figure 129. Names of ECB-controlled programs

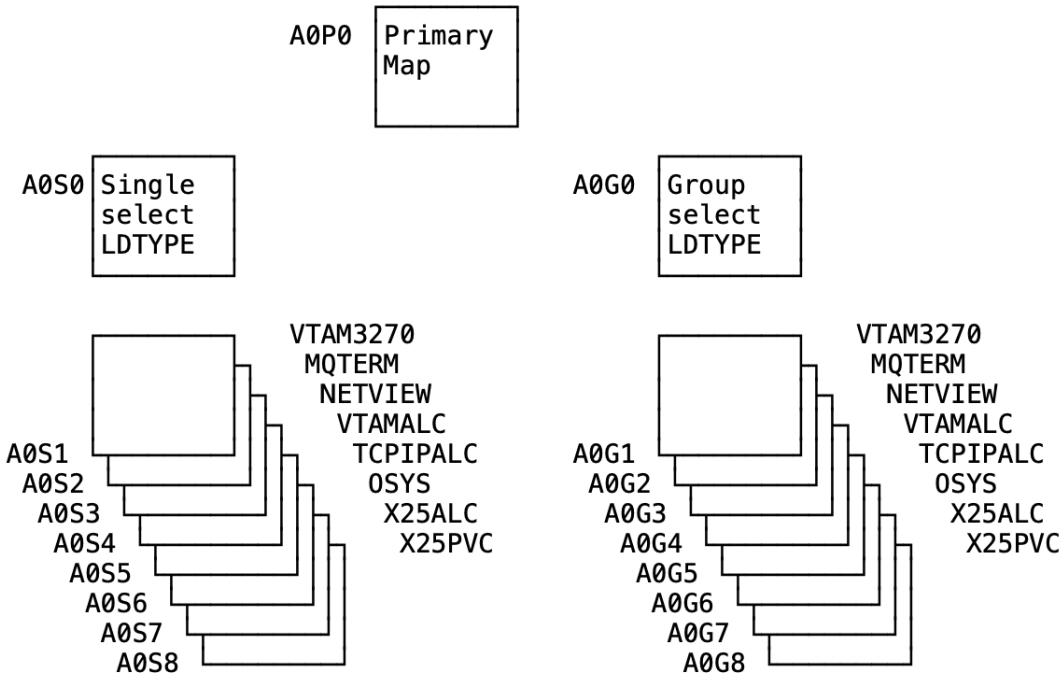


Figure 130. Names of 3270 screen maps

Installing the Sample CEUS

The following describes the primary tasks required to install the sample CEUS application on your ALCS system.

- `Install the sample CEUS 3270 screen map members`

  Install the 19 3270 screen map members in a source statement library. The names of these members are A0S0 to A0S8 (9 members), A0G0 to A0G8 (9 members) and A0P0.

- `Install and customize the sample CEUS A01PP member`

  Install the A01PP copy member in a source statement library. The A01PP copy member contains two items that you may wish to alter before assembling the sample CEUS programs.
  The first item in A01PP is the name of the sample CEUS application. That is currently defined as OCTM so that it matches the communications generation COMDEF macro described below. You can select any 4-character name for the sample CEUS application and define that name in the A01PP member and the COMDEF macro.
  The second item in A01PP is the name of the application that the sample CEUS will return to when exiting. That is currently defined as RES0 because most ALCS customers use the application name of RES0 for the primary ALCS application. If you require the sample CEUS to return to a different application when you exit from the sample CEUS, then define the name of that application in the A01PP member.

- `Install and assemble the sample CEUS ECB-controlled programs`

  Install the twenty-eight ECB-controlled programs in a source statement library. The names of these programs are A0Ax (8 members), A0Gx (9 members), A0Sx (9 members), A0PP and A0PE. The last character of each program name is alphabetic. These programs should be assembled against your ALCS macro libraries etc. Ensure that you include in your SYSLIB the source statement library that contains the screen map members and the A01PP member.

- `Create load module for the sample CEUS programs`

  Verify that each ECB-controlled program has assembled without error. Create an application program load module for the twenty-eight programs in the sample CEUS (to be loaded via the ZPCTL command).

- `Create sequential file for the screen maps`

  Assemble each screen map member with the assembler parameter SYSPARM(GENERATE), using the MVS JCL parameter PARM. Copy the SYSPUNCH output from this assembly to a sequential data set using the IEBGENER utility program. The ALCS ISPF panels can be used for running these two jobs. Alternatively, extract the sample JCL from the ISPF panels, update it and then run these two jobs.

- `Load the sample CEUS programs and maps onto your ALCS system`

  Verify that your ALCS sequential file configuration table contains a definition for the screen map sequential file. Load the screen map sequential file using the ZCMSP command. Load the application program load module using the ZPCTL command.

- `Define sample CEUS application in ALCS communications generation`

  In order to invoke the sample CEUS application, it must be defined to ALCS in a communication generation COMDEF macro. The following is an example of the COMDEF macro that should be coded when the application name is OCTM.

```
COMDEF LDTYPE=ALCSAPPL,NAME=OCTM,PROG=A0PP,SYSSTATE=IDLE,ISTATUS=ACTIVE
```

  When the ALCS communications table contains this definition, the ZROUT OCTM command can be used to route the 3270 terminal to the sample CEUS application.

Using the Sample CEUS

The first step that is required when you are ready to start using the sample CEUS is to route your 3270 terminal to the application. Use the ZROUT command to do this. If you have installed the sample CEUS

with the application name defined as OCTM, use the ZROUT OCTM command to enable your 3270 to access the sample CEUS. After you receive the response to ZROUT, just hit the enter key, and this will display the primary screen map.

The following sections describe how the different screen maps are used.

A0P0 - The Primary Screen Map

There are five options provided on this primary map.

- Option 1 - List Groups

  Enter 1 in the option field of the map and hit enter. This will display a list of the currently allocated communications groups. The sample CEUS uses the COMTC GROUPS macro to obtain this list. When a system administrator needs to allocate a new communications group, they may need to determine if the communications group name they are planning to use for the new group is already in use. They may also wish to verify how many groups are currently active. The COMTC GROUPS macro returns to the sample CEUS information about each communications group in the Communications Groups Information DSECT (CT3TM). Although COMTC provides status information about each group, the sample CEUS only displays the name of the first 86 allocated groups. If more information is required, use the ZOCTM GROUPS command.

- Option 2 - Work with a Group

  Enter 2 in the option field of the map and enter the name of the group in the Groupname field. This will display the A0G0 screen map which enables you to start working with this communications group. The A0G0 screen map is described in “A0G0 - Screen Map for Managing a Group” on page 586.

- Option 3 - Allocate Group

  Enter 3 in the option field of the map and enter the name of the group to be allocated in the Groupname field. The sample CEUS uses the COMTC ALLOCATE macro to allocate the group name. You should allocate a communications group name when you need to work with a batch of communication change requests. If you use a communications group when submitting multiple change requests to OCTM, those change requests can be activated in the ALCS communications table as a single group. A unique name must be given for the communications group (it can be a maximum of 7 characters) and it must be allocated before any change requests are submitted for the group.

- Option 4 - Unallocate Group

  Enter 4 in the option field of the map and enter the name of the group to be unallocated in the Groupname field. The sample CEUS uses the COMTC UNALLOCATE macro to unallocate the communications group. You should normally unallocate a communications group after the change requests belonging to that group have been committed. When a communications group has been committed, there will normally be no further requirement to retain the group (or the group name), therefore you should unallocate it soon after the commit.

  There is a feature of COMTC UNALLOCATE which is not used by the sample CEUS. If a number of change requests have been submitted for a group and the group is not yet loaded (or has been backed out), a COMTC UNALLOCATE can be used to delete the group and cancel all the change requests in that group. This may be required if a batch of new terminals have been incorrectly added and all the change requests for that batch require cancelling. If you wish to use this feature, you could enhance this A0P0 screen map to include an additional option that will provide this feature.

- Option 5 - Work on single resource

  Enter 5 in the option field of the map (and hit the enter key) when you do not wish to work with a communications group and need to issue a change request for a single communication resource. This will display the A0S0 screen map which enables you to start working with a single communication resource. The A0S0 screen map is described below.

Using the Sample CEUS

The first step that is required when you are ready to start using the sample CEUS is to route your 3270 terminal to the application. Use the ZROUT command to do this. If you have installed the sample CEUS with the application name defined as OCTM, use the ZROUT OCTM command to enable your 3270 to access the sample CEUS. After you receive the response to ZROUT, just hit the enter key, and this will display the primary screen map.

The following sections describe how the different screen maps are used.

A0S0 - Screen Map for Single Resource Selection

There are eight options provided on this screen map. Each option is for a specific communications resource type. For example, option 1 is for the VTAM 3270 terminal type, option 2 is for the MQ terminal type, etc. Select the resource type that you need to work with by entering the appropriate number in the option field and hitting the enter key. This will display one of the screen maps, A0S1 to A0S8. The A0S1 map is for VTAM 3270 terminals, the A0S2 map is for MQ terminals, and so on. The A0S1 to A0S8 screen maps are described below.

A0S1 to A0S8 - Screen Maps for Working with a Single Resource

There are nine options provided on each one of these screen maps.

- `Option 1 - Show a Resource`

  Enter 1 in the option field of the map and also enter either the name of the resource (the CRN) or the CRI address of the resource (in fields at the top of the screen). This will provide a detailed display of the communication resource. When a communication resource is to be changed or deleted, the system administrator should display and check the current definition and status of the resource in the ALCS online communication table. The sample CEUS uses the COMTC QUERY macro to obtain details of the communication resource and its current OCTM status. The COMTC QUERY macro returns the information in the Communications Resource and Group Information DSECT (CT2TM). When a new resource is being displayed, the sample CEUS displays the definition of the new resource plus status information about it. When a changed (replaced) resource is being displayed, the sample CEUS displays the definition of the modified resource.
  Before submitting a change (replace) request, use option 1 to display a screen map that contains a complete definition of the resource. Modify those parts of the definition that require changing, and then use option 3 to submit the change.

- `Options 2 to 5 - Submit Change Request`

  These four options are used for submitting different types of change request for the communication resource. You can use option 2 to add a new resource, option 3 to change an existing resource, or option 4 to delete an obsolete resource. The sample CEUS uses the COMTC ADD, COMTC REPLACE and COMTC DELETE macros to perform these functions. If you have incorrectly submitted a change request (for example, you have used the wrong resource name) you can cancel that change request by using option 5. The sample CEUS uses the COMTC CANCEL macro to cancel a previous COMTC ADD, COMTC REPLACE or COMTC DELETE. Refer to “Submitting communications change requests” on page 587 for more details on this.

- `Options 6 to 9 - Activate Change Request`

  These four options are used for activating the change request in the ALCS online communication table. When you have submitted the change request, you must activate it in the online communication table via a three stage process. Those three stages are called load, confirm and commit. Firstly, use option 6 to load the change request, secondly, use option 7 to confirm the change request, and lastly, use option 8 to commit the change request. The sample CEUS uses the COMTC LOAD, COMTC CONFIRM and COMTC COMMIT macros to do this. If you decide that the change request should not have been loaded, use option 9 to back out the change request. A change request can be backed out of the online communication table after either a load (option 6) or a confirm (option 7). It can not be backed out after

a commit (option 8). The sample CEUS uses the COMTC BACKUP macro to do this. Refer to "Activating communications change requests" on page 588 for more details on this.

A0G0 - Screen Map for Managing a Group

There are thirteen options provided on this screen map.

- Options 1 to 8 - Select Resource Type

  Each of these 8 options are for a specific communications resource type. For example, option 1 is for the VTAM 3270 terminal type, option 2 is for the MQ terminal type, etc. Select the resource type that you need to work with by entering the appropriate number in the option field and hitting the enter key. This will display one of the screen maps, A0G1 to A0G8. The A0G1 map is for VTAM 3270 terminals, the A0G2 map is for MQ terminals, and so on. The A0G1 to A0G8 screen maps are described in "A0G1 to A0G8 - Screen Maps for Working with each Resource" on page 586.

- Options L C M and B - Activate Change Requests

  These four options are used for activating a batch of change requests in the ALCS online communication table. When you have submitted all the change requests for this group, you must activate them in the online communication table via a three stage process. Those three stages are called load, confirm and commit. Firstly, use option L to load the change requests, secondly, use option C to confirm the change requests, and lastly, use option M to commit the change requests. The sample CEUS uses the COMTC LOAD, COMTC CONFIRM and COMTC COMMIT macros to do this. If you decide that this batch of change requests should not have been loaded, use option B to back out all the change requests. The change requests can be backed out of the online communication table after either a load (option L) or a confirm (option C). They can not be backed out after a commit (option M). The sample CEUS uses the COMTC BACKUP macro to do this. Refer to "Activating communications change requests" on page 588 for more details on this.

- Option S - Show Group Information

  Enter S in the option field of the map and hit enter. This will display a list of the communication resources that currently belong to this communications group. The sample CEUS uses the COMTC QUERY macro to do this. When a system administrator is submitting a large number of change requests, it can be advisable to occasionally display the list of resources for which change requests have been submitted. The COMTC QUERY macro not only provides the list of resources for which change requests have been submitted, but also provides information about each change request. The response output by the sample CEUS contains only the names of the resources.

A0G1 to A0G8 - Screen Maps for Working with each Resource

There are five options provided on this screen map.

- Option 1 - Show a Resource

  Enter 1 in the option field of the map and also enter either the name of the resource (the CRN) or the CRI address of the resource (in fields at the top of the screen). This will provide a detailed display of the communication resource. When a communication resource is to be changed or deleted, the system administrator should display and check the current definition and status of the resource in the ALCS online communication table. The sample CEUS uses the COMTC QUERY macro to obtain details of the communication resource and its current OCTM status. The COMTC QUERY macro returns the information in the Communications Resource and Group Information DSECT (CT2TM). When a new resource is being displayed, the sample CEUS displays the definition of the new resource plus status information about it. When a changed (replaced) resource is being displayed, the sample CEUS displays the definition of the modified resource.
  Before submitting a change (replace) request, use option 1 to display a screen map that contains a complete definition of the resource. Modify those parts of the definition that require changing, and then use option 3 to submit the change.

- Options 2 to 5 - Submit Change Request

These four options are used for submitting different types of change request for this communications group. You can use option 2 to add a new resource, option 3 to change an existing resource, or option 4 to delete an obsolete resource. The sample CEUS uses the `COMTC ADD`, `COMTC REPLACE` and `COMTC DELETE` macros to perform these functions. If you have incorrectly submitted a change request (for example, you have used the wrong resource name) you can cancel that change request by using option 5. The sample CEUS uses the `COMTC CANCEL` macro to cancel a previous `COMTC ADD`, `COMTC REPLACE` or `COMTC DELETE`. Further information on submitting communications change requests is provided below.

Submitting communications change requests

Options 2 to 4 are provided for submitting different types of change request. These change requests enable new communication resources to be added, current resources to be changed, and obsolete resources to be deleted. Option 5 is provided for cancelling an incorrect change request. The following describes each of these four options.

- `Option 2 - Add a communication resource`

  You can request OCTM to add a terminal resource (except Prime or RO Cras) or an X.25 PVC resource (types 1, 6 and 7). Different screen maps are required for each resource type because the type of communications data that must be provided is different for each type of resource. The screen maps contain fields for all the required communications data that is needed for new terminals and new X.25 PVC's. You do not need to complete every field in the screen map. The sample CEUS provides default values for many fields, although some fields must be completed. The sample CEUS forwards to OCTM the data you have entered via parameters in the `COMTC ADD` macro and via a Communication Resource Definition DSECT (CT1TM).

  Adding X.25 PVC and ALC resources require additional considerations. An X.25 ALC terminal can not be added if the definition for the owning X.25 PVC does not already exist on the OCTM database. If you need to add a new X.25 PVC and its associated X.25 ALC terminals, you can do this in either of the following two ways.

  1. Add the X.25 PVC resource on its own (do not use a communications group) and then load, confirm and commit that resource. Now add the associated X.25 ALC terminals (use a communications group for adding these terminals).
  2. Allocate a communications group and add the X.25 PVC and its associated X.25 ALC terminals in the same group. The X.25 PVC must be added before any of the terminals.

  When adding an X.25 ALC terminal resource, the resource name must be comprised of a base CRN plus one or two other elements depending on the type of X.25 PVC that the ALC terminal is associated with. There are three types of X.25 PVC that can have associated ALC terminals. The length of the base CRN for these ALC terminals is dependent on the type of X.25 PVC that they are associated with. When these terminals are associated with a ***Type 1*** X.25 PVC, the base CRN (a string of 1 to 6 characters) must have a two-character terminal address appended to it. When these terminals are associated with a ***Type 6*** or ***Type 7*** X.25 PVC, the base CRN (a string of 1 to 4 characters) must have a two-character interchange address followed by a two-character terminal address appended to it. Note that the interchange address (if required) and the terminal address must also be defined in the **TA**= and **IA**= fields on the screen map.

  In addition to the X.25 ALC terminal resource type, there are other terminal resource types that have owning resources. They are the MQ terminal, the TCPIPALC terminal, and the VTAMALC terminal. When a terminal belonging to one of these three resource types is added, the owning resource (MQ queue, TCPIP server connection or ALCI LU) must already be defined in the ALCS offline communications generation (and loaded in the online communication table).

- `Option 3 - Change a communication resource`

  You can request OCTM to change a terminal resource (except Prime or RO Cras) or an X.25 PVC resource (types 1, 6 and 7). You can not change the resource name (CRN), the CRI address or the resource ordinal number. If you need to change any of these, the resource must be deleted and added again. Different screen maps are required for each resource type because the type of communications

data that must be provided is different for each type of resource. The screen maps contain fields for all the required communications data that can be changed. Display the resource via option 1 before making any change, and then alter those fields on the screen where the data requires changing. The sample CEUS forwards to OCTM the complete definition for the resource (OCTM replaces the current definition with this updated one) via COMTC REPLACE macro parameters and a Communication Resource Definition DSECT (CT1TM).

- Option 4 - Delete a communication resource

  You can request OCTM to delete a terminal resource (except Prime and RO Cras) or an X.25 PVC resource (types 1, 6 and 7). If an X.25 PVC resource is being deleted, all its associated X.25 ALC terminal resources must also be deleted. If a batch of X.25 terminals must be moved from one PVC to another (and the CRI and ordinal number for each terminal is be retained), this must be achieved by deleting the terminals and adding them again. The sample CEUS forwards to OCTM the delete request via the COMTC DELETE macro.

- Option 5 - Cancel a communication change request

  You can request OCTM to cancel a change request that you have recently submitted The change request being cancelled must have been submitted via options 2, 3 or 4 (add, replace or delete). A change request can not be cancelled if it is currently loaded in the online communication table. If the change request has already been loaded (or has been confirmed) in the online communication table, you can request the sample CEUS to back it out. When the back out is complete, you can then cancel the change request. The sample CEUS forwards to OCTM the cancel request via the COMTC CANCEL macro.

Activating communications change requests

There are four options for activating change requests in the online communication table. The first three must all be performed to fully activate change requests in the online communication table. Those three are called load, confirm and commit. The last option enables a change request to be backed out. After a load or confirm, any change request can be backed out (removed) from the online communication table. The following describes each of these four options.

- Option 6 or L - Load communication change requests

  When option 6 is used (on screen maps A0S1 to A0S8) the sample CEUS requests OCTM to load a single change request. When option L is used (on screen map A0G0) the sample CEUS requests OCTM to load a group of change requests. The change requests are loaded into the online communication table. Change requests that have been loaded (but not confirmed) are not retained in the online communication table over an ALCS restart (although they are still retained in the OCTM database). If an ALCS restart occurs, you must request OCTM to load the change request(s) again.

- Option 7 or C - Confirm communication change requests

  After a successful load, you should request OCTM to confirm the change request (or group of change requests). When option 7 is used (on screen maps A0S1 to A0S8) the sample CEUS requests OCTM to confirm a single change request. When option C is used (on screen map A0G0) the sample CEUS requests OCTM to confirm a group of change requests. The change requests are confirmed in the online communication table. After a confirm, the change request(s) are retained in the online communication table over an ALCS restart.

- Option 8 or M - Commit communication change requests

  After a successful confirm, you should request OCTM to commit the change request (or group of change requests). When option 8 is used (on screen maps A0S1 to A0S8) the sample CEUS requests OCTM to commit a single change request. When option M is used (on screen map A0G0) the sample CEUS requests OCTM to commit a group of change requests. The change requests are committed in the online communication table. When change request(s) have been committed, they can not be backed out.

- Option 9 or B - Back out communication change requests

  When change request(s) that have been loaded or confirmed are causing problems on your ALCS system, you should request OCTM to back them out from the online communication table. When option 9 is used (on screen maps A0S1 to A0S8) the sample CEUS requests OCTM to back out a single change

request. When option B is used (on screen map A0G0) the sample CEUS requests OCTM to back out a group of change requests. The change requests are removed from the online communication table. A back out does not delete the change requests from the OCTM database, therefore when a group of change requests have been backed out, any change request(s) that were incorrect can be corrected (or cancelled) and the group loaded again.

Limitations of Sample CEUS

Each ALCS user will need to develop their own CEUS application. If the sample CEUS provided by IBM is to be used as the primary building block for the development of your own CEUS, then the limitations that currently exist in the sample CEUS may need to be removed as part of that development effort. The following describes some of the limitations.

- `Communications User Data`

  The processing of the communications user data will always be unique for each ALCS customer. The sample CEUS provides a single field in the screen maps for inputting user data. That field is USERDAT=, and it allows a maximum of 20 characters of data to be input. This field allows only alphanumeric characters to be input and does not support hex data.
  When you customize the sample CEUS for your ALCS system, you may wish to provide a field in the screen maps for each field in your communications user data area. You are recommended to validate your communications user data before submitting it to OCTM (via the CT1TM DSECT) because OCTM will not validate it for you. You may already use the offline communications generation DXCZCUSR macro for validating your user data, therefore you should replicate the DXCZCUSR functionality in your CEUS application.

- `Error and information messages`

  Most of the screen maps in the sample CEUS required many fields to be defined in them. This therefore resulted in only limited space being available in each screen map for the error and information messages that can be output by the sample CEUS. Some actions, such as loading a large number of change requests in a communications group, may generate many error messages. Although OCTM returns to the sample CEUS (via the `COMTC` macro) a storage block containing all the error messages, only the first error message is displayed in the screen map. On all of the screen maps, only one line is provided for displaying the error and information messages. A single error message will normally fit on a line, but some messages may be too long and therefore may be truncated. When errors occur during the loading or backing out of communication change requests, you may need to refer to other sources (RO CRAS, system log or diagnostic file) to understand fully the errors that occurred. In summary, `COMTC ADD` and `COMTC REPLACE` can return multiple error messages in a storage block, and so can `COMTC LOAD` and `COMTC BACKUP`.
  When you customize the sample CEUS for your ALCS system, you may wish to provide additional lines on your screen maps for the error and information messages.

- `Displaying a communications group`

  The OCTM facility provides a `COMTC QUERY` macro and a `ZOCTM` **`GROUP`**=*`groupname`* command for obtaining a list of the communication resources that belong to a communications group. OCTM allows a maximum of 400 resources in a group, but the sample CEUS displays the names of only the first 64. The `COMTC` macro and the `ZOCTM` command both indicate the type of change request that has been issued for each resource in the group, but the sample CEUS does not display that information.
  When you customize the sample CEUS for your ALCS system, you may wish to enhance the display of group information. Alternatively, you may wish the CEUS user to start a second 3270 session with ALCS and use that session for `ZOCTM` commands.

- `Other limitations in sample CEUS`

  There are a number of minor limitations which you may wish to review when customizing the sample CEUS for your ALCS system. The function for displaying a list of the allocated communications groups displays the first 86, although ALCS allows a maximum of 400 (the `COMTC` macro and the `ZOCTM GROUPS` command both provide all the group names). The function for displaying (showing) a communications resource will display only those resources managed by OCTM, although `COMTC QUERY` will provide information on any communication resource (the `ZDCOM` command can be used to

display any resource). The function for unallocating a communications group will not allow a group that contains uncommitted change requests to be unallocated (the COMTC UNALLOCATE macro provides the DELETE=YES option for allowing a group to be unallocated and all its associated change requests to be cancelled).

Additional CEUS Functionality

There are additional functions that could be included in the CEUS implemented by ALCS users. The following provides some suggestions on this.

Maintaining CEUS control records

The CEUS could maintain control records that would enable information to be stored on the ALCS database and retained over a group of CEUS transactions. Information about COMTC macros that have been issued for a communication resource or group could be saved, enabling additional validation checks to be performed. For example. when a system administrator allocates a communications group, if they had been required to logon with a user-ID (and password), the CEUS could associate the group with their user-ID and save this information in a CEUS control record. The CEUS could then verify that each change request that is submitted for that group is input by the same person (same user-ID). CEUS control records could also be used for maintaining an audit trail of the change requests submitted for a communications group (type of change request, time/date of change request, etc.).

Restricting usage of CEUS functions

The ALCS access control function could be used to restrict the usage of the CEUS to authorized system administrators. There may be some CEUS functions (for example, displaying information about a communication resource) that do not need to be restricted, but other functions may need to be restricted. A RACF ALCSAUTH profile could be created for the CEUS and system administrators whose user-IDs have access to that profile could be allowed to perform the restricted functions. The system administrator would be required to logon to the system with a user-ID and password before being given the initial screen map by the CEUS. The CEUS would then use the AUTHC macro to check a user's authority before performing specific functions.

Managing communications groups

The OCTM facility places some restrictions on the usage of communications groups, but additional restrictions may be required. For example:

- The total number of groups that can be active at any time could be restricted to 50 (OCTM allows 400).
- The total number of groups that each system administrator can have active at any one time could be restricted to 10.
- The total number of change requests that can be submitted for a group could be restricted to 100 (OCTM allows 400). If terminals are being deleted in a group, you may wish to restrict the number of deletes to 50.

Some of these restrictions could be managed by providing a CEUS control record for each communications group that is allocated. That control record could hold the user-ID of the person responsible for the group and keep an audit trail of every COMTC issued for the group. A senior system administrator could display the information in the control record when monitoring the status of active groups.

Managing CRIs and ordinals

The CEUS can request the OCTM facility to allocate an available CRI and ordinal number for a new communication resource (it can also explicitly define the CRI and ordinal number for a new resource). For example, if a terminal is being added by a COMTC ADD macro, the CRIBASE= and ORDBASE= parameters can be used to request OCTM to allocate an available CRI and ordinal.

If the base CRI address is defined as CRIBASE=020001, OCTM will allocate the next available CRI starting from 020001. Different CRIBASE values could be used for allocating CRI addresses for different resource types. For example, if CRIBASE=250000 is used on a COMTC ADD for an "other system" terminal, OCTM would allocate the next available CRI starting from 250000. Each ALCS installation has its own rules on the allocation of CRI addresses, but the CRI= and CRIBASE= parameters on COMTC ADD should provide ample flexibility on the selection of CRIs for new communication resources.

In a similar way, you can control the ordinal numbers allocated to new communication resources. The ORDBASE parameter requests OCTM to allocate the next available ordinal number. If ORDBASE=500 is used, OCTM will allocate the next available ordinal number starting at ordinal 500.

Using the OCTM policing exit

The OCTM policing function is activated once each hour for monitoring the contents of the base and update communications areas on the OCTM database. The primary policing function is to monitor the status of communication resources and groups that currently have change requests in progress for them. If the policing function identifies a period of inactivity (since the last COMTC macro was issued for the communication resource or group), it sends a warning message to the RO Cras indicating that further action is required. The policing function also activates the ECB-controlled installation-wide exit AOCM. The AOCM exit program can be used to perform additional functions. For example, it can suppress the transmission of warning messages and can issue a COMTC macro to activate the next COMTC function for the communication resource or group. For a detailed description of the AOCM installation-wide exit, see ALCS Installation and Customization.

The AOCM exit program is activated when there has been a period of inactivity for either a communications resource or a communications group. For example, if a communications group with five active change requests has been loaded (but not confirmed), and 48 hours has elapsed since the COMTC LOAD was issued, the policing exit program is activated. The policing function activates the exit program whenever there has been a period of inactivity for a communications resource or group that is waiting for either the COMTC LOAD, COMTC CONFIRM, COMTC COMMIT or COMTC UNALLOCATE macro to be issued. The exit is initially activated after a 48 hour period of inactivity, but it will be activated again after 56 hours, 64 hours, and every 8 hours after that until the required COMTC macro is issued. After each period of inactivity, the policing function sends a warning message to RO Cras stating which COMTC macro (load, confirm, etc.) is outstanding for the resource or group. The policing exit can suppress the transmission of the warning message, and/or issue the COMTC macro that the resource or group is waiting for. For example, if a COMTC COMMIT had been issued for a communications group, but the group had not yet been unallocated, after 48 hours the policing exit would be activated for this communications group. On this activation, the exit could just return to ALCS and allow the warning message to be sent to RO Cras. If the CEUS has still not unallocated the communications group after 56 hours, the exit will be activated again, and this time the exit could suppress transmission of the warning message and issue a COMTC UNALLOCATE to request OCTM to perform the unallocate function for that group.

Adding and changing communication resources

The ALCS communications generation function provides the COMDFLT macro for defining default values for groups of communication resources. Some default values are suitable for all the resources that belong to a specific communication type (for example, VTAM 3270). Some ALCS installations do not use the IBM defaults provided on the COMDEF macros and have their own set of defaults which they define on COMDFLT. When new resources are being added, the CEUS could display a screen map that included default values for specific fields, defaults which match those provided in the COMDFLT generation macro. When the CEUS is used for adding new communication resources or changing current resources, you may want to restrict the values that can be input in specific fields. You could therefore validate these fields in your CEUS application and reject any value which does not conform to your installation standards.

COMTC ADD and COMTC REPLACE allow alternate CRAS status to be defined for a terminal resource. This can be high CRAS status (AT1 to AT16 and AP1 to AP16) or low CRAS status. The CEUS could inhibit the allocation of CRAS status to those CEUS users who have appropriate CRAS authority. For example, if a CEUS user has high CRAS authority, they could be allowed to allocate any CRAS status, but if they have low CRAS authority they could allocate only low CRAS status.

Activating communications change requests

Communication change requests are activated using the COMTC LOAD, COMTC CONFIRM and COMTC COMMIT macros. A three stage process has been provided so that if incorrect change requests are submitted to the OCTM facility by the CEUS, there is an opportunity to detect this and back out the changes (via COMTC BACKUP). The sample CEUS application has kept these three stages separate, requiring the system administrator to specifically request the load, confirm and then commit functions (via separate options). When a communications group is used, a COMTC UNALLOCATE is also required after the commit, making it a four stage process.

ALCS users may prefer to combine some of these functions together, therefore reducing the number of actions to be taken by the system administrator. For example, the load and confirm functions could be combined, the confirm and commit functions could be combined, the commit and unallocate functions combined, etc. The following provides some suggestions on this.

- Combined load and confirm functions

  The CEUS can request OCTM to load a single change request or a group of change requests. The COMTC LOAD macro is used to load the change request(s) into the online communication table. After a successful load, the CEUS could immediately use the COMTC CONFIRM macro to request OCTM to confirm the change request(s). After a confirm, the change request(s) are retained over an ALCS restart. A back out of the change request(s) is still possible after the confirm. The commit (and unallocate, if required) could be issued a few hours later when the change request(s) have been verified.
- Combined confirm and commit functions

  With this option, the COMTC LOAD is issued on its own. What are the benefits and disadvantages of issuing the COMTC LOAD on its own? The primary benefit is that if a major problem occurred because of the load, it would have no effect on the system during the next ALCS restart. The disadvantage is that if the ALCS system fails, the change request(s) are not re-loaded on ALCS restart and another load request (COMTC LOAD) is required after the restart.
- Combined commit and unallocate functions

  If the load and confirm functions have been combined together (for a communications group) you may wish to also combine the commit and unallocate functions. When a communications group has been committed, there will normally be no further requirement to retain the group, therefore the CEUS could issue the COMTC COMMIT and follow it immediately with a COMTC UNALLOCATE.

Inactivating and Activating Communication Resources

The COMTC LOAD and COMTC BACKUP macros update resources in the online communication table. If the communication resources being loaded or backed out are in the active (logged on) status in the online communication table, then the load (or back out) will fail. The CEUS user could use the ZACOM command to inactivate these resources, but you may prefer to perform the inactivation within the CEUS via the &comss. macro. You may also wish to consider the requirement to activate new or modified resources that have been loaded into the online communication table. The CEUS user could use ZACOM to activate the resources, but you may prefer to perform the activation within the CEUS via the &comss. macro. The following suggests how the CEUS could perform the inactivation and activation of resources.

- Inactivation before a COMTC LOAD

  When communication resources are being replaced or deleted, immediately prior to issuing the COMTC LOAD macro, use the COMIC macro to determine if the resource is active (test field ICESCST after the COMIC). If the resource is active, use the &comss. macro with FIELD=REC1IFR to inactivate (logoff) the resource.
- Inactivation before a COMTC BACKUP

  When communication resources have been added or replaced, immediately prior to issuing the COMTC BACKUP macro, use the COMIC macro to determine if the resource is active (test field ICESCST after the COMIC). If the resource is active, use the &comss. macro with FIELD=REC1IFR to inactivate (logoff) the resource.

- Activation after a COMTC LOAD

  When communication resources are being added or replaced, immediately after a successful return from the COMTC LOAD macro, use the &comss. macro with FIELD=REC1ISR to activate (logon) the resource. If your ALCS system provides a terminal control record (for example, a AAA record) for each terminal, and a new terminal is being added, the CEUS could also perform initialization of the terminal control record for the terminal (or activate a utility program that performs the initialization).

- Activation after a COMTC BACKUP

  When communication resources have been replaced or deleted, immediately after a successful return from the COMTC BACKUP macro, use the &comss. macro with FIELD=REC1ISR to activate (logon) the resource.

ALCS places restrictions on the usage of &comss. when the macro is used to inactivate or activate a resource. The terminal must have CRAS status or the CEUS user must have CRAS authority (any CRAS, AT1 to AT255). If the &comss. macro is to be used for inactivation and activation, then include a COMIC macro in the CEUS input message editor program to verify CRAS status or authority before processing any requests that submit or activate OCTM change requests.

Managing impact of the UCOMCHG Callable Service

ALCS provides a callable service called UCOMCHG that can be used by monitor exits (for example, USRCOM6 monitor exit) to change resource names (CRN's) in the online communication table. When this callable service is used, the CRN is changed only in the online communication table, therefore the resource continues to use the previous CRN in the OCTM database.

If your ALCS system uses the UCOMCHG service, then it should also contain a batch of resources (normally terminal resources) that are for the specific use of UCOMCHG. These may be resources that are activated (logged on to) only after the UCOMCHG service has changed the resource name. Because these resources have a special purpose in your ALCS system, the CEUS could perform specific checks on them. For example, If the CEUS user attempts to change the definition of one of these resources, and the name of the resource has already been changed by the UCOMCHG service, then the CEUS could reject that change. Alternatively, you may wish to enhance the resource name validation in the CEUS prior to issuing the COMTC QUERY, REPLACE and DELETE macros to determine if the resource belongs to the group dedicated to UCOMCHG usage. If it is, the CEUS could reject the action or limit the actions that are allowed.

Appendix L. COMTC Communication Table Update Macro

The COMTC macro allows the Communications End User System (CEUS) to obtain information about communication resources, submit change requests for those resources and to permanently update them in the OCTM database and the online communication table.

The COMTC macro provides twelve different actions that can be requested by the CEUS. This section provides a detailed description of each action that is provided by the COMTC macro.

Note: All fields containing COMTC macro parameters must have the entry storage protect key (key 8).

COMTC QUERY - Query status of communication resource or group

Format

```
[label]  COMTC ACTION=QUERY,{CRN={field|(reg1)}|CRI={field|(reg2)}|GROUP={field|
(reg3)}}
                          ,LEVEL={Dn|(reg4)}
```

label
Any valid assembler label.

CRN={*field*|(*reg1*)}
The Communication Resource Name (CRN) of the communication resource for which status information is required. The CRN must be an alphanumeric string of 1 to 8 characters. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of an 8-byte field that contains the CRN.

(*reg1*)
Register containing the address of an 8-byte field that contains the CRN. Use general registers 0 through 7.

CRI={*field*|(*reg2*)}
The Communication Resource Identifier (CRI) of the communication resource for which status information is required. The CRI must be 6 hexadecimal digits. Specify one of:

field
Assembler label of a 3-byte field that contains the CRI.

(*reg2*)
Register containing the address of a 3-byte field that contains the CRI. Use general registers 0 through 7.

GROUP={*field*|(*reg3*)}
The name of a communications group for which status information is required. It must be an alphanumeric string of 4 to 7 characters, although the first character can not be numeric. If the group name is less than 7 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of a 7-byte field that contains the name of the communications group.

(*reg3*)
Register containing the address of a 7-byte field that contains the name of the communications group. Use general registers 0 through 7.

LEVEL={D*n*|(*reg4*)}
An available storage level in the Entry Control Block (ECB). A 4000-byte storage block (size L3) containing status information on a communication resource or a communications group will be provided at this storage level. Specify one of:

D*n*
Level symbol: D0, D1, and so on up to DF for level 15.

(*reg4*)
Register containing the level value (use LA *reg*,D*n*). Use general registers 0 through 7.

Description

Use the COMTC macro with the ACTION=QUERY option to query the status and obtain information about a communication resource or a communications group.

Use the **CRN**= or **CRI**= parameter to obtain information about a specific communication resource. When the definition of a communication resource is to be changed (via **COMTC REPLACE**), use COMTC QUERY to obtain the current definition for the communication resource. COMTC QUERY can also be used to obtain details of the owning LU for a specific communication resource before changes are applied via **COMTC REPLACE** to the resource itself. ALCS extracts information about the communication resource from the Online Communication Table Maintenance (OCTM) database or from the online communication table.

Alternatively, use the **GROUP**= parameter to obtain information about a communications group that is in use on the OCTM database. ALCS provides status information and a list of the communication resources that belong to the communications group.

When COMTC QUERY is used with the **CRN**= or **CRI**= parameters, a symbolic CRAS address must not be used (for example, you can not use CRN=AT*nnn*). On return from COMTC QUERY, called with the **CRN**= or **CRI**= parameter, a 4000-byte storage block will reside at the ECB storage level specified by the **LEVEL**= parameter. This storage block will contain detailed information about the communication resource. If a communication change request is currently being processed for the resource, details of the change request and its status are also provided. Use the Communications Resource and Group Information DSECT (CT2TM) to reference the contents of this storage block.

On return from COMTC QUERY, called with the **GROUP**= parameter, a 4000-byte storage block will reside at the ECB storage level specified by the **LEVEL**= parameter. This storage block will contain a list of the communication resources that belong to this communications group and its current status on the OCTM database. Use the Communications Resource and Group Information DSECT (CT2TM) to reference the contents of this storage block.

Register use

On return from COMTC QUERY, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC QUERY macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC QUERY macro because of system error - check RO CRAS for a system error dump

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_FCRN
The communication resource name on the **CRN**= parameter is invalid

COMTC_S_NCRN
The communication resource name on the **CRN**= parameter is unknown

COMTC_S_ACRN
The address of the field that the **CRN**= parameter references is invalid

COMTC_S_FCRI
The communication resource identifier on the **CRI**= parameter is invalid

COMTC_S_NCRI
The communication resource identifier on the **CRI**= parameter is unknown

COMTC_S_ACRI
The address of the field that the **CRI**= parameter references is invalid

COMTC_S_FGRP
The communications group name on the **GROUP**= parameter is invalid

COMTC_S_NGRP
The communications group name on the **GROUP**= parameter is unknown

COMTC_S_AGRP
The address of the field that the **GROUP**= parameter references is invalid

COMTC_S_CGRP
Another COMTC macro may currently be attempting to unallocate the communications group defined on the **GROUP**= parameter. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_EGRP
No communication resources are associated with the communications group defined on the **GROUP**= parameter

COMTC QUERY may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC QUERY will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=QUERY to obtain all the information about a specific communication resource in a storage block on ECB storage level 3 (D3).

```
         COMTC ACTION=QUERY,       ISSUE QUERY REQUEST
               CRN=EBW008,         CRN IS IN EBW008
               LEVEL=D3            RETURN COMMS RESOURCE DATA ON LEV 3
```

The following example shows how you could use the COMTC macro with ACTION=QUERY to obtain a list of communication resources (in a storage block on ECB storage level 4) that belong to a specific communications group on the OCTM database.

```
         CT2TM REG=R05             DEFINE BASE FOR CT2TM BLOCK
QUERYREQ EQU   *
         COMTC ACTION=QUERY,       ISSUE QUERY REQUEST
               GROUP=(R06),        GROUP NAME POINTED TO BY REGISTER 6
               LEVEL=D4            RETURN COMMS RESOURCE LIST ON LEV 4
         LR    R14,R15             LOAD RETURN CODE / REASON CODE
         SLL   R14,16              ISOLATE RETURN CODE
         SRL   R14,16              SHIFT TO LOW-ORDER BYTES
         B     *+4(R14)            BRANCH ON RETURN CODE
         B     QRC0                0 - OK
         B     QRC4                4 - RETRY IMMEDIATELY
         B     QRC8                8 - USER ERROR
         B     QRC12               12 - SYSTEM ERROR
QRC4     EQU   *
         DEFRC ,                   WAIT A MOMENT
         B     QUERYREQ            RETRY THE COMTC REQUEST
QRC0     EQU   *
         L     R05,CE1CR4          LOAD BASE ADDRESS OF CT2TM BLOCK
         L     R04,CT2CNT          LOAD COUNT OF RESOURCES IN GROUP
         LTR   R04,R04             IS COUNT ZERO?
         BZ    GRPEMPTY            IF YES - BRANCH
         LR    R03,R05             CT2TM BASE IN REGISTER 3
         A     R03,CT2LST          ADD DISPLACEMENT TO CRN'S
         LA    R02,MSGCRN          POINT TO CRN MSG DISPLAY FIELD
RESRLOOP EQU   *
         MVC   0(8,R02),0(R03)     MOVE CRN TO MSG DISPLAY AREA
         .....
         LA    R02,16(R02)         INCREMENT BASE OF MSG DISPLAY AREA
         LA    R03,9(R03)          INCREMENT BASE OF CRN LIST IN CT2TM
         BCT   R04,RESRLOOP        GO BACK TO TOP OF LOOP
         .....
         .....
GRPEMPTY EQU   *
```

Related information

"CT2TM - Communications Resource and Group Information DSECT" on page 653.

COMTC GROUPS - Obtain status of communications groups

Format

```
[label]  COMTC ACTION=GROUPS,LEVEL={Dn|(reg)}
                        [,SORT={NAME|DATE}]
```

label
Any valid assembler label.

LEVEL={D*n*|(*reg*)}
An available storage level in the Entry Control Block (ECB). A 4000-byte storage block (size L3) containing information about the allocated communications groups will be provided at this storage level. Specify one of:

D*n*
Level symbol: D0, D1, and so on up to DF for level 15.

(*reg*)
Register containing the level value (use LA *reg*,D*n*). Use general registers 0 through 7.

SORT={NAME|DATE}
Specifies how the communications group information should be sorted. Specify one of:

NAME
Sort the communications group information in alphabetic sequence based on group name. This is the default.

DATE
Sort the communications group information in date and time sequence based on the date and time the group was allocated, with the oldest group first.

Description

Use the COMTC macro with the ACTION=GROUPS option to obtain information about all the allocated communications groups.

On return from COMTC GROUPS, a 4000-byte storage block will reside at the ECB storage level specified by the **LEVEL**= parameter. This storage block provides information on a maximum of 100 communications groups. The information includes the group name, the time/date when the group was allocated, the latest action performed for that group, etc. If there are more than 100 allocated groups, L3ST pool records will be chained to this storage block. Each pool record will hold information on a further 100 allocated groups. The number of L3ST pool records forward chained from this storage block will depend on the number of groups currently allocated. Use the Communications Groups Information DSECT (CT3TM) to reference the contents of this storage block and the overflow pool records. You should ensure that the L3ST pool records are released before the entry terminates processing.

Register use

On return from COMTC GROUPS, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC GROUPS macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC GROUPS macro because of system error - check RO CRAS for a system error dump

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_EGRS
No communications groups exist

COMTC GROUPS may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC GROUPS will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=GROUPS to obtain information about all the communications groups in a storage block on ECB storage level 3 (D3).

```
         CT3TM REG=R07             DEFINE BASE FOR CT3TM BLOCK
GRPQUERY EQU   *
         COMTC ACTION=GROUPS,      ISSUE GROUPS REQUEST
               LEVEL=D3,           RETURN COMMS GROUP DATA ON LEV 3
               SORT=DATE           SORT GROUPS INTO DATE SEQUENCE
         LR    R14,R15             LOAD RETURN CODE / REASON CODE
         SLL   R14,16              ISOLATE RETURN CODE
         SRL   R14,16              SHIFT TO LOW-ORDER BYTES
         B     *+4(R14)            BRANCH ON RETURN CODE
         B     GRC0                0  - OK
         B     GRC4                4  - RETRY IMMEDIATELY
         B     GRC8                8  - RETRY LATER
         B     GRC12               12 - USER ERROR
         B     GRC16               16 - SYSTEM ERROR
GRC4     EQU   *
         DEFRC ,                   WAIT A MOMENT
         B     GRPQUERY            RETRY THE COMTC REQUEST
GRC0     EQU   *
         L     R07,CE1CR3          LOAD BASE ADDRESS OF CT3TM
         L     R03,CT3CNT          LOAD COUNT OF GROUPS IN BLOCK
         LTR   R03,R03             IS COUNT ZERO?
         BZ    NOGROUP             IF YES - BRANCH
ITEMLOOP EQU   *
         TM    CT3STA,CT3CHNG      CHANGE REQUESTS NOT YET LOADED?
         BO    CHANGE              IF YES - BRANCH
         TM    CT3STA,CT3LOAD      CHANGE REQUESTS LOADED?
         BO    LOADED              IF YES - BRANCH
         TM    CT3PRG,CT3LDPR      IS LOAD IN PROGRESS?
         BO    LOADED              IF YES - BRANCH
         ...
         ...
BOTLOOP  EQU   *
         LA    R06,L'CT3ITM        LOAD LENGTH OF ITEM
         AR    R07,R06             INCREMENT TO NEXT ITEM
         BCT   R03,ITEMLOOP        GO BACK TO TOP OF LOOP
         B     FCHCHECK            GO CHECK FOR FORWARD CHAIN
```

Related information

"CT3TM - Communications Groups Information DSECT" on page 657.

COMTC ALLOCATE - Allocate a new communications group

Format

```
[label]  COMTC ACTION=ALLOCATE,GROUP={field|(reg1)}
```

label
Any valid assembler label.

GROUP={*field*|(*reg1*)}
The name of a new communications group that is to be allocated on the Online Communication Table Maintenance (OCTM) database. The name of the communications group must be an alphanumeric string of 4 to 7 characters, although the first character can not be numeric. If the group name is less than 7 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of a 7-byte field that contains the name of the new communications group.

(*reg1*)
Register containing the address of a 7-byte field that contains the name of the new communications group. Use general registers 0 through 7.

Description

Use the COMTC macro with the ACTION=ALLOCATE option to allocate the name of a new communications group.

The COMTC macro can be used to submit a wide range of communications change requests to ALCS. For example, you can modify a current communication resource using **COMTC REPLACE**, you can add new communication resources using **COMTC ADD**, and you can delete obsolete communication resources using **COMTC DELETE**. Use COMTC with ACTION=ALLOCATE to assign a communications group name to a batch of communication change requests that will be submitted to ALCS via the COMTC macro. When you have submitted all the communication change requests for your communications group, you can use a single COMTC macro to *load* all those change requests (using the communications group name in the **COMTC LOAD** macro). You can subsequently use COMTC macros to *confirm* and *commit* the change requests for that communications group (or use COMTC to *backout* the change requests).

Register use

On return from COMTC ALLOCATE, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC ALLOCATE macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC ALLOCATE macro because of system error - check RO CRAS for a system error dump

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_FGRP
The communications group name on the **GROUP**= parameter is invalid

COMTC_S_AGRP
The address of the field that the **GROUP**= parameter references is invalid

COMTC_S_CGRP
Another COMTC macro may currently be attempting to unallocate the communications group defined on the **GROUP**= parameter. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_TGRP
The maximum number of permitted communications groups have already been allocated. Use COMTC GROUPS to obtain a list of the communications groups and unallocate those groups that have been *committed*.

COMTC_S_DGRP
The communications group name that the **GROUP**= parameter references is already in use

COMTC ALLOCATE may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC ALLOCATE will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=ALLOCATE to allocate the name of a new communications group.

```
         MVC   EBW016(7),=C'GRPFIVE' LOAD NAME OF COMMS GROUP
         COMTC ACTION=ALLOCATE,      ISSUE ALLOCATE REQUEST
               GROUP=EBW016          GROUP NAME IS IN EBW016
```

Related information

"COMTC UNALLOCATE - Unallocate a communications group" on page 627.

COMTC ADD - Add a new communication resource

Format

```
[label]  COMTC ACTION=ADD,CRN={field|(reg1)}
                          ,{CRI={field|(reg2)}|CRIBASE={field|(reg3)}}
                          ,{ORD={field|(reg4)}|ORDBASE={field|(reg5)}}
                          ,COMSDAT={field|(reg6)}
                          ,LEVEL={Dn|(reg7)}
                          [,GROUP={field|(reg8)}]
```

label
Any valid assembler label.

CRN={*field*|(*reg1*)}
The Communication Resource Name (CRN) of the new communication resource. The CRN must be an alphanumeric string of 1 to 8 characters, it must be unique within ALCS and the first character must be alphabetic. If the CRN is less than 8 characters, it should be left justified and padded with spaces. For VTAM 3270 terminal resources and X.25 PVC resources, the CRN must be the LU name as defined to VTAM.

The CRN for an X.25 ALC terminal resource is comprised of a base CRN plus one or two other elements depending on the type of X.25 PVC that the X.25 ALC terminal is associated with. There are

three types of X.25 PVC that can have associated X.25 ALC terminals. The length of the base CRN for these terminals is dependent on the type of X.25 PVC that they are associated with.
For an X.25 ALC terminal that is associated with a ***Type 1*** X.25 PVC, the base CRN must be an alphanumeric string of 1 to 6 characters. You must append a two-character terminal address to the 6-character base CRN to create the full CRN.
For an X.25 ALC terminal that is associated with a ***Type 6*** or ***Type 7*** X.25 PVC, the base CRN must be an alphanumeric string of 1 to 4 characters. You must append a two-character interchange address followed by a two-character terminal address to the 4-character base CRN to create the full CRN.
The X.25 ALC terminal and interchange addresses that are used to create the full CRN must also be supplied in the `Communication Resource Definition DSECT, CT1TM`. The terminal address must be provided in field `CT1CTA` and the interchange address must be provided in field `CT1CIA`. Use the **COMSDAT**= parameter to specify the storage area that contains the `CT1TM` DSECT.
Specify one of:

field
Assembler label of an 8-byte field that contains the CRN of the new communication resource.

(*reg1*)
Register containing the address of an 8-byte field that contains the CRN of the new communication resource. Use general registers 0 through 7.

CRI={*field*|(*reg2*)}
The Communication Resource Identifier (CRI) of the new communication resource. The CRI address must be 6 hexadecimal digits and it must be unique within ALCS. Specify one of:

field
Assembler label of a 3-byte field that contains the CRI address.

(*reg2*)
Register containing the address of a 3-byte field that contains the CRI address. Use general registers 0 through 7.

CRIBASE={*field*|(*reg3*)}
Base Communication Resource Identifier. ALCS will select an available CRI for the new communication resource. Specify one of:

field
Assembler label of a 3-byte field that contains the base CRI address. It must be 6 hexadecimal digits and must be in the range `020000` to FFFFFE. The first available CRI whose address is equal to or higher than the base CRI is allocated to the new communication resource.

(*reg3*)
Register containing the address of a 3-byte field that contains the base CRI address. Use general registers 0 through 7. It must be 6 hexadecimal digits and must be in the range `020000` to FFFFFE. The first available CRI whose address is higher than the base CRI is allocated by ALCS to the new communication resource.

ORD={*field*|(*reg4*)}
The communication resource ordinal number for the new communication resource. The resource ordinal number is a hexadecimal number. Specify one of:

field
Assembler label of a fullword field that contains the resource ordinal number in hexadecimal.

(*reg4*)
Register containing the address of a fullword field that contains the resource ordinal number in hexadecimal. Use general registers 0 through 7.

ORDBASE={*field*|(*reg5*)}
Base communication resource ordinal number for the new communication resource. The base resource ordinal number is a hexadecimal number. ALCS will select an available resource ordinal number for the new communication resource. Specify one of:

field
Assembler label of a fullword field that contains the base resource ordinal number in hexadecimal. The first available ordinal number that is equal to or higher than the base resource ordinal number is allocated by ALCS to the new communication resource.

(*reg5*)
Register containing the address of a fullword field that contains the base resource ordinal number in hexadecimal. Use general registers 0 through 7. The first available ordinal number that is higher than the base resource ordinal number is allocated by ALCS to the new communication resource.

COMSDAT={*field*|(*reg6*)}
The communications data that defines the characteristics of the new communication resource. This data must be provided in a storage area that has been formatted using the `Communication Resource Definition DSECT, CT1TM`. Specify one of:

field
Assembler label of a storage area that contains the communications data formatted by the `CT1TM` DSECT.

(*reg6*)
Register containing the address of a storage area that contains the communications data formatted by the `CT1TM` DSECT. Use general registers 0 through 7.

LEVEL={D*n*|(*reg7*)}
An available storage level in the Entry Control Block (ECB). An L3 size storage block is attached by ALCS to this storage level for recording any errors that are detected in the communication resource data that is provided (via the **COMSDAT**= parameter) for this new communications resource. The storage block can hold a maximum of 49 error messages. Specify one of:

D*n*
Level symbol: `D0, D1`, and so on up to `DF` for level `15`.

(*reg7*)
Register containing the level value (use `LA` *reg*`,D`*n*). Use general registers 0 through 7.

GROUP={*field*|(*reg8*)}
The name of the communications group that this new communication resource belongs to. It must be an alphanumeric string of 4 to 7 characters, although the first character can not be numeric. If the group name is less than 7 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of a 7-byte field that contains the name of the communications group.

(*reg8*)
Register containing the address of a 7-byte field that contains the name of the communications group. Use general registers 0 through 7.

Description

Use the `COMTC` macro with the `ACTION=ADD` option to add a new communication resource to the Online Communication Table Maintenance (OCTM) database. A single `COMTC ADD` should be used to provide all the details of the new communication resource. The new communication resource can be a terminal (except Prime and RO CRAS) or an X.25 PVC (types 1, 6 or 7).

Use the **CRN**= parameter to specify the Communications Resource Name that will be used to identify this new communication resource.

Use the **CRI**= parameter to specify the Communications Resource Identifier that will be used to identify this new communication resource. Alternatively, use the **CRIBASE**= parameter to request ALCS to select an available CRI for this new communication resource.

Use the **ORD**= parameter to specify the communication resource ordinal number for this new communication resource. Alternatively, use the **ORDBASE**= parameter to request ALCS to select an available communication resource ordinal number for this new communication resource.

Use the **COMSDAT**= parameter to point to a storage area containing detailed information about the new communication resource. The Communication Resource Definition DSECT (CT1TM) defines the layout of this storage area and the individual fields within it. Initialize this storage area to hexadecimal zeros and then place the detailed information about the new communication resource in the relevant fields. ALCS systems that use the communications user data area can provide the required user data for the new communication resource in the CT1TM DSECT. The description of the CT1TM DSECT provides guidance on the fields that require data for each type of communication resource.

On return from COMTC ADD, an L3 size storage block resides at the ECB storage level specified by the **LEVEL**= parameter. During validation of the communication resource data provided in the CT1TM DSECT, if errors are detected, appropriate error messages are placed in this storage block. Test the return code in register 15 to determine if any errors were found. The storage block will contain a maximum of 49 error messages. If more than 49 errors are found, ALCS will report only the first 49. Each error message will be a maximum of 80 bytes, therefore the storage block is formatted into 80-byte items. A 4-byte header resides at the beginning of the block containing a count of the error messages that have been placed in the block. Immediately following the header is the first 80-byte error message item. The error message text is in EBCDIC and the message contains a prefix DXC*nnnn*, where the message number *nnnn* is in the range 9100 to 9139. Details of these error messages can be found in ALCS Messages and Codes.

When a batch of new communication resources are being added, allocate a ***communications group name*** to that batch. The **COMTC ALLOCATE** macro allows you to do this. Although a separate COMTC ADD is required for adding each new communication resource, when you use a communications group to add a batch of new resources, it enables you to use a single **COMTC LOAD** macro for loading the batch of new resources on to the ALCS online communication table. Use the **GROUP**= parameter on COMTC ADD to identify the name of the communications group that the new communication resource belongs to.

X.25 PVC and ALC resources

ALCS supports different types of X.25 PVC communication resources, including those for Type-A traffic, Type-B traffic and host-to-host traffic. Use COMTC ADD for adding X.25 PVC communication resources for Type-A traffic. Other types of X.25 PVC resources must be defined to ALCS in the communication generation COMDEF generation macros.

An X.25 ALC terminal resource will always be associated with an X.25 PVC resource that manages Type-A traffic. The X.25 PVC resource must therefore always be defined to ALCS via a COMTC ADD before its associated X.25 ALC terminals. An X.25 PVC and its associated ALC terminals could be added in a single X.25 *communications group*. Use **COMTC ALLOCATE** to allocate a communications group name for the X.25 PVC and its associated X.25 ALC terminals. Use COMTC ADD to add each individual X.25 resource, and then issue COMTC macros to *load, confirm* and *commit* this X.25 communications group.

Register use

On return from COMTC ADD, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC ADD macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC ADD macro because of system error - check RO CRAS for a system error dump

COMTC_R_DATAERR
The communications data provided in the CT1TM DSECT contained errors. A storage block at the ECB storage level specified by the **LEVEL**= parameter contains error messages describing up to 49 different errors in the communications data.

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_FCRN
The communication resource name on the **CRN**= parameter is invalid

COMTC_S_ACRN
The address of the field that the **CRN**= parameter references is invalid

COMTC_S_FCRI
The communication resource identifier on the **CRI**= parameter is invalid

COMTC_S_ACRI
The address of the field that the **CRI**= or **CRIBASE**= parameter references is invalid

COMTC_S_FGRP
The communications group name on the **GROUP**= parameter is invalid

COMTC_S_NGRP
The communications group name on the **GROUP**= parameter is unknown

COMTC_S_AGRP
The address of the field that the **GROUP**= parameter references is invalid

COMTC_S_CGRP
Another COMTC macro may currently be attempting to unallocate the communications group defined on the **GROUP**= parameter. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_IPGS
A previous COMTC macro for this communication resource or group failed to complete. Use COMTC QUERY to determine the status of the communication resource or group.

COMTC_S_ESEQ
There are change requests for this communication resource or group that are in the *confirmed* status

COMTC_S_TGRP
The maximum number of permitted non-group change requests have already been submitted. Identify non-group change requests that are waiting to be committed and use COMTC COMMIT to commit them.

COMTC_S_TCHG
The maximum number of permitted change requests have already been submitted for the communications group defined in the **GROUP**= parameter

COMTC_S_ADAT
The address of the field that the **COMSDAT**= parameter references is invalid

COMTC_S_DCRN
The communication resource name on the **CRN**= parameter is already in use

COMTC_S_DCRI
The communication resource identifier on the **CRI**= parameter is already in use

COMTC_S_DORD
The communication resource ordinal number that the **ORD**= parameter references is already in use

COMTC_S_AORD
The address of the field that the **ORD**= or **ORDBASE**= parameter references is invalid

COMTC_S_RCRI
The communication resource identifier on the **CRI**= parameter is within the range of CRIs that are for the exclusive use of the ALCS communications generation (as defined in the **COMGEN CRIRANGE**= parameter)

COMTC_S_ICRI
The **CRIBASE**= parameter was used to obtain the next available CRI, but there were no available CRI addresses found between this base CRI address and the maximum CRI address

COMTC_S_IORD
The communication resource ordinal number is invalid.
If a base ordinal number has been provided in the **ORDBASE**= parameter, then no available ordinal numbers have been found between this base ordinal number and the maximum ordinal number.
If an ordinal number has been provided in the **ORD**= parameter, then that ordinal number is higher than the maximum ordinal number specified in the communications generation (as defined in the **COMGEN MAXORD**= parameter).

COMTC_S_RORD
The communication resource ordinal number that the **ORD**= parameter references is within the range of ordinals that are for the exclusive use of the ALCS communications generation (as defined in the **COMGEN ORDRANGE**= parameter)

COMTC_S_FORD
The communication resource ordinal number that the **ORD**= parameter references is invalid because it contains hexadecimal zeros.

When the return code COMTC_R_DATAERR is reported in general register 15, ALCS will have detected errors in the communication data provided in the CT1TM DSECT. To obtain a list of the errors that have been detected, examine the contents of the storage block that resides at the data level specified by the **LEVEL**= parameter. The storage block contains error messages describing the error conditions that have been detected. There can be a maximum of 49 error messages in the storage block. The previous section provides information on the format of this storage block.

COMTC ADD may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC ADD will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=ADD to define a new communication resource that is being added as part of a communications group.

```
        COMTC ACTION=ADD,         ISSUE ADD REQUEST
              CRN=EBW008,         CRN IS IN EBW008
              CRIBASE=EBW016,     SELECT AN AVAILABLE CRI
              ORDBASE=EBW020,     SELECT AN AVAILABLE ORDINAL
              COMSDAT=EBX000,     COMMS DATA IN EBX000
              LEVEL=D4,           ECB LEVEL FOR ERROR MSG BLOCK
              GROUP=EBW024        GROUP NAME IN EBW024
```

Related information

"CT1TM - Communication Resource Definition DSECT" on page 631.

COMTC REPLACE - Change a communication resource

Format

```
[label]  COMTC ACTION=REPLACE,{CRN={field|(reg1)}|CRI={field|(reg2)}}
                             ,COMSDAT={field|(reg3)}
                             ,LEVEL={Dn|(reg4)}
                             [,GROUP={field|(reg5)}]
```

label
Any valid assembler label.

CRN={*field*|(*reg1*)}
The Communication Resource Name (CRN) of the communication resource that is being changed. The CRN must be an alphanumeric string of 1 to 8 characters. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of an 8-byte field that contains the CRN.

(*reg1*)
Register containing the address of an 8-byte field that contains the CRN. Use general registers 0 through 7.

CRI={*field*|(*reg2*)}
The Communication Resource Identifier (CRI) of the communication resource that is being changed. The CRI address must be 6 hexadecimal digits. Specify one of:

field
Assembler label of a 3-byte field that contains the CRI address.

(*reg2*)
Register containing the address of a 3-byte field that contains the CRI address. Use general registers 0 through 7.

COMSDAT={*field*|(*reg3*)}
A storage area that contains the changes to be applied to the communication resource. This storage area must be formatted using the `Communication Resource Definition DSECT,CT1TM`. Specify one of:

field
Assembler label of a storage area formatted by the `CT1TM` DSECT.

(*reg3*)
Register containing the address of a storage area formatted by the `CT1TM` DSECT. Use general registers 0 through 7.

LEVEL={D*n*|(*reg4*)}
An available storage level in the Entry Control Block (ECB). An L3 size storage block is attached by ALCS to this storage level for recording any errors that are detected in the communication resource data that is provided (via the **COMSDAT**= parameter) for this communications resource. The storage block can hold a maximum of 49 error messages. Specify one of:

D*n*
Level symbol: `D0, D1`, and so on up to `DF` for level `15`.

(*reg4*)
Register containing the level value (use LA *reg*,D*n*). Use general registers 0 through 7.

GROUP={*field*|(*reg5*)}
The name of the communications group that this communication resource belongs to. It must be an alphanumeric string of 4 to 7 characters, although the first character can not be numeric. If the group name is less than 7 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of a 7-byte field that contains the name of the communications group.

(*reg5*)
Register containing the address of a 7-byte field that contains the name of the communications group. Use general registers 0 through 7.

Description

Use the COMTC macro with the ACTION=REPLACE option to replace the definition of a communication resource in the Online Communication Table Maintenance (OCTM) database. The definition for the communication resource must be provided in a storage area that is formatted by the Communication Resource Definition DSECT (CT1TM). Use the COMTC macro with the ACTION=QUERY option to obtain the current definition for the resource in a CT1TM DSECT. Apply the required changes for the communications resource to the appropriate fields within the CT1TM DSECT and issue the COMTC REPLACE macro.

Use the **CRN**= or **CRI**= parameter to identify the specific communication resource that is being changed. ALCS verifies that the communication resource being changed currently exists on the OCTM database, and if it does, alters its status to *changed*.

Use the **COMSDAT**= parameter to point to the storage area containing the modified CT1TM DSECT.

On return from COMTC REPLACE, an L3 size storage block resides at the ECB storage level specified by the **LEVEL**= parameter. During validation of each change to the communication resource, if errors are detected, appropriate error messages are placed in this storage block. Test the return code in register 15 to determine if any errors were found. The storage block will contain a maximum of 49 error messages. If more than 49 errors are found, ALCS will report only the first 49. Each error message will be a maximum of 80 bytes, therefore the storage block is formatted into 80-byte items. A 4-byte header resides at the beginning of the block containing a count of the error messages that have been placed in the block. Immediately following the header is the first 80-byte error message item. The error message text is in EBCDIC and the message contains a prefix DXC*nnnn*, where the message number *nnnn* is in the range 9100 to 9139. Details of these error messages can be found in ALCS Messages and Codes.

When a batch of communication resources are being changed, allocate a ***communications group name*** to that batch. The **COMTC ALLOCATE** macro allows you to do this. Although a separate COMTC REPLACE is required for each communication resource being changed, when you use a communications group for a batch of resources, it enables you to use a single **COMTC LOAD** macro for loading all the changes on to the ALCS online communication table. Use the **GROUP**= parameter on COMTC REPLACE to identify the name of the communications group that the communication resource belongs to.

Register use

On return from COMTC REPLACE, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC REPLACE macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC REPLACE macro because of system error - check RO CRAS for a system error dump

COMTC_R_DATAERR
The communications data provided in the CT1TM DSECT contained errors. A storage block at the ECB storage level specified by the **LEVEL**= parameter contains error messages describing up to 49 different errors in the communications data.

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_FCRN
The communication resource name on the **CRN**= parameter is invalid

COMTC_S_NCRN
The communication resource name on the **CRN**= parameter is unknown

COMTC_S_ACRN
The address of the field that the **CRN**= parameter references is invalid

COMTC_S_FCRI
The communication resource identifier on the **CRI**= parameter is invalid

COMTC_S_NCRI
The communication resource identifier on the **CRI**= parameter is unknown

COMTC_S_ACRI
The address of the field that the **CRI**= parameter references is invalid

COMTC_S_FGRP
The communications group name on the **GROUP**= parameter is invalid

COMTC_S_NGRP
The communications group name on the **GROUP**= parameter is unknown

COMTC_S_AGRP
The address of the field that the **GROUP**= parameter references is invalid

COMTC_S_CGRP
Another COMTC macro may currently be attempting to unallocate the communications group defined on the **GROUP**= parameter. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_IPGS
A previous COMTC macro for this communication resource or group failed to complete. Use COMTC QUERY to determine the status of the communication resource or group.

COMTC_S_WGRP
The **GROUP**= parameter was incorrectly specified or contained the wrong group name. Other change requests have already been submitted for this communication resource, but those change requests do not belong to a communications group or belong to a different group.

COMTC_S_WCRX
A change request belonging to a communications group already exists for this communication resource, but the **GROUP**= parameter is missing from this COMTC macro

COMTC_S_ESEQ
There are change requests for this communication resource or group that are in the *confirmed* status

COMTC_S_TGRP
The maximum number of permitted non-group change requests have already been submitted. Identify non-group change requests that are waiting to be committed and use COMTC COMMIT to commit them.

COMTC_S_IDLR
The communication resource can not be changed because it is currently being deleted by a COMTC DELETE

COMTC_S_TCHG
The maximum number of permitted change requests have already been submitted for the communications group defined in the **GROUP**= parameter

COMTC_S_ADAT
The address of the field that the **COMSDAT**= parameter references is invalid

COMTC_S_ILDT
The communication resource type in the CT1TM DSECT for this communications resource does not match the communication resource type in the online communications table. An incorrect resource type may have been defined in the CT1TM DSECT for this resource. Alternatively, the wrong CRN or CRI may have been given in the COMTC macro parameters.

COMTC_S_ILDX
Modification of the PVC type for this X.25 PVC communication resource in the CT1TM DSECT is not allowed.

COMTC_S_ILDY
Modification of the interchange address, terminal address or owning PVC name in the CT1TM DSECT for this X.25 ALC terminal resource is not allowed.

COMTC_S_IUPD
ALCS can not change the X.25 ALC terminal resource because of a conflict with the ordinal numbers of the OCTM records that manage the X.25 ALC resource and its owning X.25 PVC. The X.25 ALC resource must be deleted and added again using the COMTC DELETE and COMTC ADD macros.

Alternatively, this reason code may indicate that modification of the associated device for this terminal (in the CT1TM DSECT) can not be performed because of a conflict with the ordinal number of the OCTM record that manages the associated device. This terminal resource must be deleted and added again using the COMTC DELETE and COMTC ADD macros.

When the return code COMTC_R_DATAERR is reported in general register 15, ALCS will have detected errors in the communication data provided in the CT1TM DSECT. To obtain a list of the errors that have been detected, examine the contents of the storage block that resides at the data level specified by the **LEVEL**= parameter. The storage block contains error messages describing the error conditions that have been detected. There can be a maximum of 49 error messages in the storage block. The previous section provides information on the format of this storage block.

COMTC REPLACE may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC REPLACE will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=REPLACE to change a communication resource.

```
        COMTC ACTION=REPLACE,     ISSUE REPLACE REQUEST
              CRN=EBW008,         CRN IS IN EBW008
              COMSDAT=EBX000,     COMMS DATA IN EBX000
              LEVEL=D3            ECB LEVEL FOR ERROR MSG BLOCK
```

Related information

"CT1TM - Communication Resource Definition DSECT" on page 631.

COMTC DELETE - Delete a communication resource

Format

```
[label]  COMTC ACTION=DELETE,{CRN={field|(reg1)}|CRI={field|(reg2)}}
                          [,GROUP={field|(reg3)}]
```

label
Any valid assembler label.

CRN={*field*|(*reg1*)}
The Communication Resource Name (CRN) of the communication resource that is being deleted. The CRN must be an alphanumeric string of 1 to 8 characters. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of an 8-byte field that contains the CRN.

(*reg1*)
Register containing the address of an 8-byte field that contains the CRN. Use general registers 0 through 7.

CRI={*field*|(*reg2*)}
The Communication Resource Identifier (CRI) of the communication resource that is being deleted. The CRI must be 6 hexadecimal digits. Specify one of:

field
Assembler label of a 3-byte field that contains the CRI.

(*reg2*)
Register containing the address of a 3-byte field that contains the CRI. Use general registers 0 through 7.

GROUP={*field*|(*reg3*)}
The name of the communications group that this deletion request belongs to. It must be an alphanumeric string of 4 to 7 characters, although the first character can not be numeric. If the group name is less than 7 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of a 7-byte field that contains the name of the communications group.

(*reg3*)
Register containing the address of a 7-byte field that contains the name of the communications group. Use general registers 0 through 7.

Description

Use the COMTC macro with the ACTION=DELETE option to delete a communication resource that is defined on the Online Communication Table Maintenance (OCTM) database.

Use the **CRN**= or **CRI**= parameter to identify the specific communication resource that is being deleted. ALCS verifies that the communication resource being deleted currently exists on the OCTM database and if it does, marks it for deletion. If a communications resource has a change request (**COMTC ADD** or **COMTC REPLACE**) currently in progress for it, the communication resource can not be deleted. Use the COMTC macro with the ACTION=CANCEL option to cancel the in-progress change request and then use COMTC DELETE to delete the communication resource.

When a batch of communication resources are being deleted, allocate a ***communications group name*** to that batch. The **COMTC ALLOCATE** macro allows you to do this. Although a separate COMTC DELETE is required for deleting each individual communication resource, when you use a communications group, it

enables you to use a single **COMTC LOAD** macro to delete the batch of communication resources from the ALCS online communication table. Use the **GROUP**= parameter on COMTC DELETE to identify the name of the communications group that the deletion request belongs to.

Register use

On return from COMTC DELETE, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC DELETE macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC DELETE macro because of system error - check RO CRAS for a system error dump

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_FCRN
The communication resource name on the **CRN**= parameter is invalid

COMTC_S_NCRN
The communication resource name on the **CRN**= parameter is unknown

COMTC_S_ACRN
The address of the field that the **CRN**= parameter references is invalid

COMTC_S_FCRI
The communication resource identifier on the **CRI**= parameter is invalid

COMTC_S_NCRI
The communication resource identifier on the **CRI**= parameter is unknown

COMTC_S_ACRI
The address of the field that the **CRI**= parameter references is invalid

COMTC_S_FGRP
The communications group name on the **GROUP**= parameter is invalid

COMTC_S_NGRP
The communications group name on the **GROUP**= parameter is unknown

COMTC_S_AGRP
The address of the field that the **GROUP**= parameter references is invalid

COMTC_S_CGRP
Another COMTC macro may currently be attempting to unallocate the communications group defined on the **GROUP**= parameter. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_IPGS
A previous COMTC macro for this communication resource or group failed to complete. Use COMTC QUERY to determine the status of the communication resource or group.

COMTC_S_ESEQ
There are change requests for this communication resource or group that are in the *confirmed* status

COMTC_S_TGRP
The maximum number of permitted non-group change requests have already been submitted. Identify non-group change requests that are waiting to be committed and use COMTC COMMIT to commit them.

COMTC_S_TCHG
The maximum number of permitted change requests have already been submitted for the communications group defined in the **GROUP**= parameter

COMTC_S_IDEL
Another change request has already been submitted for this communication resource (a COMTC ADD, a COMTC REPLACE or a COMTC DELETE)

COMTC DELETE may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC DELETE will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=DELETE to delete a communication resource.

```
         COMTC ACTION=DELETE,      ISSUE DELETE REQUEST
               CRN=EBW048          CRN IS IN EBW048
```

Related information

None.

COMTC CANCEL - Cancel a communication change request

Format

```
[label]  COMTC ACTION=CANCEL,{CRN={field|(reg1)}|CRI={field|(reg2)}}
                            [,GROUP={field|(reg3)}]
```

label
Any valid assembler label.

CRN={*field*|(*reg1*)}
The Communication Resource Name (CRN) of the communication resource whose change request on the Online Communication Table Maintenance (OCTM) database is being cancelled. The CRN must be an alphanumeric string of 1 to 8 characters. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of an 8-byte field that contains the CRN.

(*reg1*)
Register containing the address of an 8-byte field that contains the CRN. Use general registers 0 through 7.

CRI={*field*|(*reg2*)}
The Communication Resource Identifier (CRI) of the communication resource whose change request on the OCTM database is being cancelled. The CRI must be 6 hexadecimal digits. Specify one of:

field
Assembler label of a 3-byte field that contains the CRI.

(*reg2*)
Register containing the address of a 3-byte field that contains the CRI. Use general registers 0 through 7.

GROUP={*field*|(*reg3*)}
The name of the communications group that the change request being cancelled belongs to. It must be an alphanumeric string of 4 to 7 characters, although the first character can not be numeric. If the group name is less than 7 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of a 7-byte field that contains the name of the communications group.

(*reg3*)
Register containing the address of a 7-byte field that contains the name of the communications group. Use general registers 0 through 7.

Description

Use the COMTC macro with the ACTION=CANCEL option to cancel a communication change request that had been requested by **COMTC ADD**, **COMTC REPLACE** or by **COMTC DELETE**. If the communication change request belongs to a communications group, identify the name of that group in the **GROUP**= parameter.

ALCS verifies that the communication change request being cancelled currently exists on the OCTM database, that it belongs to the communications group that has been specified (in the **GROUP**= parameter) and that it does not have the *loaded*, *confirmed* or *committed* status. ALCS removes the change request from the OCTM database.

Register use

On return from COMTC CANCEL, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC CANCEL macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC CANCEL macro because of system error - check RO CRAS for a system error dump

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_FCRN
The communication resource name on the **CRN**= parameter is invalid

COMTC_S_NCRN
The communication resource name on the **CRN**= parameter is unknown

COMTC_S_ACRN
The address of the field that the **CRN**= parameter references is invalid

COMTC_S_FCRI
The communication resource identifier on the **CRI**= parameter is invalid

COMTC_S_NCRI
The communication resource identifier on the **CRI**= parameter is unknown

COMTC_S_ACRI
The address of the field that the **CRI**= parameter references is invalid

COMTC_S_FGRP
The communications group name on the **GROUP**= parameter is invalid

COMTC_S_NGRP
The communications group name on the **GROUP**= parameter is unknown

COMTC_S_AGRP
The address of the field that the **GROUP**= parameter references is invalid

COMTC_S_CGRP
Another COMTC macro may currently be attempting to unallocate the communications group defined on the **GROUP**= parameter. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_IPGS
A previous COMTC macro for this communication resource or group failed to complete. Use COMTC QUERY to determine the status of the communication resource or group.

COMTC_S_WGRP
The **GROUP**= parameter was incorrectly specified or contained the wrong group name. The change request being cancelled does not belong to a communications group or belongs to a different group.

COMTC_S_WCRX
The change request being cancelled belongs to a communications group, but the **GROUP**= parameter is missing from this COMTC macro

COMTC_S_ESEQ
The change request for this communication resource is in the *loaded* or *confirmed* status

COMTC_S_NCRQ
There are no change requests for this communication resource or the change request has already been cancelled

COMTC CANCEL may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC CANCEL will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=CANCEL to cancel a change request that had been previously submitted via COMTC REPLACE for a communication resource.

```
        COMTC ACTION=CANCEL,      ISSUE CANCEL REQUEST
              CRN=EBW008          CRN IS IN EBW008
```

Related information

“COMTC ADD - Add a new communication resource” on page 601.
“COMTC REPLACE - Change a communication resource” on page 607.
“COMTC DELETE - Delete a communication resource” on page 611.

COMTC LOAD - Load communication change requests

Format

```
[label]  COMTC ACTION=LOAD,{CRN={field|(reg1)}|CRI={field|(reg2)}|GROUP={field|
(reg3)}}
                          ,LEVEL={Dn|(reg4)}
```

label
Any valid assembler label.

CRN={*field*|(*reg1*)}
The Communication Resource Name (CRN) of the communication resource whose change request is being loaded. The CRN must be an alphanumeric string of 1 to 8 characters. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of an 8-byte field that contains the CRN.

(*reg1*)
Register containing the address of an 8-byte field that contains the CRN. Use general registers 0 through 7.

CRI={*field*|(*reg2*)}
The Communication Resource Identifier (CRI) of the communication resource whose change request is being loaded. The CRI must be 6 hexadecimal digits. Specify one of:

field
Assembler label of a 3-byte field that contains the CRI.

(*reg2*)
Register containing the address of a 3-byte field that contains the CRI. Use general registers 0 through 7.

GROUP={*field*|(*reg3*)}
The name of the communications group whose change requests are being loaded. It must be an alphanumeric string of 4 to 7 characters, although the first character can not be numeric. If the group name is less than 7 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of a 7-byte field that contains the name of the communications group.

(*reg3*)
Register containing the address of a 7-byte field that contains the name of the communications group. Use general registers 0 through 7.

LEVEL={D*n*|(*reg4*)}
An available storage level in the Entry Control Block (ECB). An L3 size storage block is attached by ALCS to this storage level for recording errors that occur when the communication change requests

are loaded in the online communication table. The storage block contains up to 49 error messages (with each message giving the name of the resource and a description of the load error). Specify one of:

D*n*
Level symbol: D0, D1, and so on up to DF for level 15.

(*reg4*)
Register containing the level value (use LA *reg*,D*n*). Use general registers 0 through 7.

Description

Use the COMTC macro with the ACTION=LOAD option to load communication change requests that have been requested by **COMTC ADD**, **COMTC REPLACE** or by **COMTC DELETE** macros. Use the **CRN**= or **CRI**= parameter to load a change request for a specific communication resource. Use the **GROUP**= parameter to load a group of change requests for a communications group. When COMTC LOAD is used for a communications group, all the change requests that have been submitted via COMTC macros for individual communication resources that belong to the communications group are *loaded*.

The COMTC LOAD applies the change requests to the ALCS online communication table. For example, if you are loading a communications group that adds new 3270 terminals, when the COMTC LOAD has successfully completed, those 3270 terminals will be defined in the online communication table. The initial status of resources that are loaded in the communication table is *inactive* (except for OSYS *other system terminals* and resources defined as *test* terminals). If you require the status to be changed to *active*, use the &comss. macro to do this.

ALCS verifies that the communication change requests being loaded currently exist on the Online Communication Table Maintenance (OCTM) database. ALCS also verifies that they are not in the *confirmed* or *committed* status. ALCS changes the status of the change requests on the OCTM database to *loaded*.

On return from COMTC LOAD, an L3 size storage block resides at the ECB storage level specified by the **LEVEL**= parameter. When the communication change requests are loaded, if errors occur, appropriate error messages are placed in this storage block. Test the return code in register 15 to determine if any errors were found.
Communication load errors can also occur when ALCS processes the ZACOM LOAD command (this command is used for loading a communications load module created by ALCS communication generation). Communication load errors that occur during COMTC LOAD and ZACOM LOAD are reported on the MVS console. The load error messages include the prefix DXC*nnn*, where the message number *nnn* is in the range 000 to 479. Details of these error messages can be found in ALCS Messages and Codes. COMTC LOAD therefore reports each load error on the MVS console and in the storage block. When more than 49 load errors occur during COMTC LOAD, the storage block contains only the first 49 load errors. Each error message will be a maximum of 80 bytes, therefore the storage block is formatted into 80-byte items. A 4-byte header resides at the beginning of the block giving the count of error messages in the block. Immediately following the header is the first error message item.

When you analyze the load errors that have been reported for a communications group, you may wish to cancel each change request that resulted in a load error. To do this, use **COMTC BACKUP** to back out the communications group, use **COMTC CANCEL** to cancel each change request that caused a load error, and then use COMTC LOAD to reload the communications group.

Register use

On return from COMTC LOAD, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC LOAD macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC LOAD macro because of system error - check RO CRAS for a system error dump

COMTC_R_LOADERR
ALCS was unable to load one or more communication change requests in the online communications table. A storage block at the ECB storage level specified by the **LEVEL**= parameter contains error messages describing up to 49 different load errors. The storage block contains an error message count in the first fullword followed by 80-byte items that contain the individual error messages.

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_FCRN
The communication resource name on the **CRN**= parameter is invalid

COMTC_S_ACRN
The address of the field that the **CRN**= parameter references is invalid

COMTC_S_FCRI
The communication resource identifier on the **CRI**= parameter is invalid

COMTC_S_ACRI
The address of the field that the **CRI**= parameter references is invalid

COMTC_S_FGRP
The communications group name on the **GROUP**= parameter is invalid

COMTC_S_NGRP
The communications group name on the **GROUP**= parameter is unknown

COMTC_S_AGRP
The address of the field that the **GROUP**= parameter references is invalid

COMTC_S_CGRP
Another COMTC macro may currently be attempting to unallocate the communications group defined on the **GROUP**= parameter. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_IPGS
A previous COMTC macro for this communication resource or group failed to complete. Use COMTC QUERY to determine the status of the communication resource or group.

COMTC_S_ESEQ
The change requests for this communication resource or group are in the *confirmed* or *committed* status

COMTC_S_NCHG
There are no change requests for this communication resource or communications group

COMTC_S_LOAD
ALCS experienced a serious error when loading the change requests into the online communications table. Some data in the online communications table may be incorrect and it may require an ALCS restart to clear this problem. When this error occurs, ALCS outputs a dump (dump number 000EEA) for diagnostic purposes. If required, forward this dump to IBM for analysis.

COMTC_S_NOLD
ALCS has not loaded any change requests into the online communications table. This can occur when a COMTC LOAD has previously been issued for the communications resource or group and ALCS found no additional change requests that required loading. This can also occur when COMTC LOAD has been issued for a communications group, but ALCS was unable to load any of the change requests in the group because of multiple load errors. A storage block at the ECB storage level specified by the **LEVEL**= parameter contains error messages for up to 49 load errors.

COMTC LOAD may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC LOAD will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=LOAD to load a change request that had been previously submitted via COMTC ADD for a communication resource.

```
         COMTC ACTION=LOAD,        ISSUE LOAD REQUEST
               CRN=EBW020,         CRN IS IN EBW020
               LEVEL=D5            ECB LEVEL FOR ERROR MSG BLOCK
```

Related information

“COMTC ADD - Add a new communication resource” on page 601.
“COMTC REPLACE - Change a communication resource” on page 607.
“COMTC DELETE - Delete a communication resource” on page 611.

COMTC BACKUP - Back out communication change requests

Format

```
[label]  COMTC ACTION=BACKOUT,{CRN={field|(reg1)}|CRI={field|(reg2)}|GROUP={field|
(reg3)}}
                            ,LEVEL={Dn|(reg4)}
```

label
Any valid assembler label.

CRN={*field*|(*reg1*)}
The Communication Resource Name (CRN) of the communication resource whose change request is being backed out. The CRN must be an alphanumeric string of 1 to 8 characters. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of an 8-byte field that contains the CRN.

(*reg1*)
Register containing the address of an 8-byte field that contains the CRN. Use general registers 0 through 7.

CRI={*field*|(*reg2*)}
The Communication Resource Identifier (CRI) of the communication resource whose change request is being backed out. The CRI must be 6 hexadecimal digits. Specify one of:

field
Assembler label of a 3-byte field that contains the CRI.

(*reg2*)
Register containing the address of a 3-byte field that contains the CRI. Use general registers 0 through 7.

GROUP={*field*|(*reg3*)}
The name of the communications group whose change requests are being backed out. It must be an alphanumeric string of 4 to 7 characters, although the first character can not be numeric. If the group name is less than 7 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of a 7-byte field that contains the name of the communications group.

(*reg3*)
Register containing the address of a 7-byte field that contains the name of the communications group. Use general registers 0 through 7.

LEVEL={D*n*|(*reg4*)}
An available storage level in the Entry Control Block (ECB). An L3 size storage block is attached by ALCS to this storage level for recording errors that occur when the communication change requests are backed out in the online communication table. The storage block contains up to 49 error messages (with each message giving the name of the resource and a description of the error). Specify one of:

D*n*
Level symbol: D0, D1, and so on up to DF for level 15.

(*reg4*)
Register containing the level value (use LA *reg*,D*n*). Use general registers 0 through 7.

Description

Use the COMTC macro with the ACTION=BACKOUT option to back out communication change requests that have been previously *loaded* in the ALCS online communication table (via **COMTC LOAD**). Communication change requests that have been *confirmed* (via **COMTC CONFIRM**) can also be backed out. The **COMTC ADD**, **COMTC REPLACE** and **COMTC DELETE** macros are used to submit change requests to ALCS. Use the **CRN**= or **CRI**= parameter to back out a change request for a specific communication resource. Use the **GROUP**= parameter to back out a group of change requests for a communications group. When COMTC BACKUP is used for a communications group, all the change requests that have been submitted for individual communication resources that belong to the group are *backed out*.

The COMTC BACKUP removes the change requests from the ALCS online communication table. For example, if you are backing out a communications group that added new 3270 terminals, after the *back out* those 3270 terminals will no longer have access to the ALCS system. If you are backing out a communications group that deleted a group of ALC terminals, after the *back out* those ALC terminals will have been reinstated in the ALCS online communication table.

ALCS verifies that the communication change requests being backed out currently exist on the Online Communication Table Maintenance (OCTM) database and that they are in the *loaded* or *confirmed* status.

On return from COMTC BACKUP, an L3 size storage block resides at the ECB storage level specified by the **LEVEL**= parameter. When the communication change requests are backed out, if errors occur, appropriate error messages are placed in this storage block. Test the return code in register 15 to determine if any errors were found.
Similar error messages can be output when ALCS processes a COMTC LOAD. For example, when a COMTC DELETE is being backed out, the communications resource is reinstated (loaded) in the online communication table. Communication load errors can also occur when ALCS processes the ZACOM LOAD command (this command is used for loading a communications load module created by ALCS communication generation). Communication load errors that occur during COMTC BACKUP and ZACOM LOAD are reported on the MVS console. The load error messages include the prefix DXC*nnn*, where the message number *nnn* is in the range 000 to 479. Details of these error messages can be found in ALCS

Messages and Codes. COMTC BACKUP therefore reports each error on the MVS console and in the storage block. When more than 49 errors occur during COMTC BACKUP, the storage block contains only the first 49. Each error message will be a maximum of 80 bytes, therefore the storage block is formatted into 80-byte items. A 4-byte header resides at the beginning of the block giving the count of error messages in the block. Immediately following the header is the first error message item.

Register use

On return from COMTC BACKUP, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC BACKUP macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC BACKUP macro because of system error - check RO CRAS for a system error dump

COMTC_R_LOADERR
ALCS was unable to backout one or more communication change requests from the online communications table. A storage block at the ECB storage level specified by the **LEVEL**= parameter contains error messages describing up to 49 different backout errors. The storage block contains an error message count in the first fullword followed by 80-byte items that contain the individual error messages.

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_FCRN
The communication resource name on the **CRN**= parameter is invalid

COMTC_S_ACRN
The address of the field that the **CRN**= parameter references is invalid

COMTC_S_FCRI
The communication resource identifier on the **CRI**= parameter is invalid

COMTC_S_ACRI
The address of the field that the **CRI**= parameter references is invalid

COMTC_S_FGRP
The communications group name on the **GROUP**= parameter is invalid

COMTC_S_NGRP
The communications group name on the **GROUP**= parameter is unknown

COMTC_S_AGRP
The address of the field that the **GROUP**= parameter references is invalid

COMTC_S_CGRP
Another COMTC macro may currently be attempting to unallocate the communications group defined on the **GROUP**= parameter. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_IPGS
A previous COMTC macro for this communication resource or group failed to complete. Use COMTC QUERY to determine the status of the communication resource or group.

COMTC_S_ESEQ
The change requests for this communication resource or group must be in the *loaded* or *confirmed* status

COMTC_S_NCHG
There are no change requests for this communication resource or communications group

COMTC_S_LOAD
ALCS experienced a serious error when performing a backout of change requests from the ALCS online communications table. Some data in the online communications table may be incorrect and it may require an ALCS restart to clear this problem. When this error occurs, ALCS outputs a dump (dump number 000EEA) for diagnostic purposes. If required, forward this dump to IBM for analysis.

COMTC_S_NOLD
Multiple change requests in a communications group were being backed out from the online communications table, but ALCS was unable to backout any of them. A storage block at the ECB storage level specified by the **LEVEL**= parameter contains error messages for up to 49 backout errors.

COMTC_S_ABCK
The communications group or resource has already been backed out.

COMTC BACKUP may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC BACKUP will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=BACKOUT to back out change requests that had been previously loaded for a communications group.

```
        COMTC ACTION=BACKOUT,     ISSUE BACKOUT REQUEST
              GROUP=EBX040,       NAME OF COMMS GROUP IN EBX040
              LEVEL=D3            ECB LEVEL FOR ERROR MSG BLOCK
```

Related information

“COMTC LOAD - Load communication change requests” on page 616.
“COMTC CONFIRM - Confirm communication change requests” on page 622.

COMTC CONFIRM - Confirm communication change requests

Format

```
[label]  COMTC ACTION=CONFIRM,{CRN={field|(reg1)}|CRI={field|(reg2)}|GROUP={field|
(reg3)}}
```

label
Any valid assembler label.

CRN={*field*|(*reg1*)}
The Communication Resource Name (CRN) of the communication resource whose change request is being confirmed. The CRN must be an alphanumeric string of 1 to 8 characters. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of an 8-byte field that contains the CRN.

(*reg1*)
Register containing the address of an 8-byte field that contains the CRN. Use general registers 0 through 7.

CRI={*field*|(*reg2*)}
The Communication Resource Identifier (CRI) of the communication resource whose change request is being confirmed. The CRI must be 6 hexadecimal digits. Specify one of:

field
Assembler label of a 3-byte field that contains the CRI.

(*reg2*)
Register containing the address of a 3-byte field that contains the CRI. Use general registers 0 through 7.

GROUP={*field*|(*reg3*)}
The name of the communications group whose change requests are being confirmed. It must be an alphanumeric string of 4 to 7 characters, although the first character can not be numeric. If the group name is less than 7 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of a 7-byte field that contains the name of the communications group.

(*reg3*)
Register containing the address of a 7-byte field that contains the name of the communications group. Use general registers 0 through 7.

Description

Use the COMTC macro with the ACTION=CONFIRM option to confirm communication change requests that have been previously *loaded* in the ALCS online communication table (via **COMTC LOAD**). The **COMTC ADD**, **COMTC REPLACE** and **COMTC DELETE** macros are used to submit communication change requests to ALCS. The COMTC CONFIRM enables those change requests to be reloaded in the ALCS online communication table during the next ALCS restart.

Use the **CRN**= or **CRI**= parameter to confirm a change request for a specific communication resource. Use the **GROUP**= parameter to confirm a group of change requests for a communications group. When COMTC CONFIRM is used for a communications group, all the change requests that have been submitted for individual communication resources that belong to the communications group are *confirmed*.

ALCS verifies that the communication change requests being confirmed currently exist on the Online Communication Table Maintenance (OCTM) database and that they are in the *loaded* status (but not *committed*). The COMTC CONFIRM changes the status of the communication change requests on the OCTM database to *confirmed*. After a communication resource or a communications group has been *confirmed*, no further change requests can be applied to it.

Register use

On return from COMTC CONFIRM, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC CONFIRM macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC CONFIRM macro because of system error - check RO CRAS for a system error dump

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_FCRN
The communication resource name on the **CRN**= parameter is invalid

COMTC_S_ACRN
The address of the field that the **CRN**= parameter references is invalid

COMTC_S_FCRI
The communication resource identifier on the **CRI**= parameter is invalid

COMTC_S_ACRI
The address of the field that the **CRI**= parameter references is invalid

COMTC_S_FGRP
The communications group name on the **GROUP**= parameter is invalid

COMTC_S_NGRP
The communications group name on the **GROUP**= parameter is unknown

COMTC_S_AGRP
The address of the field that the **GROUP**= parameter references is invalid

COMTC_S_CGRP
Another COMTC macro may currently be attempting to unallocate the communications group defined on the **GROUP**= parameter. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_IPGS
A previous COMTC macro for this communication resource or group failed to complete. Use COMTC QUERY to determine the status of the communication resource or group.

COMTC_S_ESEQ
The change requests for this communication resource or group must be in the *loaded* status

COMTC_S_NCHG
There are no change requests for this communication resource or communications group

COMTC CONFIRM may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC CONFIRM will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=CONFIRM to confirm a change request that had been previously submitted via COMTC ADD for a communication resource.

```
         COMTC ACTION=CONFIRM,     ISSUE CONFIRM REQUEST
               CRN=EBW020          CRN IS IN EBW020
```

Related information

"COMTC LOAD - Load communication change requests" on page 616.

COMTC COMMIT - Commit communication change requests

Format

```
[label]  COMTC ACTION=COMMIT,{CRN={field|(reg1)}|CRI={field|(reg2)}|GROUP={field|
(reg3)}}
```

label
Any valid assembler label.

CRN={*field*|(*reg1*)}
The Communication Resource Name (CRN) of the communication resource whose change request is being committed. The CRN must be an alphanumeric string of 1 to 8 characters. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of an 8-byte field that contains the CRN.

(*reg1*)
Register containing the address of an 8-byte field that contains the CRN. Use general registers 0 through 7.

CRI={*field*|(*reg2*)}
The Communication Resource Identifier (CRI) of the communication resource whose change request is being committed. The CRI must be 6 hexadecimal digits. Specify one of:

field
Assembler label of a 3-byte field that contains the CRI.

(*reg2*)
Register containing the address of a 3-byte field that contains the CRI. Use general registers 0 through 7.

GROUP={*field*|(*reg3*)}
The name of the communications group whose change requests are being committed. It must be an alphanumeric string of 4 to 7 characters, although the first character can not be numeric. If the group name is less than 7 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of a 7-byte field that contains the name of the communications group.

(*reg3*)
Register containing the address of a 7-byte field that contains the name of the communications group. Use general registers 0 through 7.

Description

Use the COMTC macro with the ACTION=COMMIT option to commit communication change requests on the Online Communication Table Maintenance (OCTM) database that have been previously *confirmed* (via **COMTC CONFIRM**). The **COMTC ADD**, **COMTC REPLACE** and **COMTC DELETE** macros are used to submit communication change requests to ALCS. The COMTC COMMIT makes those change requests permanent on the OCTM database.

Use the **CRN**= or **CRI**= parameter to commit a change request for a specific communication resource. Use the **GROUP**= parameter to commit a group of change requests for a communications group. When COMTC COMMIT is used for a communications group, all the change requests that have been submitted for individual communication resources that belong to the communications group are *committed*.

ALCS verifies that the communication change requests being committed currently exist on the OCTM database and that they are in the *confirmed* status. After change requests have been *committed* for one or more communication resources, new change requests can be submitted for those communication resources.

Register use

On return from COMTC COMMIT, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC COMMIT macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC COMMIT macro because of system error - check RO CRAS for a system error dump

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_FCRN
The communication resource name on the **CRN**= parameter is invalid

COMTC_S_ACRN
The address of the field that the **CRN**= parameter references is invalid

COMTC_S_FCRI
The communication resource identifier on the **CRI**= parameter is invalid

COMTC_S_ACRI
The address of the field that the **CRI**= parameter references is invalid

COMTC_S_FGRP
The communications group name on the **GROUP**= parameter is invalid

COMTC_S_NGRP
The communications group name on the **GROUP**= parameter is unknown

COMTC_S_AGRP
The address of the field that the **GROUP**= parameter references is invalid

COMTC_S_CGRP
Another COMTC macro may currently be attempting to unallocate the communications group defined on the **GROUP**= parameter. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_IPGS
A previous COMTC macro for this communication resource or group failed to complete. Use COMTC QUERY to determine the status of the communication resource or group.

COMTC_S_ESEQ
The change requests for this communication resource or group must be in the *confirmed* status

COMTC_S_NCHG
There are no change requests for this communication resource or communications group

COMTC COMMIT may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC COMMIT will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=COMMIT to commit a change request that had been previously submitted via COMTC ADD for a communication resource.

```
         COMTC ACTION=COMMIT,      ISSUE COMMIT REQUEST
               CRN=EBW020          CRN IS IN EBW020
```

Related information

"COMTC CONFIRM - Confirm communication change requests" on page 622.

COMTC UNALLOCATE - Unallocate a communications group

Format

```
[label]  COMTC ACTION=UNALLOCATE,GROUP={field|(reg)}
                          [,DELETE={NO|YES}]
```

label
Any valid assembler label.

GROUP={*field*|(*reg*)}
The name of the communications group that is being unallocated on the Online Communication Table Maintenance (OCTM) database. It must be an alphanumeric string of 4 to 7 characters, although the first character can not be numeric. If the group name is less than 7 characters, it should be left justified and padded with spaces. Specify one of:

field
Assembler label of a 7-byte field that contains the name of the communications group.

(*reg*)
Register containing the address of a 7-byte field that contains the name of the communications group. Use general registers 0 through 7.

DELETE={NO|YES}
Specifies how the unallocate request should be processed if the communications group is not *committed* and there are communication change requests that belong to it.

NO
Do not unallocate the communications group if there are communication change requests belonging to it and the communications group has not been *committed* (via **COMTC COMMIT**).

YES
Unallocate the communications group and delete (cancel) all the communications change requests that belong to it.

Description

Use the COMTC macro with the ACTION=UNALLOCATE option to unallocate (remove) a communications group on the OCTM database. When all the communications change requests belonging to a group have been *committed* (via **COMTC COMMIT**) the group name should be unallocated. After the communications group has been unallocated, the group name becomes available again for reuse. If the same group name is required again for applying new communications change requests, the group name must be unallocated and then allocated again (via **COMTC ALLOCATE**).

If the DELETE=NO option is used and the communications group has not yet been *committed*, ALCS will not allow the communications group to be unallocated.

If the communications group has communication change requests belonging to it that have not yet been *loaded* (via **COMTC LOAD**) or have been *backed out* (via **COMTC BACKUP**), use the DELETE=YES option to unallocate this communications group and delete (cancel) from the OCTM database all the change requests that belong to it.

Register use

On return from COMTC UNALLOCATE, a return code will reside in the low-order two bytes (bits 16 to 31) of general register 15. Test the return code that COMTC has set by using the following symbols:

COMTC_R_OK
Processing completed without error

COMTC_R_RETRY
Unable to obtain lock for communication resource or group - retry the COMTC UNALLOCATE macro immediately as this is only a temporary condition

COMTC_R_ERROR
An error occurred during the processing of the COMTC macro. There are various types of error that might occur including errors in the COMTC macro parameters. Check the reason code in the high-order two bytes of register 15 (as described below) for details of the error.

COMTC_R_SYSERR
Unable to process the COMTC UNALLOCATE macro because of system error - check RO CRAS for a system error dump

When general register 15 contains the return code COMTC_R_ERROR, a reason code will reside in the high-order two bytes (bits 0 to 15) of the register. Test the reason code that COMTC has set by using the following symbols:

COMTC_S_ACTM
The OCTM database is currently unavailable because the OCTM facility has been stopped by the ZOCTM STOP command - retry the COMTC macro when OCTM is active again

COMTC_S_USRL
The OCTM database is currently unavailable because the length of the communications user data area has been changed in a communications generation load module (**COMGEN USERLEN**= parameter) - retry the COMTC macro when an OCTM backup and restore has been performed

COMTC_S_FGRP
The communications group name on the **GROUP**= parameter is invalid

COMTC_S_NGRP
The communications group name on the **GROUP**= parameter is unknown

COMTC_S_AGRP
The address of the field that the **GROUP**= parameter references is invalid

COMTC_S_CGRP
Another COMTC macro may currently be attempting to unallocate the communications group defined on the **GROUP**= parameter. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_IPGS
A previous COMTC macro for this communications group failed to complete. Use COMTC QUERY to determine the status of the communications group.

COMTC_S_NUNL
The communications group can not be unallocated. The DELETE=NO option was used but there are communication change requests which belong to the communications group that have not yet been *committed*.

COMTC_S_YUNL
The communications group can not be unallocated. The DELETE=YES option was used but there are communication change requests which belong to the communications group that are in the *loaded* or *confirmed* status. Use COMTC BACKUP to back out the change requests from the online communication table and issue the COMTC UNALLOCATE macro again.

COMTC UNALLOCATE may also use general register 14. It does not corrupt any other registers.

Loss of control

COMTC UNALLOCATE will cause the entry to lose control.

Example

The following example shows how you could use the COMTC macro with ACTION=UNALLOCATE to unallocate a communications group. In this example, the communications group included some invalid change requests, therefore COMTC UNALLOCATE is used to not only unallocate the communications group but to delete every communications change request that belongs to the group.

```
UNALLOC  EQU   *
         COMTC ACTION=UNALLOCATE,  ISSUE UNALLOCATE REQUEST
               GROUP=(R06),        GROUP NAME POINTED TO BY REGISTER 6
               DELETE=YES          REQUEST DELETION OF CHANGE REQUESTS
         LR    R04,R14             LOAD COUNT OF COMMS RESOURCES
         LR    R14,R15             LOAD RETURN CODE / REASON CODE
         SLL   R14,16              ISOLATE RETURN CODE
         SRL   R14,16              SHIFT TO LOW-ORDER BYTES
         B     *+4(R14)            BRANCH ON RETURN CODE
         B     URC0                0 - OK
         B     URC4                4 - RETRY IMMEDIATELY
         B     URC8                8 - USER ERROR
         B     URC12               12 - SYSTEM ERROR
URC4     EQU   *
         DEFRC ,                   WAIT A MOMENT
         B     UNALLOC             RETRY THE COMTC REQUEST
URC0     EQU   *
```

Related information

"COMTC ALLOCATE - Allocate a new communications group" on page 599.

Appendix M. DSECTs used by COMTC macro

The OCTM interface provided for the CEUS not only includes the COMTC macro but also DSECT macros. This section provides a detailed description of the DSECT's that are used via the COMTC macro.

CT1TM - Communication Resource Definition DSECT

Format

```
CT1TM REG=reg[,SUFFIX=s]
```

REG=*reg*
Base register for DSECT addressability.

SUFFIX=*s*
Suffix for this DSECT, a single alphanumeric character.

Description

Use the CT1TM DSECT to reference fields in the Communication Resource Definition data block. This DSECT is used by the ALCS Online Communication Table Maintenance (OCTM) facility, and it contains information about a communication resource. There are eight types of communication resource that this DSECT can hold information for.

The OCTM Communication Table Update monitor-request macro COMTC uses this DSECT for three of its options.
The first option is **COMTC QUERY** which provides detailed information about a communication resource in a data block that is formatted by this DSECT. **COMTC QUERY** does not reference this DSECT directly, but via the Communication Resource and Group Information DSECT (CT2TM).
The other two options are **COMTC ADD** and **COMTC REPLACE**. These two options require detailed information to be provided about a communication resource in a storage area that is formatted by this DSECT. When this DSECT is used for the **COMTC ADD** macro, the storage area should be initialized with hexadecimal zeros. When this DSECT is used for the **COMTC REPLACE** macro, the storage area should be initialized by obtaining the current definition of the resource via a **COMTC QUERY**.

This DSECT supports eight different communication resource types. The field CT1TYP in this DSECT contains the name of the communication resource type.

When a Communication Resource Definition block is being prepared for the COMTC ADD macro, many of the fields are optional but some are mandatory. Guidance is given below on the fields that are mandatory for the COMTC ADD macro.

The CT1TM DSECT defines the following symbols for fields within the Communication Resource Definition data block. The symbols are described below under various headings with each heading identifying the communication resource types that the symbols relate to.

Symbols for all resource types

The following symbols can be used for all types of communication resource.

CT1TYP
Communication resource type
Eight byte field that identifies the communication resource type that this communication resource belongs to. For example, if this field contains the character string VTAM3270, the resource type

© Copyright IBM Corp. 2003, 2019

is a 3270 terminal. In the communications generation, this is provided in the COMDEF **LDTYPE**= parameter.

Resource Type
Resource Type Description

MQTERM
The MQSeries terminal resource type, provided for those terminals that connect to ALCS through an MQSeries queue.

NETVIEW
The NetView operator ID resource type, provided for NetView operator terminals.

OSYS
The Other System terminal resource type, provided for those terminals that access this ALCS system through another system (who owns and controls the terminal).

TCPIPALC
The TCP/IP ALC terminal resource type, provided for those terminals that connect to ALCS through a TCP/IP network.

VTAM3270
The VTAM 3270 terminal resource type, provided for 3270 terminals.

VTAMALC
The VTAM/ALCI ALC terminal resource type, provided for those ALC terminals that connect to ALCS through ALCI and VTAM.

WASTERM
The WAS terminal resource type, provided for those terminals that connect to ALCS through WAS.

X25PVC
The X.25 permanent virtual circuit (PVC) resource type, provided for the connectivity of ALC terminals through NPSI and VTAM.

X25ALC
The X.25 ALC terminal resource type, provided for those ALC terminals that connect to ALCS through NPSI, VTAM and an X.25 PVC.

`COMTC ADD` considerations:
This field is mandatory. The name of the communication resource type must be left justified and padded with spaces.

`COMTC REPLACE` considerations:
Although the communication resource type can not be changed, this field must though contain the name of the communication resource type that this communication resource belongs to.

CT1USD
Communication resource user data
A 256 byte field that contains communications user data that is associated with this communication resource. If your ALCS installation provides user data for the communication resources, you should have a data macro that defines each field in the user data area. Define the user data that requires refreshing in the online communication table on an ALCS restart. In the communications generation, this is provided in the COMDEF **USERDAT**= or **USERnn**= parameters.

`COMTC ADD` considerations:
The user data that is to be placed in this field is normally less than 256 bytes. Therefore, before inserting the user data in this field, initialize it by inserting a *pad character* in each byte. The CT1UPD symbol described below defines the hexadecimal value for the *pad character* (which can be hexadecimal zero). Insert the user data that you require in this field, overlaying the *pad character*. ALCS does not validate the user data, therefore any errors in this data will not be detected by ALCS. If user data is not being provided, each byte should either contain hexadecimal zero or the *pad character*.

`COMTC REPLACE` considerations:
If the communications user data is not being changed, do not alter the user data supplied by `COMTC QUERY`. If communications user data is being changed, initialize this field by placing a *pad character* in

each byte. The CT1UPD symbol described below defines the hexadecimal value for the *pad character* (which can be hexadecimal zero). Insert the user data that requires changing in this field, overlaying the *pad character*.

CT1UPD
User data pad character
One byte field that contains the communications user data *pad character*.

COMTC ADD considerations:
If communications user data is not being provided for this communication resource (field CT1USD is hexadecimal zeros) this field should contain hexadecimal zero. If communications user data is being provided, define a *pad character* in this field. The hexadecimal *pad character* must reside in each byte of CT1USD that does not contain communications user data. The *pad character* can be any hexadecimal value.

COMTC REPLACE considerations:
If the communications user data is not being changed, leave the contents of this field as hexadecimal zero (this is the value that COMTC QUERY places in this field).
If communications user data is being changed, provide a *pad character* in this field. Initialize the 256-byte user data field CT1USD by placing this *pad character* in each byte, and then insert the user data that requires changing in CT1USD (overlaying the *pad character*).

CT1IST
Initial status
Status byte that defines the initial status of the communication resource when the ALCS system starts. In the communications generation, this is provided in the COMDEF **ISTATUS**= parameter. Initial status must not be defined for the resource type NETVIEW.

Symbol
Meaning if on (1)

CT1ACTV
The initial status is *active*. ALCS will initiate communication with this resource when the system starts.

CT1INAC
The initial status is *inactive*. ALCS will start this communication resource only in response to an external request (for example, a ZACOM command).

CT1SHAR
The initial status is *shared*. This symbol is valid only for VTAM 3270 printers. ALCS will share this 3270 printer with one or more other z/OS subsystems.

COMTC ADD considerations:
If you do not specify the initial status for the communication resource, ALCS will allocate a status of *active*.

COMTC REPLACE considerations:
If the initial status is not being changed, do not modify the value provided by COMTC QUERY. If the initial status is being changed, set the contents of the status byte using the appropriate symbol.

CT1FLA
Status flag byte
Status byte that defines the *logon* and *test* status for the communication resource.

Symbol
Meaning if on (1)

CT1LOGO
The user of this communication resource must logon to the ALCS system with *userid* and *password*. The *logon* option is used for System Authorization Facility (SAF) authorization checking. In the communications generation, this is provided in the COMDEF **LOGON**= parameter. Logon is not allowed for the resource types NETVIEW, OSYS, and X25PVC. If the user of this communication resource is not required (or allowed) to logon, the resource requires a default user ID (provided in field CT1UID).

CT1TEST
The communication resource is a *test* resource. ALCS *test* resources receive their input messages from the ALCS System Test Vehicle (STV). In the communications generation, this is provided in the COMDEF **TEST**= parameter. The following resource types can be defined as *test* resources, VTAM3270, VTAMALC, X25ALC, and X25PVC.

COMTC ADD considerations:
If logon is required for this resource, or if it is a test resource, set the contents of this status flag byte using the appropriate symbol. Note that a communication resource can not have both of these options set on (it can be neither of them, or just one of them, but not both).

COMTC REPLACE considerations:
If the logon or test status for this resource is not being changed, do not modify the value provided by COMTC QUERY. Alternatively, set the contents of this status flag byte using the appropriate symbol. Note that only one of these options can be selected, not both.

CT1UID
Default user ID
Eight byte field that contains the default user ID for this communication resource. In the communications generation, this is provided in the COMDEF **LOGON**= parameter. A default user ID must not be defined for the resource type NETVIEW. When this field is not used, it should contain spaces or hexadecimal zeros. The default user ID is used by the z/OS System Authorization Facility (SAF) for authorization checking and is associated with the logon option for this communications resource. The CT1LOGO symbol described above provides the logon option. For some communication resource types, logon is not allowed, but for others it is an optional feature.

COMTC ADD considerations:
The default user ID must be left justified and padded with spaces. If a default user ID is not being provided, set the contents of this field to spaces or hexadecimal zero.
If you specify that users of this communications resource must always logon to the ALCS system with *user ID* and *password* (the CT1LOGO symbol is set on), you do not need to provide a default user ID.
If you specify that logon is not required, you must provide a default user ID (ALCS does not use the default user ID provided in the **DEFAULTID**= parameter on the communication generation COMGEN macro).

COMTC REPLACE considerations:
If the default user ID for this resource is not being changed, do not modify the value provided by COMTC QUERY.
If you are deleting the default user ID, place spaces or hexadecimal zero in this field.
If you are providing a new or changed default user ID, but you have also specified that users of this communications resource must always logon to the ALCS system with *user ID* and *password* (the CT1LOGO symbol is set on) ALCS will not use this default user ID. If the CT1LOGO symbol is set off (logon is not required), ensure that this field contains a default user ID.

Symbols for the terminal resource types

The following symbols can be used for the terminal resource types. If the communication resource is an X.25 permanent virtual circuit (PVC), the symbols in the following section can be ignored.

CT1APL
Application name
Four byte field that contains the name of the ALCS application to which ALCS routes messages from this communication resource. In the communications generation, this is provided in the COMDEF **APPL**= parameter. The application resource type definitions in the communication generation (LDTYPE=ALCSAPPL) define the valid application names for your ALCS system.

COMTC ADD considerations:
An application name ***must*** be provided for the resource. ALCS does not provide a default application name. The application name must be an alphanumeric string, and if it is less than 4 characters, it should be left justified and padded with spaces.

COMTC REPLACE considerations:

If the application name for the resource is not being changed, do not modify the value provided by COMTC QUERY. If the application name is being changed and it is less than 4 characters, it should be left justified and padded with spaces.

CT1ASD
Associated resource CRN
Eight byte field that contains the CRN of the associated communication resource. In the communications generation, this is provided in the COMDEF **ASSDEV**= parameter. When this field is not used, it should contain spaces or hexadecimal zeros.

COMTC ADD considerations:
Ensure that the communications definition for the associated resource has already been defined in your ALCS system. If the CRN of the associated resource is less than 8 characters, it should be left justified and padded with spaces. If this communication resource is a CRAS display, you are recommended to associate a CRAS printer with that display because some CRAS commands output response messages to an associated printer.

COMTC REPLACE considerations:
If the associated communication resource is not being changed, do not modify the value provided by COMTC QUERY. If the current associated resource is being removed (the terminal will no longer have an associated resource), set the contents of this field to hexadecimal zero. If the CRN of the associated resource is being changed, ensure that the communications definition for the associated resource has already been defined in your ALCS system. If the CRN of the associated resource is less than 8 characters, it should be left justified and padded with spaces.

CT1CRS
Alternate CRAS type
Flag byte that indicates if the communication resource is an alternate CRAS terminal. In the communications generation, this is provided in the COMDEF **CRAS**= parameter. All terminal resource types (displays and printers) can be an alternate CRAS type of AT, but only printer resource types can be an alternate CRAS type of AP. The alternate CRAS number is defined in field CT1ALT (described below).

Note: The terminals which have Prime CRAS status or RO CRAS status can not be defined in the OCTM database (these terminals must be defined in the ALCS communications generation).

Symbol
Meaning if on (1)

CT1CRSAT
This communication resource is an alternate (AT*nnn*) CRAS

CT1CRSAP
This communication resource is an alternate (AP*nnn*) printer CRAS

COMTC ADD considerations:
If the terminal is not being given alternate CRAS status, this flag byte must contain hexadecimal zero. If the terminal requires alternate CRAS status, set the contents of this flag byte to indicate the alternate CRAS type (AT or AP). Ensure that you also set the alternate CRAS number in field CT1ALT (described below). Terminal resource types VTAM3270,, NETVIEW,, MQTERM (TERM=3270DSP),, MQTERM (TERM=3270PRT),, WASTERM (TERM=3270DSP) and WASTERM (TERM=3270PRT) can have any alternate CRAS number, but all other terminal resource types are restricted to the alternate CRAS number of AT17 to AT255 and AP17 to AP255.

COMTC REPLACE considerations:
If the CRAS status for this communication resource is not being changed, do not modify the value provided by COMTC QUERY. If you are removing alternate CRAS status from this terminal, set the contents of this flag byte to hexadecimal zero. If you are changing the CRAS status (for example, from an AT CRAS to an AP CRAS) set the contents of this flag byte using the appropriate symbol.

CT1ALT
Alternate CRAS number

One byte field that contains the alternate CRAS number *nnn* (AT*nnn* or AP*nnn*). In the communications generation, this is provided in the COMDEF **CRAS**= parameter. If the terminal has an alternate CRAS number, the flag byte CT1CRS, as described above, indicates the alternate CRAS type (AT or AP). If the terminal is not an alternate CRAS status, this field will contain hexadecimal zero. For terminal resource types VTAM3270, NETVIEW,, MQTERM (TERM=3270DSP),, MQTERM (TERM=3270PRT),, WASTERM (TERM=3270DSP) and WASTERM (TERM=3270PRT) the alternate CRAS number is between 1 and 255. For all other terminals, the alternate CRAS number is between 17 and 255.

COMTC ADD considerations:
For AT CRAS terminals, the alternate CRAS number must not be allocated to any other AT CRAS terminal. Similarly, for AP CRAS printers, the alternate CRAS number must not be allocated to any other AP CRAS printer. Terminal resource types VTAM3270, NETVIEW,, MQTERM (TERM=3270DSP),, MQTERM (TERM=3270PRT),, WASTERM (TERM=3270DSP), and WASTERM (TERM=3270PRT) can have any alternate CRAS number, but all other terminal resource types are restricted to the alternate CRAS number of 17 to 255. If the terminal is not an alternate CRAS, set the contents of this field to hexadecimal zero.

COMTC REPLACE considerations:
If the alternate CRAS number for this terminal resource is not being changed, do not modify the contents of this field (as provided by COMTC QUERY). If you are removing the alternate CRAS status from this terminal (field CT1CRS has been set to hexadecimal zero), set the contents of this field to hexadecimal zero.
If you are changing the alternate CRAS number for this terminal, note that terminal resource types VTAM3270, NETVIEW,, MQTERM (TERM=3270DSP),, MQTERM (TERM=3270PRT),, WASTERM (TERM=3270DSP), and WASTERM (TERM=3270PRT) can have any alternate CRAS number, but all other terminal resource types are restricted to the alternate CRAS number of 17 to 255. For AT CRAS terminals, the alternate CRAS number must not be allocated to any other AT CRAS terminal. Similarly, for AP CRAS printers, the alternate CRAS number must not be allocated to any other AP CRAS printer.

CT1TRM
Terminal device type
Flag byte that defines the device type of the terminal resource. In the communications generation, this is provided in the COMDEF **TERM**= parameter.

Symbol
Meaning if on (1)

CT11977
This communication resource emulates an ALC 1977 terminal (ALC keyboard printer)

CT11980
This communication resource emulates an ALC 1980 terminal (ALC receive-only printer)

CT12915
This communication resource emulates an ALC 2915 terminal (ALC 12 line display)

CT14505
This communication resource emulates an ALC 4505 terminal (ALC 30 line display)

CT1327P
This communication resource is a 3270 printer

CT1327D
This communication resource is a 3270 display

COMTC ADD considerations:
For most terminal resources, the terminal device type ***must*** be provided. The only exception is for VTAM 3270 terminals which are not system test vehicle (STV) *test* resources (for these terminals, ALCS accepts the device type you have specified until a VTAM session is established with the terminal). Any of the ALC terminal device types can be used for the MQTERM WASTERM , OSYS, TCPIPALC, VTAMALC and X25ALC resource types. Either of the 3270 terminal device types can be used for the MQTERM,, WASTERM , NETVIEW, OSYS and VTAM3270 resource types.

COMTC REPLACE considerations:

If the terminal device type for this communication resource is not being changed, do not modify the value provided by COMTC QUERY. If you are changing the terminal device type, set the contents of this flag byte using the appropriate symbol. Any of the ALC terminal device types can be used for the MQTERM,, WASTERM , OSYS, TCPIPALC, VTAMALC and X25ALC resource types. Either of the 3270 terminal device types can be used for the MQTERM,, WASTERM , NETVIEW, OSYS and VTAM3270 resource types.

CT1CSI
Cross system identifier
Three byte field that contains the cross system identifier for this terminal resource. In the communications generation, this is provided in the COMDEF **CSID**= parameter. For terminal resources that do not have a cross system identifier, this field contains hexadecimal zero. The cross system identifier, which is 6 hexadecimal digits, is required by terminal resources that are owned by another system. Resource types MQTERM,, WASTERM , and NETVIEW do not have a cross system identifier.

COMTC ADD considerations:
When a cross system identifier is not being provided for the communication resource, this field must contain hexadecimal zero. A cross system identifier ***must*** though be provided when the communications resource type is an Other System terminal (OSYS). When the communications resource type is MQTERM,, WASTERM , or NETVIEW a cross system identifier must not be provided.

COMTC REPLACE considerations:
If the cross system identifier for this communication resource is not being changed, do not modify the contents of this field (as provided by COMTC QUERY). If you are changing it, insert the new cross system identifier in this field. Do not though provide a cross system identifier when the communications resource type is MQTERM,, WASTERM , or NETVIEW. If you are deleting the cross system identifier, set the contents of this field to hexadecimal zero. If the communications resource type is an Other System terminal (OSYS), you must not delete the cross system identifier.

CT1FLB
Terminal characteristics
Flag byte that defines specific characteristics of the terminal resource.

Symbol
Meaning if on (1)

CT1FALL
CRAS fallback candidate
The terminal resource is a fallback candidate for Prime CRAS or RO CRAS. In the communications generation, this is provided by the FALLBACK option on the COMDEF **CRAS**= parameter. The only communication resource types that can be CRAS fallback candidates are NETVIEW and VTAM3270. Terminal resources must be an alternate CRAS AT1 to AT16 or AP1 to AP16 to be eligible as a fallback candidate.

COMTC ADD and REPLACE considerations:
If this terminal resource is to be a CRAS fallback candidate, ensure that it is defined as an alternate CRAS in field CT1CRS. Also ensure that it has an alternate CRAS number between 1 and 16 defined in field CT1ALT and that it is a resource type of NETVIEW or VTAM3270. On most ALCS systems there is at least one alternate CRAS display (AT1 to AT16) defined as a fallback candidate for Prime CRAS, and at least one alternate CRAS printer (AT1 to AT16 or AP1 to AP16) defined as a fallback candidate for RO CRAS.

CT1SYSS
Suppress system generated messages
The communication resource must not be sent system generated messages. In the communications generation, this is provided in the COMDEF **SYSSEND**= parameter. The ALCS system can generate *system* messages that it wants to send to the communication resource. This can include system error and unsolicited messages.

COMTC ADD and COMTC REPLACE considerations:
If the communication resource is not to receive system generated messages, ensure that this indicator is set on.

CT1TABC
Tabular skip feature
The communication resource has the tabular skip feature. This feature is available only on the ALC 1977 printer terminal. In the communications generation, this is provided in the COMDEF **TAB**= parameter. The VTAM3270 and NETVIEW resource types do not support ALC printers, therefore this feature is not relevant for those communication resource types.

COMTC ADD and COMTC REPLACE considerations:
The terminal resource must be an ALC 1977 printer if the tabular skip feature is to be used. Ensure that it is defined as an ALC 1977 printer in the field CT1TRM. Also ensure that it is not a NETVIEW or VTAM3270 resource type.

CT1UNSL
Unsolicited light feature
The communication resource has the unsolicited light feature. This feature is available only on the ALC 1977 printer terminal. In the communications generation, this is provided in the COMDEF **UNSLITE**= parameter. This feature is used by ALCS to inform the ALC 1977 terminal that an unsolicited message is waiting to be printed on the terminal. The VTAM3270 and NETVIEW resource types do not support ALC printers, therefore this feature is not relevant for those communication resource types.

COMTC ADD and COMTC REPLACE considerations:
The terminal resource must be an ALC 1977 printer if the unsolicited light feature is to be used. Ensure that it is defined as an ALC 1977 printer in the field CT1TRM. Also ensure that it is not a NETVIEW or VTAM3270 resource type.

CT1DBCS
DBCS support
The communication resource supports the double byte character set. This feature is available only on the VTAM 3270 terminal, MQTERM 3270 terminal, and WASTERM 3270 terminal. In the communications generation, this is provided in the COMDEF **DBCS**= parameter.

COMTC ADD and COMTC REPLACE considerations:
The terminal resource must be a VTAM3270, MQTERM or WASTERM if the DBCS feature is to be used.

CT1LSM
Inhibit scrolling and send the complete unformatted response response to this communication resource.

Notes:

1. The response can be one or multiple blocks when connected to ALCS using MATIP TYPE A TCP/IP connection. Intermediate blocks terminate with an EOMp character and the final block with an EOMc character.
2. The application that constructs the response may only use the DISPC ADD and SEND macros. The DISPC SEND DEST parameter is not allowed and the DISPC SEND START, NUM, SPACE, BOTTOM, LINE, CONFIRM,CLEAR parameters are ignored.
3. Heap storage is used for scrolling. To allow for this, you must allocate type 3 storage units. Specify a type 3 storage unit size greater than or equal the largest response plus 50 bytes (to allow for message headers and trailers).

COMTC ADD and COMTC REPLACE considerations: The terminal resource must be a TCPIPALC or WASTERM if this feature is to be used.

CT1RETR
Command retrieve active
The ALCS message retrieve facility is active for the communication resource. In the communications generation, this is provided in the COMDEF **RETRV**= parameter. The ALCS ZRETR command controls usage of the retrieve facility (start, stop, display, etc.).

CT1SCR

Screen size (lines)

One byte field that contains the screen size. In the communications generation, this is provided in the COMDEF **SCRSZE**= parameter. For terminal resources that do not require a screen size, or use the default screen size, this field contains hexadecimal zero. The screen size is required primarily for ALC display terminals. The default screen size for ALC 2915 displays is 12 lines and for ALC 4505 displays is 30 lines. The default screen size for 3270 displays is 24 lines. 3270 displays which are NETVIEW or VTAM3270 resource types, do not require a screen size.

COMTC ADD considerations:
This field can be set to hexadecimal zero if the terminal resource does not require a screen size or is to use the system defaults (12 lines for the ALC 2915 display, 30 lines for the ALC 4505 display, and 24 lines for the 3270 display). The Other System (OSYS), WAS terminal (WASTERM), and MQSeries terminal (MQTERM) resource types support both ALC and 3270 devices. If a screen size is not provided for an OSYS 3270 display, an MQTERM 3270 display, or a WASTERM 3270 display, a default screen size of 24 lines is used.

COMTC REPLACE considerations:
If the screen size for this communication resource is not being changed, do not modify the value provided by COMTC QUERY. If you are changing the screen size, either modify this field with the required size or store hexadecimal zero to request ALCS to use the default screen size.

CT1COL

Screen size (columns)

One byte field that contains the screen size. In the communications generation, this is provided in the COMDEF **SCRCOL**= parameter. For terminal resources that do not require a screen size, or use the default screen size, this field contains hexadecimal zero. The default screen size for ALC displays is 64 columns. The default screen size for 3270 displays is 80 columns. 3270 displays which are NETVIEW or VTAM3270 resource types, do not require a screen size.

COMTC ADD considerations:
This field can be set to hexadecimal zero if the terminal resource does not require a screen size or is to use the system defaults (64 columnns for ALC displays, and 80 columns for 3270 displays). The Other System (OSYS), WAS terminal (WASTERM), and MQSeries terminal (MQTERM) resource types support both ALC and 3270 devices. If a screen size is not provided for an OSYS 3270 display, an MQTERM 3270 display, or a WASTERM 3270 display, a default screen size of 80 columns is used.

COMTC REPLACE considerations:
If the screen size for this communication resource is not being changed, do not modify the value provided by COMTC QUERY. If you are changing the screen size, either modify this field with the required size or store hexadecimal zero to request ALCS to use the default screen size.

CT1BUF

Printer buffer size

Halfword field that contains the printer buffer size (in bytes). In the communications generation, this is provided in the COMDEF **BUFSZE**= parameter. The printer buffer size, which is between 0 and 4000 bytes, is required primarily for ALC printers. 3270 printers which are NETVIEW or VTAM3270 resource types do not require a printer buffer size. If a terminal resource does not require a printer buffer size, this field contains hexadecimal zero.
If a terminal resource does require a printer buffer size, and this field contains hexadecimal zero, field CT1INI (the printer flag byte) will indicate if a zero buffer size is to be used. If the CT1ZERB symbol is set on in field CT1INI (described below), the buffer size is zero, otherwise ALCS uses the default printer buffer size.

COMTC ADD considerations:
This field can be set to hexadecimal zero if the terminal resource does not require a buffer size. If the terminal resource requires a buffer size of zero, set this field to hexadecimal zero and also set on the CT1ZERB symbol in the CT1INI printer flag byte (described below). If the terminal resource is to use the default buffer size, just set this field to hexadecimal zero. ALCS uses a default buffer size of 97 bytes for ALC printers and a default buffer size of 1920 bytes for 3270 printers (when they

are an OSYS resource type). If a default buffer size is provided on the **BUFSZE**= parameter on the communication generation COMGEN macro, that will be used as the default for ALC printers.

COMTC REPLACE considerations:
If the buffer size for this communication resource is not being changed, do not modify the value provided by COMTC QUERY. If you are changing the buffer size, modify this field with the required size. If you are changing the buffer size to zero, set on the CT1ZERB symbol in the CT1INI printer flag byte (described below). If you want ALCS to use the default buffer size (instead of the size currently being used), store hexadecimal zero in this field (and ensure that symbol CT1ZERB in CT1INI is set off).

CT1TMO
Printer timeout values
Six byte field that contains the printer timeout values. There are three timeout values, and each one occupies a halfword. In the communications generation, this is provided in the COMDEF **TIMEOUT**= parameter. The first halfword contains the answerback timeout in *seconds* (a value between 0 and 300). The second halfword contains the test message transmission interval in *seconds* (a value between 0 and 600). The third halfword contains the retry error count (a value between 0 and 30). Further information on the timeout values can be found in the description of the COMGEN **TIMEOUT**= parameter in the ALCS Installation and Customization manual. Printer timeout values are required only for ALC printers. Printers which are NETVIEW, OSYS or VTAM3270 resource types do not require timeout values. If a terminal resource does not require timeout values, these three halfwords contain hexadecimal zeros.
If a terminal resource requires timeout values, and one or more of the timeout values in this field is set to hexadecimal zero, field CT1INI (the printer flag byte) will indicate if a value of zero is to be used for any of the timeout values. When the printer flag byte CT1INI (described below) has not been used (and the timeout value in this field is zero), ALCS uses the default printer timeout value.

COMTC ADD considerations:
This field can be set to hexadecimal zero if the terminal resource does not require timeout values. If the terminal resource requires any of the three timeout values to be set to zero, set the required halfwords in this field to hexadecimal zero and also set on the appropriate symbol in the CT1INI printer flag byte (described below). If the terminal resource is to use any of the default timeout values, just set the appropriate halfword in this field to hexadecimal zero. ALCS uses the following default timeout values. A default of 50 seconds for the answerback timeout; a default of 300 seconds for the test message transmission interval; and a default of 3 for the retry error count. If a default timeout value is provided on the **TIMEOUT**= parameter on the communication generation COMGEN macro, that will be used as the default for the ALC printers.

COMTC REPLACE considerations:
If the timeout values for this communication resource are not being changed, do not modify the values provided by COMTC QUERY. If you are changing the timeout values, modify this field with the required values. If you are changing any timeout value to zero, set on the appropriate symbol in the CT1INI printer flag byte (described below). If you want ALCS to use the default timeout value instead of the value currently being used, store hexadecimal zero in the appropriate halfword in this field and ensure that the appropriate symbol in CT1INI is set off.

CT1INI
Printer Flag Byte
Flag byte that is used in conjunction with the CT1BUF and CT1TMO fields (as described above) to provide printer buffer and timeout values.

Symbol
Meaning if on (1)

CT1ZERB
A buffer size of zero is to be used for the printer. If the CT1BUF field (the printer buffer size) contains hexadecimal zeros, ALCS will use a zero buffer size.

CT1ZER1
An answerback timeout of zero seconds is to be used for the printer. If the first halfword of the CT1TMO field (printer answerback timeout) contains hexadecimal zeros, ALCS will use a zero value for that timeout (ALCS will wait indefinitely for an answerback).

CT1ZER2
A test message transmission interval of zero seconds is to be used for the printer. If the second halfword of the CT1TMO field (test message transmission interval) contains hexadecimal zeros, ALCS will use a zero value for that interval (ALCS will therefore not send test messages).

CT1ZER3
A recovery retry count of zero is to be used for the printer. If the third halfword of the CT1TMO field (recovery retry count) contains hexadecimal zeros, ALCS will use a zero value for the retry count (ALCS will not retry transmission of the message).

COMTC ADD and REPLACE considerations:
This flag is used only when a zero value is required for the printer buffer size or for any of the printer timeout values. Refer to the descriptions of the CT1BUF and CT1TMO fields for more information on the usage of the printer flag byte.

Symbol for the MQSeries terminal resource type

The following symbol must be used for the MQSeries terminal resource type (MQTERM).

CT1MQL
Owning MQ queue resource
Eight byte field that contains the CRN of the MQ queue resource through which ALCS accesses the MQSeries terminal. In the communications generation, this CRN is provided in the COMDEF **LINK**= parameter. The MQ queue resource is defined in the MQ communication resource type in the ALCS communications generation (COMDEF LDTYPE=MQ).

COMTC ADD considerations:
The CRN of the owning MQ queue resource ***must*** be provided for the MQSeries terminal. ALCS does not provide a default owning MQ queue resource name. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Ensure that the communications definition for the owning MQ queue resource has already been defined in your ALCS communications generation.

COMTC REPLACE considerations:
If the owning MQ queue resource is not being changed, do not modify the value provided by COMTC QUERY. If the CRN of the owning MQ queue resource is being changed, ensure that the communications definition for the owning resource has already been defined in your ALCS system. If the CRN is less than 8 characters, it should be left justified and padded with spaces.

Symbol for the Other System terminal resource type

The following symbol must be used for the Other System terminal resource type (OSYS).

CT1COM
Other system communication ID
One byte field that contains the communication ID of the other system that owns this Other System terminal resource. In the communications generation, this communication ID is provided in the COMDEF **COMID**= parameter. The connection between this ALCS system and the other system will normally be an SNA LU6.1 link. The communication ID is defined for the LU6.1 link in the ALCSLINK communication resource type in the ALCS communications generation (COMDEF LDTYPE=ALCSLINK).

COMTC ADD considerations:
The communication ID of the other system ***must*** be provided for the Other System terminal resource. ALCS does not provide a default communication ID. The communication ID must be an alphabetic character A through Z. Ensure that the other system communication ID has already been defined in your ALCS communications generation.

COMTC REPLACE considerations:
If the other system communication ID is not being changed, do not modify the value provided by COMTC QUERY. If the communication ID of the other system is being changed, ensure it is already defined in your ALCS system. The communication ID must be an alphabetic character A through Z.

Symbols for the TCP/IP ALC terminal resource type

The following symbols can be used for the TCP/IP ALC terminal resource type（TCPIPALC).

CT1CDE

Output message translation
Flag byte that defines how ALCS should translate output messages sent over TCP/IP to this terminal resource. In the communications generation, this is provided in the COMDEF **CODE**= parameter.

Symbol
Meaning if on (1)

CT1CALC
ALCS will translate the output message using the ALC transmission code.

CT1IAT7
ALCS will translate the output message using the IATA7 transmission code.

CT1IAT5
ALCS will translate the output message using the IATA5 transmission code.

CT1NONE
ALCS will not translate the output message.

COMTC ADD considerations:
If you do not specify the translation option for output messages, ALCS will use the default option of translating from EBCDIC to ALC.

COMTC REPLACE considerations:
If the translation option is not being changed, do not modify the value provided by COMTC QUERY. If the translation option is being changed, set the contents of this flag byte using the appropriate symbol. If you set the contents of this flag byte to hexadecimal zero, ALCS will use the default translation option of translating from EBCDIC to ALC.

CT1HEX

H1/H2 address values
Halfword field that contains the H1 and H2 address values for this terminal's address in the SITA network. This field may be required when the terminal communicates with ALCS through a MATIP TCP/IP server resource. The H1 and H2 address values, when combined with the address values A1 and A2 (described below) constitute the address of the ASCU (Agent Set Control Unit or terminal cluster). In the communications generation, the H1 and H2 address values are provided in the COMDEF **HEX**= parameter.

COMTC ADD considerations:
If the H1 and H2 address values are not being provided, this field must contain hexadecimal zeros.

COMTC REPLACE considerations:
If the H1 and H2 address values are not being changed, do not modify the value provided by COMTC QUERY. If the current H1 and H2 address values are being changed, place the new values in this field. If the current H1 and H2 address values are being deleted, set the contents of this field to hexadecimal zero.

CT1TCI

A1 address value
One byte field that contains the A1 address value for this terminal's address in the SITA network. The A1 address value, when combined with the address values H1/H2 (described above) and the A2 address value (described below) constitute the address of the ASCU (Agent Set Control Unit or terminal cluster). In the communications generation, the A1 address value is provided in the COMDEF **TCID**= parameter.

COMTC ADD considerations:
If the terminal is communicating with ALCS through a MATIP TCP/IP server connection, the A1 address value ***must*** be provided for this TCP/IP terminal resource. If the A1 address value is not being provided, this field must contain hexadecimal zero.

COMTC REPLACE considerations:

If the A1 address value is not being changed, do not modify the value provided by COMTC QUERY. If the current A1 address value is being changed, place the new value in this field. If the current A1 address value is being deleted, set the contents of this field to hexadecimal zero.

CT1CIA

A2 address value

One byte field that contains the A2 address value for this terminal's address in the SITA network. The A2 address value, when combined with the address values H1, H2 and A1 (described above) constitute the address of the ASCU (Agent Set Control Unit or terminal cluster). In the communications generation, the A2 address value is provided in the COMDEF **IA**= parameter.

COMTC ADD considerations:
If the terminal is communicating with ALCS through a MATIP TCP/IP server connection, the A2 address value ***must*** be provided for this TCP/IP terminal resource. If the A2 address value is not being provided, this field must contain hexadecimal zero.

COMTC REPLACE considerations:
If the A2 address value is not being changed, do not modify the value provided by COMTC QUERY. If the current A2 address value is being changed, place the new value in this field. If the current A2 address value is being deleted, set the contents of this field to hexadecimal zero.

CT1CTA

Terminal address

One byte field that contains the terminal address. If this terminal resource is attached to an ASCU (Agent Set Control Unit or terminal cluster), the address of the ASCU is provided in the H1, H2, A1 and A2 address values described above. In the communications generation, the terminal address is provided in the COMDEF **TA**= parameter.

COMTC ADD considerations:
If the terminal is communicating with ALCS through a MATIP TCP/IP server connection, the terminal address ***must*** be provided for this TCP/IP terminal resource. If the terminal address is not being provided, this field must contain hexadecimal zero.

COMTC REPLACE considerations:
If the terminal address is not being changed, do not modify the value provided by COMTC QUERY. If the current terminal address is being changed, place the new value in this field. If the current terminal address is being deleted, set the contents of this field to hexadecimal zero.

CT1TCL

Owning TCP/IP server or client

Eight byte field that contains the CRN of the TCP/IP server or client connection through which ALCS accesses this TCP/IP terminal resource. In the communications generation, this CRN is provided in the COMDEF **LINK**= parameter. The TCP/IP server or client connection is defined in the TCPIP communication resource type in the ALCS communications generation (COMDEF LDTYPE=TCPIP).

COMTC ADD considerations:
If the terminal is communicating with ALCS through a MATIP TCP/IP server or client connection, the CRN of the owning TCP/IP resource ***must*** be provided for this TCP/IP terminal resource. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Ensure that the communications definition for the owning TCP/IP server or client connection has already been defined in the ALCS communications generation.

COMTC REPLACE considerations:
If the owning TCP/IP server or client connection is not being changed, do not modify the value provided by COMTC QUERY. If the CRN of the owning TCP/IP server or client connection is being changed, ensure that the communications definition for the owning resource has already been defined in your ALCS system. If the CRN is less than 8 characters, it should be left justified and padded with spaces.

CT1RHT

Gateway IP address

Sixteen byte field that contains the remote IP address of the gateway to which this TCP/IP terminal is attached. In the communications generation, this is provided in the COMDEF **RHOST**= parameter.

The gateway IP address is a string of 1 to 15 characters with dotted decimal notation (for example, 9.190.314.167).

COMTC ADD considerations:
If the IP address is less than 15 characters, it should be left justified and padded with spaces. If the gateway IP address is not being provided, this field must contain hexadecimal zero.

COMTC REPLACE considerations:
If the gateway IP address is not being changed, do not modify the value provided by COMTC QUERY. If the IP address is being changed, it must be less than 15 characters and should be left justified and padded with spaces. If the gateway IP address is being deleted, set the contents of this field to hexadecimal zero.

CT1RH2

Gateway IP address (alternate)
Sixteen byte field that contains an alternate remote IP address of the gateway to which this TCP/IP terminal is attached. In the communications generation, this is provided in the COMDEF **RHOST**= parameter. The gateway IP address is a string of 1 to 15 characters with dotted decimal notation (for example, 9.190.314.167).

COMTC ADD considerations:
If the IP address is less than 15 characters, it should be left justified and padded with spaces. If the IP address is not being provided, this field must contain hexadecimal zero.

COMTC REPLACE considerations:
If the IP address is not being changed, do not modify the value provided by COMTC QUERY. If the IP address is being changed, it must be 15 characters or less, and should be left justified and padded with spaces. If the IP address is being deleted, set the contents of this field to hexadecimal zero.

CT1RH3

Gateway IP address (alternate)
Sixteen byte field that contains an alternate remote IP address of the gateway to which this TCP/IP terminal is attached. In the communications generation, this is provided in the COMDEF **RHOST**= parameter. The gateway IP address is a string of 1 to 15 characters with dotted decimal notation (for example, 9.190.314.167).

COMTC ADD considerations:
If the IP address is less than 15 characters, it should be left justified and padded with spaces. If the IP address is not being provided, this field must contain hexadecimal zero.

COMTC REPLACE considerations:
If the IP address is not being changed, do not modify the value provided by COMTC QUERY. If the IP address is being changed, it must be 15 characters or less, and should be left justified and padded with spaces. If the IP address is being deleted, set the contents of this field to hexadecimal zero.

CT1RH4

Gateway IP address (alternate)
Sixteen byte field that contains an alternate remote IP address of the gateway to which this TCP/IP terminal is attached. In the communications generation, this is provided in the COMDEF **RHOST**= parameter. The gateway IP address is a string of 1 to 15 characters with dotted decimal notation (for example, 9.190.314.167).

COMTC ADD considerations:
If the IP address is less than 15 characters, it should be left justified and padded with spaces. If the IP address is not being provided, this field must contain hexadecimal zero.

COMTC REPLACE considerations:
If the IP address is not being changed, do not modify the value provided by COMTC QUERY. If the IP address is being changed, it must be 15 characters or less, and should be left justified and padded with spaces. If the IP address is being deleted, set the contents of this field to hexadecimal zero.

Symbol for the VTAM 3270 terminal resource type

The following symbol must be used for the VTAM 3270 terminal resource type（VTAM3270).

CT1PRI
Resource priority
One byte field that defines the priority of this resource. In the communications generation, this is provided in the COMDEF **PRI**= parameter. During communication initialization, ALCS acquires the VTAM 3270 resources in priority order, starting with priority zero.

COMTC ADD considerations:
Set the resource priority at a value between zero and 255. If you require ALCS to use the default priority (which is zero) set the content of this field to hexadecimal zero.

COMTC REPLACE considerations:
If the resource priority is not being changed, do not modify the value provided by COMTC QUERY. If the resource priority is being changed, set it at a value between zero and 255.

Symbols for the VTAM/ALCI ALC terminal resource type

The following symbols can be used for the VTAM/ALCI ALC terminal resource type（VTAMALC).

CT1LEI
LEID address
Three byte field that contains the Logical End-point IDentifier (LEID) address for this ALC terminal. In the communications generation, this is provided in the COMDEF **LEID**= parameter. The LEID, which is 6 hexadecimal digits, is the address by which this terminal is known by ALCI.

COMTC ADD considerations:
The LEID address ***must*** be provided for the VTAM/ALCI terminal. ALCS does not provide a default LEID. The LEID address is 6 hexadecimal digits.

COMTC REPLACE considerations:
If the LEID address for the VTAM/ALCI terminal is not being changed, do not modify the address provided by COMTC QUERY. If the LEID address is being changed, this field must contain 6 hexadecimal digits.

CT1NEF
Owning ALCI resource
Eight byte field that contains the CRN of the ALCI logical unit (LU) through which ALCS accesses this ALC terminal. In the communications generation, this CRN is provided in the COMDEF **NEFLU**= parameter. The ALCI LU is defined in the ALCS communications generation by the VTAMALC communication resource type (when the TERM=NEFLU parameter is used).

COMTC ADD considerations:
If the CRN of the owning ALCI resource is less than 8 characters, it should be left justified and padded with spaces. If an owning ALCI resource is not being defined for the ALC terminal, this field should contain spaces or hexadecimal zero. ALCS does not provide a default owning ALCI resource name. Ensure that the communications definition for the owning ALCI resource has already been defined in your ALCS communications generation.

COMTC REPLACE considerations:
If the owning ALCI resource is not being changed, do not modify the value provided by COMTC QUERY. If the CRN of the owning ALCI resource is being changed, ensure that the communications definition for the owning resource has already been defined in your ALCS system. If the CRN is less than 8 characters, it should be left justified and padded with spaces.

Symbol for the WAS terminal resource type

The following symbol must be used for the WAS terminal resource type (WASTERM).

CT1WSL
Owning WAS resource

Eight-byte field that contains the CRN of the WAS resource through which ALCS accesses the WAS terminal. In the communications generation, this CRN is provided in the COMDEF LINK= parameter. The WAS resource is defined in the WAS communication resource type in the ALCS communications generation (`COMDEF LDTYPE=WAS`).

`COMTC ADD` considerations:
The CRN of the owning WAS resource must be provided for the WAS terminal. ALCS does not provide a default owning WAS resource name. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Ensure that the communications definition for the owning WAS resource has already been defined in your ALCS communications generation.

`COMTC REPLACE` considerations:
If the owning WAS resource is not being changed, do not modify the value provided by COMTC QUERY. If the CRN of the owning WAS resource is being changed, ensure that the communications definition for the owning resource has already been defined in your ALCS system. If the CRN is less than 8 characters, it should be left justified and padded with spaces.

The following fields must contain data:

CT1TYP
Communication resource type (WASTERM)

CT1APL
Application name

CT1WSL
CRN of owning WAS resource

CT1TRM
Terminal device type

The following fields are optional:

CT1USD
Communication resource user data

CT1UPD
User data pad character

CT1IST
Initial status

CT1UID
Default user ID

CT1ASD
CRN of associated resource

CT1CRS
Alternate CRAS type

CT1ALT
Alternate CRAS number

CT1FLB
Flag byte for terminal characteristics:

CT1SYSS
Suppress system generated messages

CT1TABC
Tabular skip feature

CT1SCR
ALC screen size

CT1BUF
Printer buffer size

CT1TMO
Printer timeout values

CT1INI
Flag byte for printer buffer and timeout values

Symbols for the X.25 PVC resource type

The following symbols can be used for the X.25 permanent virtual circuit (PVC) resource type（X25PVC).

CT1PRT
X.25 PVC type
Flag byte that defines the type of X.25 permanent virtual circuit. In the communications generation, this is provided in the COMDEF **PRTCOL**= parameter. ALCS supports seven types of PVC, of which three types are used for conversational (Type A) traffic from ALC terminals.

Symbol
Meaning if on (1)

CT1TYP1
This is a Type 1 PVC. This PVC type does not have multiple interchanges addresses or multiple terminal circuit identities.

CT1TYP6
This is a Type 6 PVC. This PVC type includes multiple interchanges addresses (IA's). The interchange addresses are provided in the X.25 ALC terminal definition. The description of the CT1CIA symbol (see below) includes information about the interchange address.

CT1TYP7
This is a Type 7 PVC. This PVC type includes multiple terminal circuit identities (TCID's). The terminal circuit identities are provided in the X.25 ALC terminal definition. The description of the CT1TCI symbol (see below) includes information about the terminal circuit identifier.

COMTC ADD considerations:
You can specify only PVC types that are used for Type A traffic. These are types 1, 6 and 7. If you do not specify the PVC type for this communication resource, ALCS will use Type 1 PVC as the default.

COMTC REPLACE considerations:
The X.25 PVC type can not be changed, therefore do not modify the value provided by COMTC QUERY.

CT1CDE
Output message translation
Flag byte that defines how ALCS should translate output messages sent over X.25 to this terminal resource. In the communications generation, this is provided in the COMDEF **CODE**= parameter.

Symbol
Meaning if on (1)

CT1CALC
ALCS will translate the output message using the ALC transmission code.

CT1IAT7
ALCS will translate the output message using the IATA7 transmission code.

CT1IAT5
ALCS will translate the output message using the IATA5 transmission code.

CT1NONE
ALCS will not translate the output message.

COMTC ADD considerations:
If you do not specify the translation option for output messages, ALCS will use the default option of translating from EBCDIC to ALC.

COMTC REPLACE considerations:
If the translation option is not being changed, do not modify the value provided by COMTC QUERY. If the translation option is being changed, set the contents of the this flag byte using the appropriate

symbol. If you set the contents of this flag byte to hexadecimal zero, ALCS will use the default translation option (EBCDIC to ALC).

CT1PRI
Resource priority
One byte field that defines the priority of this resource. In the communications generation, this is provided in the COMDEF **PRI**= parameter. During communications initialization, ALCS acquires the X.25 PVC resources in priority order, starting with priority zero.

COMTC ADD considerations:
Set the resource priority at a value between zero and 255. If you require ALCS to use the default priority (which is zero) set the content of this field to hexadecimal zero.

COMTC REPLACE considerations:
If the resource priority is not being changed, do not modify the value provided by COMTC QUERY. If the resource priority is being changed, set it at a value between zero and 255.

Symbols for the X.25 ALC terminal resource type

The following symbols can be used for the X.25 ALC terminal resource type (X25ALC).

CT1PVC
Owning X.25 PVC
Eight byte field that contains the CRN of the X.25 permanent virtual circuit (PVC) through which ALCS accesses this ALC terminal. In the communications generation, this CRN is provided in the COMDEF **PVC**= parameter. The ALCS communication definition for the owning X.25 PVC will reside on the OCTM database.

COMTC ADD considerations:
The CRN of the X.25 PVC resource ***must*** be provided for the X.25 ALC terminal resource. ALCS does not provide a default name for the X.25 PVC. If the CRN is less than 8 characters, it should be left justified and padded with spaces. Ensure that the communications definition for the owning X.25 PVC resource has already been defined in the OCTM database.

COMTC REPLACE considerations:
The name of the owning X.25 PVC resource for this ALC terminal can not be changed, therefore do not modify the value provided by COMTC QUERY.

CT1TCI
Terminal circuit identity
One byte field that contains the terminal circuit identity for the terminal controller that this terminal is attached to. In the communications generation, the terminal circuit identity is provided in the COMDEF **TCID**= parameter. This field is required when the X.25 PVC through which ALCS accesses this terminal is defined as a Type 7 PVC. The terminal circuit identity is 2 hexadecimal digits.

COMTC ADD considerations:
This field must contain hexadecimal zero when the owning X.25 PVC is Type 1 or Type 6. When the owning X.25 PVC is Type 7, this field ***must*** contain a terminal circuit identity (ALCS does not provide a default).

COMTC REPLACE considerations:
If the terminal circuit identity is not being changed, do not modify the value provided by COMTC QUERY. If the owning X.25 PVC is Type 7, and the the current terminal circuit identity is being changed, place the new value (as 2 hexadecimal digits) in this field.

CT1CIA
Interchange address
One byte field that contains the interchange address for the terminal controller that this terminal is attached to. In the communications generation, this interchange address is provided in the COMDEF **IA**= parameter. This field is used when the X.25 PVC through which ALCS accesses this terminal is defined as a Type 6 or Type 7 PVC. The interchange address is 2 hexadecimal digits.

COMTC ADD considerations:

This field must contain hexadecimal zero when the X.25 PVC is Type 1. When the X.25 PVC is Type 6 or 7, this field ***must*** contain an interchange address. The interchange address can not be hexadecimal zero and ALCS does not provide a default interchange address. The CRN for this ALC terminal is comprised of a 4-character base CRN followed by an interchange address and a terminal address. When the **CRN**= parameter on the COMTC ADD macro is coded for this ALC terminal, ensure that the full CRN includes this interchange address.

COMTC REPLACE considerations:
The interchange address for this ALC terminal can not be changed, therefore do not modify the value provided by COMTC QUERY.

CT1CTA
Terminal address
One byte field that contains the terminal address of this ALC terminal. In the communications generation, the terminal address is provided in the COMDEF **TA**= parameter. The terminal address is 2 hexadecimal digits.

COMTC ADD considerations:
The terminal address ***must*** be provided for the resource. ALCS does not provide a default terminal address. The terminal address can not be hexadecimal zero.
If the owning X.25 PVC for this ALC terminal is Type 1, the CRN is comprised of a 6-character base CRN followed by this terminal address. If the owning X.25 PVC for this ALC terminal is Type 6 or Type 7, the CRN is comprised of a 4-character base CRN followed by an interchange address (described above) plus this terminal address. When the **CRN**= parameter on the COMTC ADD macro is coded for this ALC terminal, ensure that the full CRN includes this terminal address.

COMTC REPLACE considerations:
The terminal address for this ALC terminal can not be changed, therefore do not modify the value provided by COMTC QUERY.

Register use

Not applicable. CT1TM does not generate executable code.

Loss of control

Not applicable. CT1TM does not generate executable code.

Example

See the section which describes the COMTC QUERY, COMTC ADD and COMTC REPLACE macros.

Summary of data required for each resource type

The CT1TM DSECT provides a large number of fields for accommodating the requirements of the eight resource types supported by OCTM. When this DSECT is used in conjunction with the COMTC macro (for example, COMTC ADD), data should be provided only in those fields that are relevant to the specific resource type that the COMTC macro is being issued for. The following provides a summary of the fields that are relevant to each resource type.

Fields for MQTERM resources

The following fields must contain data:

CT1TYP
Communication resource type (MQTERM)

CT1APL
Application name

CT1MQL
CRN of owning MQ queue resource

CT1TRM
Terminal device type

The following fields are optional:

CT1USD
Communication resource user data

CT1UPD
User data pad character

CT1IST
Initial status

CT1UID
Default user ID

CT1ASD
CRN of associated resource

CT1CRS
Alternate CRAS type

CT1ALT
Alternate CRAS number

CT1FLB
Flag byte for terminal characteristics:

CT1SYSS
Suppress system generated messages

CT1TABC
Tabular skip feature

CT1SCR
ALC screen size

CT1BUF
Printer buffer size

CT1TMO
Printer timeout values

CT1INI
Flag byte for printer buffer and timeout values

Fields for NETVIEW resources

The following fields must contain data:

CT1TYP
Communication resource type (NETVIEW)

CT1APL
Application name

The following fields are optional:

CT1USD
Communication resource user data

CT1UPD
User data pad character

CT1ASD
CRN of associated resource

CT1CRS
Alternate CRAS type

CT1ALT
Alternate CRAS number

CT1TRM
Terminal device type

CT1FLB
Flag byte for terminal characteristics:

CT1FALL
CRAS fallback candidate

CT1SYSS
Suppress system generated messages

CT1RETR
Command retrieve active

Fields for OSYS resources

The following fields must contain data:

CT1TYP
Communication resource type (OSYS)

CT1APL
Application name

CT1TRM
Terminal device type

CT1CSI
Cross system identifier

CT1COM
Other system communication ID

The following fields are optional:

CT1USD
Communication resource user data

CT1UPD
User data pad character

CT1IST
Initial status

CT1UID
Default user ID

CT1ASD
CRN of associated resource

CT1CRS
Alternate CRAS type

CT1ALT
Alternate CRAS number

CT1FLB
Flag byte for terminal characteristics:

CT1SYSS
Suppress system generated messages

CT1TABC
Tabular skip feature

CT1UNSL
Unsolicited light feature

CT1RETR
Command retrieve active

CT1SCR
ALC screen size

CT1BUF
Printer buffer size

CT1INI
Flag byte for printer buffer size

Fields for TCPIPALC resources

The following fields must contain data:

CT1TYP
Communication resource type (TCPIPALC)

CT1APL
Application name

The following fields are optional:

CT1LSM
Inhibit scrolling and send the complete response to this communication resource.

Fields for WASTERM resources

The following fields must contain data:

CT1TYP
Communication resource type (WASTERM)

CT1APL
Application name

CT1WSL
CRN of owning WAS connection resource

CT1TRM
Terminal device type

The following fields are optional:

CT1USD
Communication resource user data

CT1UPD
User data pad character

CT1IST
Initial status

CT1UID
Default user ID

CT1ASD
CRN of associated resource

CT1CRS
Alternate CRAS type

CT1ALT
Alternate CRAS number

CT1FLB
Flag byte for terminal characteristics:

CT1SYSS
Suppress system generated messages

CT1TABC
Tabular skip feature

CT1SCR
ALC screen size

CT1BUF
Printer buffer size

CT1TMO
Printer timeout values

CT1INI
Flag byte for printer buffer and timeout values

CT1LSM
Inhibit scrolling and send the complete response to this communication resource.

CT2TM - Communications Resource and Group Information DSECT

Format

```
CT2TM REG=reg[,SUFFIX=s]
```

REG=*reg*
Base register for DSECT addressability.

SUFFIX=*s*
Suffix for this DSECT, a single alphanumeric character.

Description

Use the CT2TM DSECT to reference fields in the Communications Resource and Group Information data block. When the **COMTC QUERY** macro is used to obtain information from ALCS about a communication resource or a communications group, ALCS provides the information in a Communications Resource and Group Information data block. This data block is 4000-bytes long (an L3 size block) and contains information either for a communication resource or a communications group.

When communication resource information is provided in the CT2TM DSECT, the communication resource can belong to any communication resource type. For many communication resource types, the information is obtained from the Online Communication Table Maintenance (OCTM) database, and for other communication resource types the information is obtained from the ALCS online communication table.

The CT2TM DSECT defines the following symbols for fields within the Communications Resource and Group Information data block.

CT2FLG
Flag byte that identifies the type of data in this block (communication resource data or communications group data) and where the data has originated from (the OCTM database or the online communication table).

Symbol
Meaning if on (1)

CT2FLGA
This data block contains communication resource information obtained from the OCTM database

CT2FLGB
This data block contains communication resource information obtained from the ALCS online communication table

CT2FLGG
This data block contains information about a communications group on the OCTM database

CT2NME
Eight byte field that contains the communication resource name (CRN) of the communication resource or the name of the communications group. If the field contains a CRN, it will be an alphanumeric string of 1 to 8 characters, left justified and padded with spaces. If the field contains a communications group name, it will be an alphanumeric string of 4 to 7 characters, left justified and padded with spaces.

When the block provides information that has been obtained from the OCTM database (for a ***communications resource*** or a ***communications group***) the following fields in the CT2TM DSECT contain data.

CT2TME
Eight byte field that contains the time, in HH.MM.SS character format, when the last status change occurred.

CT2DTE
Eight byte field that contains the date, in YYYY.DDD format, when the last status change occurred.

CT2LTM
Eight byte field that contains the time, in HH.MM.SS character format, when `COMTC LOAD` was issued (if `COMTC LOAD` has not been issued yet, this field contains binary zeros).

CT2LDT
Eight byte field that contains the date, in YYYY.DDD format, when `COMTC LOAD` was issued (if `COMTC LOAD` has not been issued yet, this field contains binary zeros).

CT2BTM
Eight byte field that contains the time, in HH.MM.SS character format, when `COMTC BACKUP` was issued (if `COMTC BACKUP` has not been issued, this field contains binary zeros).

CT2BDT
Eight byte field that contains the date, in YYYY.DDD format, when `COMTC BACKUP` was issued (if `COMTC BACKUP` has not been issued, this field contains binary zeros).

CT2STA
Flag byte that identifies the current status of the communication resource or communications group. If there are no change requests currently active, the flag byte contains binary zeros.

Symbol
Meaning if on (1)

CT2CHNG
Change requests have been received for the communication resource or group, but they are not currently loaded

CT2LOAD
Change requests for the communication resource or group have been loaded by a `COMTC LOAD` macro

CT2BACK
Change requests for the communication resource or group have been backed out by a `COMTC BACKUP` macro

CT2CONF
Change requests for the communication resource or group have been confirmed by a `COMTC CONFIRM` macro

CT2COMM
Change requests for the communication resource or group have been committed by a `COMTC COMMIT` macro

CT2PLDD
Not all the change requests for the communications group have been loaded.

CT2PRG
Flag byte that identifies the current processing status of the communication resource or communications group. If there is no COMTC macro currently being processed, the flag byte contains binary zeros.

Symbol
Meaning if on (1)

CT2CGPR
Change requests are currently being processed for the communication resource or group

CT2LDPR
A COMTC LOAD macro is currently being processed for the communication resource or group

CT2BKPR
A COMTC BACKUP macro is currently being processed for the communication resource or group

CT2CFPR
A COMTC CONFIRM macro is currently being processed for the communication resource or group

CT2CMPR
A COMTC COMMIT macro is currently being processed for the communication resource or group

When the block provides information about a ***communications group***, the following fields in the CT2TM DSECT contain data.

CT2GTM
Eight byte field that contains the time, in HH.MM.SS character format, when this communications group was allocated by a COMTC ALLOCATE macro.

CT2GDT
Eight byte field that contains the date, in YYYY.DDD format, when this communications group was allocated by a COMTC ALLOCATE macro.

CT2CNT
Fullword field that contains a count of communication resources that belong to this communications group. This count will be zero if no change requests have been submitted for the communications group, or the group has been committed via COMTC COMMIT.

CT2LST
Fullword field that contains a displacement from the beginning of this block to an area that contains a list of communication resources that belong to this communications group. If there are no change requests that belong to this group, this field will contain binary zeros. The resource list is comprised of 9-byte items each containing an 8-byte Communication Resource Name (CRN) preceded by a 1-byte change request type. The change request type will identify the type of change request that was submitted for this resource. The first CRN and its change request type are in the first 9 bytes, the second CRN and its change request type are in the next 9 bytes, and so on. The 1-byte change request type is "A" for COMTC ADD, "R" for COMTC REPLACE, and "D" for COMTC DELETE. There is a maximum of 400 CRN's in the resource list.

When the block provides information about a ***communication resource*** that is defined on the OCTM database, the following fields in the CT2TM DSECT contain data.

CT2CRI
Three byte field that contains the Communication Resource Identifier (CRI) for this communication resource.

CT2ORD
Fullword field that contains the communication resource ordinal number for this communication resource.

CT2GRP
Seven byte field that contains the name of the communications group that this communication resource belongs to. If it does not belong to a communications group, this field contains spaces. It will be an alphanumeric string of 4 to 7 characters.

CT2CHG
Flag byte that identifies the type of change request that is currently in progress for this communication resource. If there are no change requests currently active, the flag byte contains binary zeros.

Symbol
Meaning if on (1)

CT2RADD
This is a new communication resource that is being *added* by the `COMTC ADD` macro

CT2RCHG
This is a current communication resource that is being *changed* by the `COMTC REPLACE` macro

CT2RDEL
This is a current communication resource that is being *deleted* by the `COMTC DELETE` macro

CT2BCA
Fullword field that contains a displacement from the beginning of this block to an area that contains the base definition of this communication resource. For example, if this communication resource is currently being changed by `COMTC` macros, this base definition will not include those changes. The changes are provided in the field CT2UCA (see below). Use the `OCTM Communication Resource Definition DSECT, CT1TM` to reference the data in this area.

CT2UCA
Fullword field that contains a displacement from the beginning of this block to an area that contains the changes that are currently in progress for this communication resource. The `COMTC REPLACE` macro is used to change a communication resource. If there are no changes currently being processed for this communication resource, this field contains hexadecimal zeros. Use the `OCTM Communication Resource Definition DSECT, CT1TM` to reference the data in this area.

When the block provides information about a ***communications resource*** whose communication type is `not managed` on the OCTM database (for example, an APPC connection), the resource information will have been obtained from the ALCS online communication table. For these communication resources, the following fields in the CT2TM DSECT contain data.

CT2TYP
Two flag bytes that identify the communication resource type that this communication resource belongs to.

The following symbols are for the first flag byte (CT2TYP+0).

Symbol
Meaning if on (1)

CT2TYPAL
An ALCS application resource type

CT2TYPLK
An SNA LU6.1 communication link resource type

CT2TYPAP
An APPC connection resource type

CT2TYPIP
A TCP/IP connection resource type (client or server)

CT2TYPMQ
An MQSeries queue resource type

CT2TYPSC
A BSC point to point resource type

CT2TYPPV
An X.25 PVC resource type (for PVC types 2 to 5)

CT2TYPNF
An ALCI LU resource type

CT2TYP+1
The following symbols are for the second flag byte (CT2TYP+1).

Symbol
Meaning if on (1)

CT2TYPSL
A Virtual SLC link resource type

CT2TYPTY
A WTTY link resource type

CT2TYPWS
A WAS resource type

CT2TYPVT
A VTAM 3270 resource type (for Prime and RO CRAS only)

CT2CRI
Three byte field that contains the Communication Resource Identifier (CRI) for this communication resource.

CT2ORD
Fullword field that contains the communication resource ordinal number for this communication resource.

Register use

Not applicable. CT2TM does not generate executable code.

Loss of control

Not applicable. CT2TM does not generate executable code.

Example

See the section which describes the `COMTC QUERY` macro.

Related information

“COMTC QUERY - Query status of communication resource or group” on page 594.
“CT1TM - Communication Resource Definition DSECT” on page 631.

CT3TM - Communications Groups Information DSECT

Format

```
        CT3TM REG=reg[,SUFFIX=s]
```

REG=*reg*
Base register for DSECT addressability.

SUFFIX=*s*
Suffix for this DSECT, a single alphanumeric character.

Description

Use the CT3TM DSECT to reference fields in the Communications Groups Information data block. When the **COMTC GROUPS** macro is used to obtain information from ALCS about all of the allocated communications groups, ALCS provides the information in Communications Groups Information data blocks. Each data block is 4000-bytes long (an L3 size block) and contains information on a maximum of 100 allocated communications groups. The first data block is attached to an ECB data level and the remaining blocks are held in L3ST pool records. The number of blocks required for holding information on all the communications groups depends on the number of groups that are currently allocated. ALCS obtains information about each communications group from the Online Communication Table Maintenance (OCTM) database.

The CT3TM DSECT defines the following symbols for fields within the Communications Groups Information data block.

The following are the fields which reside in the record header.

CT3BID
Two byte record identifier. The record ID is hexadecimal AC10 (equate #RIDOCTM in DXCRID)

CT3FCH
Fullword field that contains the file address of a forward chained L3ST pool record. The forward chained record contains information on additional communications groups. When there is no forward chained pool record, this field contains hexadecimal zeros.

CT3CNT
Fullword field that contains a count of the communications groups described in this storage block (or pool record).

CT3TOT
Fullword field that contains the total count of currently allocated communications groups.

The following are the fields which reside in each communications group item.

CT3ITM
This symbol identifies the beginning of a 46 byte communications group item.

CT3NME
Seven byte field that contains the name of the communications group. It will be an alphanumeric string of 4 to 7 characters, left justified and padded with spaces.

CT3RSC
Fullword field that contains a count of communication resources that belong to this communications group. This count will be zero if no change requests have been submitted for the communications group, or the group has been committed via COMTC COMMIT.

CT3GTM
Eight byte field that contains the time, in HH.MM.SS character format, when this communications group was allocated by a COMTC ALLOCATE macro.

CT3GDT
Eight byte field that contains the date, in YYYY.DDD format, when this communications group was allocated by a COMTC ALLOCATE macro.

CT3TME
Eight byte field that contains the time, in HH.MM.SS character format, when the last status change occurred.

CT3DTE
Eight byte field that contains the date, in YYYY.DDD format, when the last status change occurred for this communications group.

CT3STA
Flag byte that identifies the current status of the communications group. If there are no change requests currently active, the flag byte contains binary zeros.

Symbol
Meaning if on (1)

CT3CHNG
Change requests have been received for the communications group, but they are not currently loaded

CT3LOAD
Change requests for the communications group have been loaded by a COMTC LOAD macro

CT3BACK
Change requests for the communications group have been backed out by a COMTC BACKUP macro

CT3CONF
Change requests for the communications group have been confirmed by a COMTC CONFIRM macro

CT3COMM
Change requests for the communications group have been committed by a COMTC COMMIT macro

CT3PLDD
Not all the change requests for the communications group have been loaded.

CT3PRG
Flag byte that identifies the current processing status of the communications group. If there is no COMTC macro currently being processed, the flag byte contains binary zeros.

Symbol
Meaning if on (1)

CT3LDPR
A COMTC LOAD macro is currently being processed for the communications group

CT3BKPR
A COMTC BACKUP macro is currently being processed for the communications group

CT3CFPR
A COMTC CONFIRM macro is currently being processed for the communications group

CT3CMPR
A COMTC COMMIT macro is currently being processed for the communications group

Register use

Not applicable. CT3TM does not generate executable code.

Loss of control

Not applicable. CT3TM does not generate executable code.

Example

See the section which describes the COMTC GROUPS macro.

Related information

"COMTC GROUPS - Obtain status of communications groups" on page 597.

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 "Acronyms and abbreviations" on page 660.
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 (#) [14] 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.

[14] 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 689.
3. Save the file address returned by step “1” on page 689.
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 689.

   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 689).
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 [15] 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.

[15] 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

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Z

Product Number: 5695-068

SH19-6954-23