GC28-1173-01
File No. S370-39

Program Product

MVS/Extended Architecture System Programming Library: TSO

MVS/System Product - JES3 Version 2 5665-291
MVS/System Product - JES2 Version 2 5740-XC6

IBM

Second Edition (November 1985)

This is a reprint of GC28-1173-00 incorporating changes released in Technical Newsletter:

GN28-0882-00 (dated 1 June 1983)

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.

Publications are not stocked at the address given below. Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments has been provided at the back of this publication. If the form has been removed, address comments to IBM Corporation, Dept. D58, Building 920-2, P.O. Box 390, Poughkeepsie, New York 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This publication describes the TSO facilities that can be influenced by the system programmer.

Part 1: TSO Services discusses considerations in preparing for TSO processing, managing data sets needed by TSO, and writing exit routines to extend or modify the operation of TSO.

Part 2: Reference -- TSO Commands describes the TSO commands ACCOUNT and OPERATOR and their associated subcommands.

Associated Publications:

MVS/Extended Architecture System Programming Library: Initialization and Tuning,, GC28-1149

MVS/Extended Architecture System Programming Library: System Modifications, GC28-1152

MVS/Extended Architecture System Programming Library: User Exits, GC28-1147

MVS/Extended Architecture System Programming Library: JES2 Initialization and Tuning, SC23-0065

MVS/Extended Architecture System Programming Library: JES3 Modifications and Macros, SC23-0060

Advanced Communications Function for TCAM, Version 2, Installation Reference, SC30-3133

Data Areas: JES2, LYB8-1191; **JES3,** LYB8-1195

MVS/Extended Architecture TSO Command Language Reference (OS/VS2 TSO Command Language Reference, GC28-0646, as updated by Supplement GD23-0259)

MVS/Extended Architecture System Generation Reference, GC26-4009

MVS/Extended Architecture System Programming Library: JES3 Initialization and Tuning, SC23-0059

MVS/Extended Architecture TSO Guide to Writing a Terminal Monitor Program or a Command Processor, GC29-1295

ACF/VTAM Version 2 Planning and Installation Reference, SC27-0610

IBM System/370 Principles of Operation, GA22-7000

MVS/Extended Architecture Operations: System Commands, GC28-1206

MVS/Extended Architecture Debugging Handbook, Volume 2, LC28-1165

MVS/Extended Architecture Debugging Handbook, Volume 3, LC28-1166

MVS/Extended Architecture Debugging Handbook, Volume 4, LC28-1167

MVS/Extended Architecture Debugging Handbook, Volume 5, LC28-1168

Note:

1. All references to RACF in this publication indicate the program product **Resource Access Control Facility (5740-XXH)**.
2. All references to TSO/E in this publication indicate the program product **TSO Extensions** (5665-293).
3. All references to MVS/System Product in this publication indicate either **MVS/System Product - JES3** (5665-291) or **MVS/System Product - JES2** (5740-XC6).
4. All references to Assembler H in this publication indicate the program product Assembler H Version 2 (5668-962).
5. All references to TCAM in this publication indicate the program product **ACF/TCAM Version 2 Release 4** (5735-RC3).
6. All references to VTAM or TSO/VTAM in this publication indicate the program product **ACF/VTAM Version 2** (5665-280).
7. All references to DFDSS in this publication indicate the program product **Data Facility Data Set Services** (5740-UT3).

Contents

Figures

Summary of Amendments

Summary of Amendments
for GC28-1173-0
as Updated June 1, 1983
by Technical Newsletter GN28-0882

This update reflects technical changes to the SLIP subcommand of OPERATOR.

- The addition of three new keywords NOSUP, NUCMOD, and REASON
- The ability to use the logical operators AND (&) and OR (|) with the DATA parameter

Part 1: TSO Services

This chapter is intended for the programmers responsible for generating and maintaining a system with TSO. The chapter outlines how to do the following things:

- Write the cataloged procedures used by TSO.
- Create, reformat, and maintain the user attribute data set (UADS) and broadcast data set.
- Write an installation exit for the SUBMIT command processor.
- Write an installation exit for the STATUS, OUTPUT, and CANCEL command processors.
- Write a LOGON pre-prompt exit.
- Write an installation exit for the COPY, MOVE, and RENUM subcommands of EDIT.
- Write an exit for an installation-written syntax checker.
- Build translation tables for TSO/VTAM users.
- Use the Global Resource Serialization function.

Preparing for TSO Processing

The steps that are performed by a system programmer after system generation but before a terminal user can log on are listed below. Some of them depend on which access method TSO is to use: the telecommunications access method (TCAM) or the virtual telecommunications access method (VTAM). Others are performed regardless of the access method.

If TCAM is being used, a system programmer does the following:

- Tailors the message control program (MCP) to suit the installation's needs. See **TCAM Installation Reference.**
- Writes the MCP cataloged procedure and includes it in SYS1.PROCLIB.
- Constructs the IKJPRM00 member (or an alternate member) of SYS1.PARMLIB to set terminal I/O coordinator (TIOC) parameters. See the discussion of IKJPRM00 in **System Programming Library: Initialization and Tuning.**

If VTAM is being used, a system programmer does the following:

- Constructs the TSOKEY00 member (or an alternate member) of SYS1.PARMLIB to set VTAM terminal I/O coordinator (VTIOC) parameters. Refer to **System Programming Library: Initialization and Tuning.**
- Builds translation tables to suit the installation's needs.
- Writes the cataloged procedure that starts TSO/VTAM time sharing.
- Writes any editing, attention handling, and nonsupported terminal exit routines that are desired.

Regardless of the access method being used, a system programmer does the following:

- Writes the LOGON cataloged procedure(s) and includes them in a procedure library.
- Includes SYS1.CMDLIB in a LNKLSTxx member of SYS1.PARMLIB or in a LOGON cataloged procedure. For information on LNKLSTxx, see **System Programming Library: Initialization and Tuning.**
- Creates or reformats the UADS and broadcast data sets.

Writing a Message Control Program Cataloged Procedure

The cataloged procedure used to start an MCP specifies the MCP to be started through the PGM= operand of the EXEC statement. The MCP should be named IEDQTCAM; if a name other than IEDQTCAM is specified, the name must be added to the program properties table (PPT) and must be marked nonswappable. The PPT describes the environment that TCAM requires to operate properly. (See "Assigning Special Program Properties" in **System Programming Library: System Modifications.** Also, the EXEC statement must include the DPRTY parameter as DPRTY=(13,9).)

The cataloged procedure used to start the MCP also must define the line addresses dedicated to TCAM. This is done by issuing the LINEGRP macro instruction (used in generating the MCP) to specify the ddname of the DD statements that define the communication lines as data sets. For more information, see **TCAM Installation Reference.**

Figure 1 shows a sample listing of cataloged procedures used to start MCPs.

```
//MCP1       EXEC     PGM=IEDQTCAM,TIME=1440,REGION=128K,DPRTY=(13,9)
//LNGP2741            DD UNIT=021      FIRST LINE GROUP DATA SET 2741
//                    DD UNIT=022
//                    DD UNIT=023
//                    DD UNIT=024
//                    DD UNIT=025
//                    DD UNIT=026
//                    DD UNIT=027
//                    DD UNIT=028
//                    DD UNIT=029
//                    DD UNIT=02A
//LNGPTWX             DD UNIT=02B      SECOND LINE GROUP DATA SET TWX
//                    DD UNIT=02C
//                    DD UNIT=02D
//                    DD UNIT=02E
//                    DD UNIT=02F
//MCP2       EXEC     PGM=IEDQTCAM,TIME=1440,REGION=128K,DPRTY=(13,9)
//DIAL5041            DD UNIT=021      LINE GROUP DATA SET
//                    DD UNIT=022
//                    DD UNIT=023
//                    DD UNIT=024
//                    DD UNIT=025
//                    DD UNIT=026
//                    DD UNIT=027
//                    DD UNIT=028
//                    DD UNIT=029
//                    DD UNIT=02A
```

Figure 1. Sample Listing of MCP Start Procedures

Writing a LOGON Cataloged Procedure

A LOGON cataloged procedure defines the system resources available to a terminal user, and defines or allows for the dynamic allocation of all data sets used by a terminal user. Also, a LOGON procedure specifies which program is to be invoked after LOGON: the terminal monitor program (TMP) distributed with MVS or an installation-written program. A LOGON cataloged procedure can be specified in the PROC operand of the LOGON command, supplied through an installation exit from the LOGON processor, or defined in the entry for the userid in UADS. (If more than one procedure is defined for a userid/password combination, the procedure must be specified on the LOGON command.) If TSO/VTAM is being used, a LOGON cataloged procedure can be specified in the data field of a VTAM logon command, or supplied in a VTAM unformatted system services (USS) definition table or VTAM interpret table. For LOGON procedures to reside in a separate library:

1. Code a PROCxx DD statement for the library in the JES2 or JES3 procedure.
2. For JES2, specify xx in the PROCLIB= parameter in the &TSU initialization parameter.

   For JES3, specify xx in the TSOPROC= parameter on the STANDARDS initialization statement. (For additional information concerning initialization parameters, refer to **System Programming Library: JES2 Initialization and Tuning** or **System Programming Library: JES3 Initialization and Tuning.**)

   LOGON cataloged procedures must reside in the data set defined in the procedure used to start the primary job entry subsystem. This data set may be either SYS1.PROCLIB or a partitioned data set dedicated to LOGON procedures.

Note: During LOGON, the step allocation routine does not wait for an unavailable volume. If this condition arises, the LOGON request fails immediately.

TSO Allocation Suggestions

The following suggestions should improve TSO allocations:

- Data sets the user wants for his TSO sessions should be placed in a LOGON procedure. This technique has these advantages:
  1. Allows volumes to be mounted.
  2. Provides recovery from an offline device condition. Messages tell the operator to 'VARY' the device online.
  3. Saves repeated allocation and freeing of the same data set by successive commands in the same TSO session.
- The DYNAMNBR parameter value in the EXEC should be carefully chosen. The value should be large enough so that it is not readily exceeded by dynamic allocation requests. Note that the maximum number of concurrently allocated resources for any TSO session is 1635.

Note: Although the LOGON will take longer to complete, the overall TSO performance should be better.

Determining TSO User Size

TSO uses the following search order (listed 1 through 5) to obtain the region size that is to be allocated to a TSO user:

Search Order	Effective Region Size
1	LOGON pre-prompt exit can specify the size.
2	Size operand of the LOGON command.
3	UADS size (as specified by the ACCOUNT command).
4	The REGION parameter on the EXEC statement in the user's LOGON procedure.
5	&TSU parameter with subparameter CONVPARM specifying a default region size.

Notes:

- All these effective region sizes are used in conjunction with the IEALIMIT control. (Consult **System Programming Library: System Modifications** for details about IEALIMIT control.)
- The region size value specified for locations 1, 2, and 3 (in the search order) cannot exceed the UADS MAXSIZE, except when a logon pre-prompt exit indicates to LOGON to ignore the UADS MAXSIZE value.
- Once TSO obtains a valid region size value, it stops searching.

EXEC Parameters

The TMP distributed with MVS is named IKJEFT01. If an installation-written TMP is to be used for a particular procedure, its module name must be substituted for IKJEFT01 in the PGM= operand in the EXEC statement. REGION= can be used to limit the amount of virtual storage obtained by variable-length GETMAINs.

DYNAMNBR= is used to calculate the allowed number of concurrent allocations via dynamic allocation. If DYNAMNBR= is omitted, the number of allocations is determined by the number of DD DYNAM statements. If DD DYNAM statements and DYNAMNBR= are both present, the number of concurrent allocations equals the combined total.

The DYNAMNBR parameter value in the EXEC statement should be carefully chosen. The value should be large enough so that it is not readily exceeded by dynamic allocation requests.

Note that the maximum number of concurrently allocated resources for any TSO session is 1635.

If you need job step timing, include the TIME parameter on the EXEC statement in the LOGON procedure.

Any PARM operand on the EXEC statement is interpreted by the terminal monitor program as the first line of input from the terminal. This input could be a command or the implicit execution of a command procedure containing setup commands for the TSO user.

The EXEC statement is the only statement required in a LOGON procedure.

Optional Data Sets

Data sets needed by a processing program such as a compiler or a system utility can be defined dynamically through the ALLOCATE command or through dynamic allocation.

Data sets the user wants for his TSO sessions should be placed in a LOGON procedure. This technique has three advantages:

1. Allows volumes to be mounted.
2. Provides recovery from an offline device condition. Messages tell the operator to VARY the device online.
3. Saves repeated allocation and freeing of the same data set by successive commands in the same TSO session.

Certain DD statements have special meaning and can be included in a LOGON procedure depending upon the installation's needs. They are:

SYSPROC — The SYSPROC DD statement defines the current command procedure library to the TSO EXEC command when the implicit form of the TSO EXEC command is used. The data set described by this DD statement must be partitioned with a record format of V, VB, F or FB. This statement can be defined in the LOGON procedure or can be dynamically allocated using the ALLOCATE command.

Sample listing (when used in LOGON procedure)

```
//SYSPROC DD    DSN=CLIST.PROC.LIB,DISP=SHR
```

Sample terminal input (when used in ALLOCATE command)

```
allocate da('clist.proc.lib')  fi(sysproc)  shr
```

STEPLIB — A LOGON procedure can have a private step library defined by a STEPLIB DD statement. This library can contain command processors or a user-written TMP that the installation wants to make available to selected TSO users. (Note: SYS1.CMDLIB can be specified as a step library. However, it is not recommended; it would nullify the improvements that can be obtained by putting command processors in the LPA.) Most TSO users should not have STEPLIB in their LOGON procedure because of the extra search time required for each command and command procedure.

SYSUADS — The ddname SYSUADS is used by the ACCOUNT facility to access the user attribute data set (UADS). SYSUADS can be allocated in the LOGON procedure or it can be dynamically allocated using the ALLOCATE command. Only those users authorized for the ACCOUNT command should have the SYSUADS DD statement included in their procedure.

Sample Procedure

Figure 2 shows a sample listing of a LOGON cataloged procedure. The sample LOGON procedure can be useful to a programmer using Assembler H. The statements specify the TSO standard TMP for execution; define the library for the users EXEC commands, the work data sets for the assembler, the command procedure library, and the assembler macro library; and specify that SYSIN and SYSPRINT are to be directed to the user's terminal.

```
//AFPROC    EXEC    PGM=IKJEFT01,DYNAMNBR=7
//SYSUT1    DD      DSN=&SYSUT1,UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSUT2    DD      DSN=&SYSUT2,UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSUT3    DD      DSN=&SYSUT3,UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSPROC   DD      DSN=CLIST.PROC.LIB,DISP=SHR
//SYSLIB    DD      DSN=SYS1.MACLIB,DISP=SHR
//SYSIN     DD      TERM=TS
//SYSPRINT  DD      TERM=TS
```

Figure 2. Sample Listing of LOGON Cataloged Procedure

Space Allocation for Utility Work Data Sets Used by EDIT

If a user is editing a new or an existing data set using the MOVE or COPY subcommand, EDIT uses:

1. temporary utility work data sets when the user's profile indicates NORECOVER, when the user is executing commands in the background, or when the user specifies NORECOVER.
2. permanent utility work data sets when the user is authorized to use the EDIT recovery facility and specifies RECOVER on the EDIT command.

For both temporary and permanent utility work data sets, an installation has two options for space allocation:

1. default space allocation
2. controlled space allocation

Default Space Allocation

The following paragraphs explain the algorithms used by EDIT to calculate the default space allocation for utility work data sets.

Editing a New Data Set — A utility work data set is allocated four blocks with a default block size of 4096 bytes for primary space (16384 bytes) and one half that amount (8192 bytes) for secondary space. (If the data set being edited contains 80-character records, the utility data set has a total capacity of approximately 300 records.)

Editing an Existing Data Set — A utility work data set is allocated:

```
2 ((((x + y)b) ÷ 4096)+2) = primary space in 4K blocks
```

where: x = number of records in existing data set

y = additional number of records to be added to the data set (This number is variable and depends on the user specifications on a MOVE or COPY subcommand. The value may be 0.)

b = number of bytes (characters) per record

```
Secondary space = primary-space ÷ 2
```

Example:

- an existing data set contains 6000 records
- 200 additional records are to be copied into the data set
- each record contains 120 characters

```
2 ((((6000 + 200) 120) ÷ 4096) +2) = 366 (primary space:
                                          number of 4K blocks)
    Secondary space =183 (number of 4K blocks)
```

The utility data set has a capacity of approximately 18,600 records.

Controlled Space Allocation

If an installation wants to control the allocation of the utility work data sets and the direct access space used for those data sets, it can preallocate the data sets.

Temporary Utility Work Data Sets — Include in the user's LOGON procedure two DD statements that define the data sets.

```
//SYSEDIT   DD  DSN=&EDIT,UNIT=SYSDA,SPACE=(2048,(20,10))
//SYSEDIT2  DD  DSN=&EDIT2,UNIT=SYSDA,SPACE=(6144,(50,20))
```

- SYSEDIT is a sample ddname for the temporary utility work data set (&EDIT) that is allocated to a user when editing a new data set.
- SYSEDIT2 is a sample ddname for the temporary utility work data set (&EDIT2) that is allocated to a user when editing an existing data set.

Permanent Utility Work Data Sets — Name the data sets

DSN=userid.EDITUTL1
and
DSN=userid.EDITUTL2

Catalog the data sets prior to the edit session. No DD statements are required in the user's logon procedure because the catalog is searched for their location.

- userid.EDITUTL1 is allocated to a user when editing a new data set.
- userid.EDITUTL2 is allocated to a user when editing an existing data set.

Note: The use of the EDIT recovery facility (the specification of RECOVER/NORECOVER) and Permanent Utility Work Data Sets apply only if you have TSO/E installed.

Creating, Reformatting, and Maintaining UADS and Broadcast Data Sets

This section tells the system programmer how to:

- Create the UADS and broadcast data sets
- Reformat the UADS and broadcast data sets to MVS format
- Maintain the UADS and broadcast data sets

Introduction

After system generation, the system programmer must ensure that the UADS and broadcast data sets are available to the system. After the UADS and broadcast data sets are established, authorized individuals can update them by adding, changing, or deleting entries.

Do not place either SYS1.UADS or SYS1.BRODCAST on a shared DASD device that is accessed by more than one CPU, unless you use global resource serialization. (Refer to the paragraph Global Resource Serialization for additional information.)

ACCOUNT Commmand and Its Subcommands

The ACCOUNT command and its subcommands are used to create and update the entries in the UADS. Specifically, the ACCOUNT command can:

- Add new entries or more data to an existing entry (ADD subcommand)
- Delete entries or parts of entries (DELETE subcommand)
- Change data in an entry (CHANGE subcommand)
- Display the contents of an entry (LIST subcommand)
- Display the user identifications for all entries (LISTIDS subcommand)
- Build a new broadcast data set and synchronize it with an existing UADS (SYNC subcommand)
- End operation of the command (END subcommand)
- Find out how to use ACCOUNT subcommands (HELP subcommand)

To use the ACCOUNT command under TSO, a terminal user must be authorized by the installation. The ACCOUNT command and its subcommands are explained in "Part 2: Reference — TSO Commands" in this publication.

To permit creating and maintaining the UADS and broadcast data sets from a batch job, MVS allows the terminal monitor program (TMP) to execute as a batch job that creates a "TSO environment" for ACCOUNT and its subcommands. Thus, the UADS and broadcast data sets can be created or maintained with or without having TSO active.

The UADS Reformatting Program — UADSREFM

Creating a new UADS by reformatting the old UADS before execution is accomplished by a batch job via the UADSREFM program, which operates under the TMP. In addition, the UADSREFM program can eliminate wasted space in the UADS caused by the periodic additions, deletions, and changes to UADS entries. The UADSREFM program reads a member from the old UADS, builds a logical copy of that member, eliminates any inefficient space, and places the newly-formatted member in the new UADS. This process is repeated automatically for each member in the new UADS. The UADSREFM program will not, however, reformat a users entry if the user is currently logged on the system; a message is written for every user that was not reformatted because the userid was in use. You must invoke the UADSREFM program if you change the blocksize of the UADS.

The UADSREFM program is invoked by executing the terminal monitor program and supplying a command, UADSREFM, in the SYSIN stream. Two DD statements are required by the UADSREFM program. A SYSUADN DD statement specifies the input UADS. The SYSUADS DD statement specifies the output UADS, which will be reformatted.

Content and Structure of UADS

The user attribute data set (UADS) is basically a list of terminal users who are authorized use TSO. The UADS contains information about each of the users, and is used to regulate access to the system. There is an entry in the UADS for each terminal user. Each entry consists of the following information:

- A user identification.
- One or more passwords, or a single null field, associated with the user identification.
- One or more account numbers, or a single null field, associated with each password.
- One or more procedure names associated with each account number. Each name identifies a procedure that may be invoked when the user enters the LOGON command.
- The region size requirements for each procedure.
- The name of the group of devices that the procedure will be permitted to use. Data sets located via the catalog are an exception.
- The authority to use or a restriction against using the ACCOUNT command.
- The authority to use or a restriction against using the OPERATOR command.
- The authority to use or a restriction against using the SUBMIT, STATUS, CANCEL, and OUTPUT commands.
- Maximum region size allowed.
- Installation-defined data.
- The authority to specify or a restriction against specifying performance groups during LOGON processing.
- The authority to specify or a restriction against specifying dynamic allocation requests for volume mounting.
- A default destination for SYSOUT data sets (either the system output device or a remote work station).

The organization of the information contained in the UADS is shown in Figure 3, Figure 4 shows the simplest structure that an entry in the UADS can have, and Figure 5 shows a more complex structure.

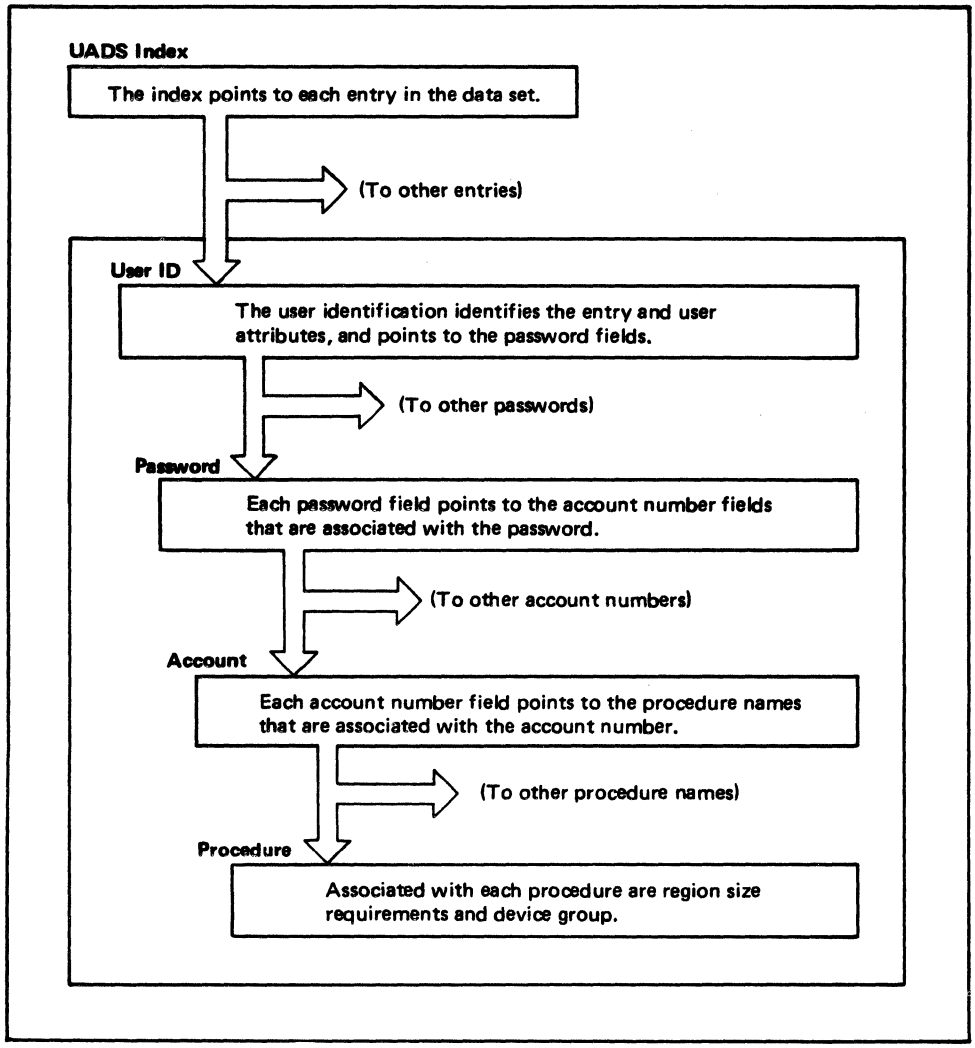


Figure 3. Organization of the UADS

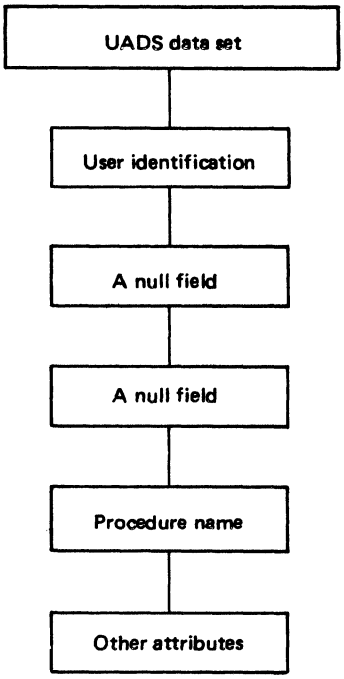


Figure 4. The Simplest Structure for a Typical UADS Entry

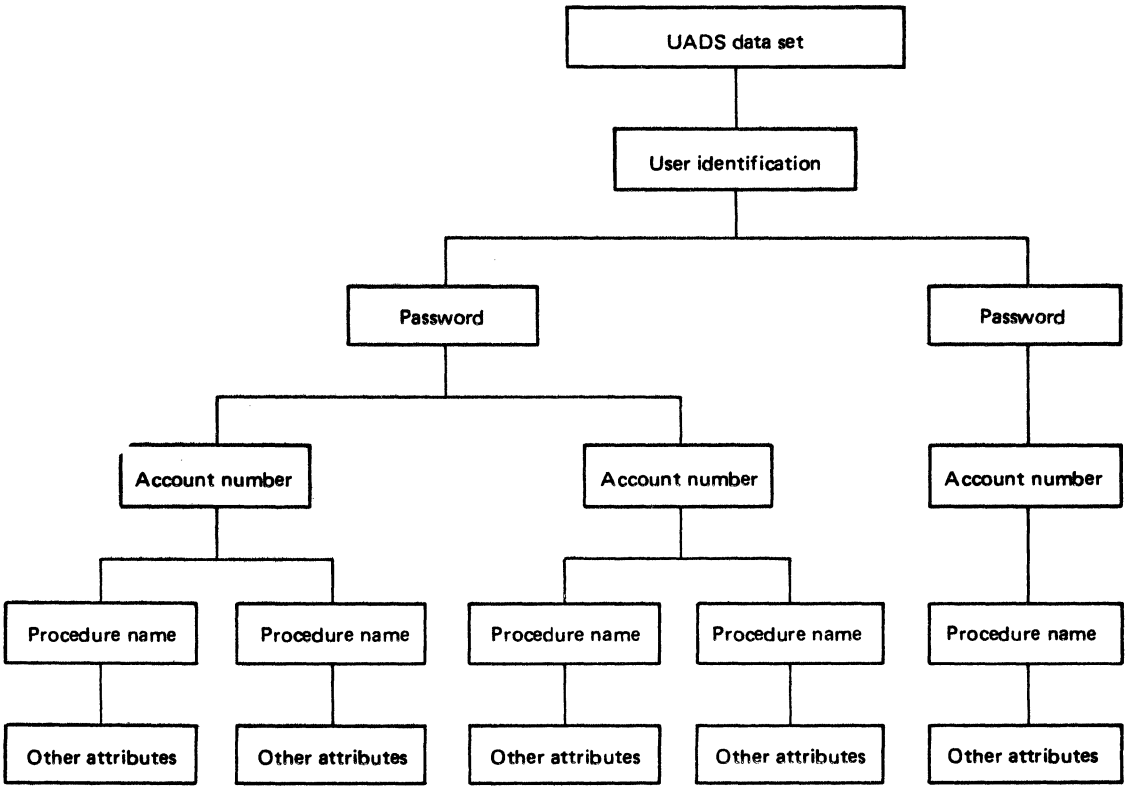


Figure 5. A Complex Structure for a Typical UADS Entry

Creating the UADS and Broadcast Data Sets from a Terminal

To create the UADS and broadcast data sets from a terminal, add to the procedure library a LOGON procedure named IKJACCNT. During system generation, one userid (IBMUSER) is copied into the newly-created UADS. IBMUSER is authorized to use one LOGON procedure, IKJACCNT. A sample IKJACCNT LOGON procedure follows:

```
//IKJACCNT  EXEC    PGM=IKJEFT01,DYNAMNBR=10
//SYSUADS   DD      DSN=new UADS created during sysgen
//SYSLBC    DD      DSN=SYS1.BRODCAST created during sysgen
```

Activate TCAM and initialize the TIOC (see "Initializing Time-Sharing" in this publication).

Log on using the following command:

```
logon ibmuser nonotices nomail
```

The keywords NONOTICES and NOMAIL prevent the LOGON processor from accessing the broadcast data set before broadcast is formatted. Enter the ACCOUNT command and issue the SYNC subcommand to format a skeleton for the broadcast data set. Issue ADD subcommands to add the new userids to both UADS and broadcast.

Log on again with a new userid that has ACCOUNT authority. Enter the ACCOUNT command and issue the DELETE subcommand to delete the IBMUSER userid.

Creating the UADS and Broadcast Data Sets with a Batch Job

You can execute the ACCOUNT facility as a batch job. Use the SYNC subcommand of ACCOUNT to format a skeleton of the broadcast data set. Then, as each userid is added to UADS via the ADD subcommand of ACCOUNT, a corresponding entry is made in the broadcast data set. IBMUSER, a userid with ACCOUNT authority provided during system generation, can be used to create a new UADS. Because a new UADS has been created, IBMUSER should be deleted. Figure 6 is a sample listing showing the creation of the UADS and broadcast data set with a batch job. An explanation of this JCL can be found under "Executing the TMP as a Batch Job in the Background."

```
//jobname   JOB     job card parameters
//          EXEC    PGM=IKJEFT01
//SYSTSPRT  DD      SYSOUT=A
//SYSUADS   DD      DSN=uads created during sysgen
//SYSLBC    DD      DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSIN   DD      *
ACCOUNT
SYNC
ADD new userid (see ADD subcommand of ACCOUNT for other operands)
 .
 .
 .
DELETE (IBMUSER)
END
/*
```

Figure 6. A Sample Listing, Showing the Creation of the UADS and the Broadcast Data Set with a Batch Job

Reformatting the UADS and Broadcast Data Sets

To reformat the UADS and broadcast data sets execute the ACCOUNT facility as a batch job. Use the UADSREFM program to create a reformatted UADS containing the userids from the old format UADS. Use the SYNC subcommand of ACCOUNT to reformat the broadcast data

set with an entry for each userid in the UADS. IBMUSER, a userid with ACCOUNT authority provided during system generation, can be used to reformat the UADS. Because a new UADS has been created, IBMUSER should be deleted. Figure 7 is a sample listing showing the reformatting of UADS and broadcast. An explanation of this JCL can be found under "Executing the TMP as a Batch Job in the Background."

```
//jobname   JOB     job card parameters
//          EXEC    PGM=IKJEFT01
//SYSTSPRT  DD      SYSOUT=A
//SYSUADN   DD      DSN=old format uads
//SYSUADS   DD      DSN=reformatted uads
//SYSLBC    DD      DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSIN   DD      *
UADSREFM
ACCOUNT
DELETE (IBMUSER)
SYNC
END
/*
```

Figure 7. A Sample Listing, Showing the Reformatting of UADS and Broadcast

Maintaining the UADS and Broadcast Data Sets from a Terminal

To maintain UADS and broadcast from a terminal, a terminal user must log on with a userid that is authorized to enter the ACCOUNT command. The terminal user must also ensure that the UADS to be updated is allocated to the SYSUADS DD statement (either in a LOGON procedure or via an ALLOCATE command). The actual changes to UADS are accomplished by issuing the ACCOUNT command followed by its subcommands.

The Terminal Monitor Program

The terminal monitor program (TMP) is attached APF-authorized and executes supervisor state and problem program key. It provides an interface between the user, command processors, and the TSO control program. The TMP is called in one of two ways:

- Via the LOGON procedure for foreground execution
- Via the EXEC statement in the input stream for background execution

Executing the TMP as a Batch Job in the Background

The terminal monitor program (TMP) creates a TSO environment so that the TSO service routines, supported command processors, and the access method services utilities can function in the background. The DD statements SYSIN and SYSPRINT have been replaced by SYSTSIN (for input) and SYSTSPRT (for output) when executing the TMP as a batch job. Input for a GETLINE is obtained from the data set defined by the SYSTSIN DD statement. The TMP issues the STACK macro instruction to put the SYSTSIN data set on the input stack. The commands that are read from SYSTSIN are logged on SYSTSPRT; therefore, the commands and subcommands can be entered via SYSTSIN. Each command must begin on a separate card. Continuation is indicated according to "Continuation Lines" elsewhere in this book.

Messages issued via PUTLINE are written to the data set defined by the SYSTSPRT DD statement. Multilevel informational messages are automatically written out as if you entered a question mark (?). Prompting messages, those that require responses, are not considered informational messages and are not written out. In addition, the "HELP" messages for the prompting messages are not written out.

No prompting is done because the TMP sets options as if the following PROFILE command was issued:

```
profile noprompt
```

Since "no prefixing" of data set names is the default in the background, an unqualified data set name will not be prefixed with a userid. If you want a userid prefixed to the data set names, include the following command:

```
profile prefix(userid)
```

at the beginning of the SYSTSIN stream.

The JCL used to run the TMP as a batch job in the background includes:

1. an EXEC statement that specifies PGM=IKJEFT01. If any command processors will dynamically allocate data sets, specify the DYNAMNBR parameter. If you specify the PARM parameter, its value is interpreted as the first line of input from SYSTSIN.
2. a SYSTSPRT DD statement for commands and subcommands executed, plus messages.
3. a SYSTSIN DD statement for data sets containing commands and subcommands.

Note: The SYSTSIN and SYSTSPRT DD statements may refer to a sequential or partitioned data set. If the data set is partitioned, the member-name must be specified on the DD statement as DSN=pdsname(membername). The SYSTSIN data set cannot be a concatenated data set.

Maintaining the UADS and Broadcast Data Sets with a Batch Job

To maintain the UADS broadcast data sets, use the JCL described previously. In addition, three other DD statements may be used:

1. SYSUADS DD statement specifying the UADS to be accessed by ACCOUNT and UADSREFM.
2. SYSUADN DD statement specifying a UADS to be accessed by UADSREFM.
3. An optional DD statement specifying the broadcast data set to be accessed by ACCOUNT. If a DD statement is not specified, data set SYS1.BRODCAST must be cataloged.

Note: Any job that invokes the TMP in a batch job is given the authorization to execute the ACCOUNT command. For security, the UADS should be password protected or a JCL exit should be written to limit access to the background TMP.

Initializing TSO/TCAM Time-Sharing

Before a terminal user can log on, TCAM must be active in the system and the Terminal I/O Controller (TIOC) must be initialized. The initialization of TIOC completes the initialization for the time-sharing subsystem and allows TCAM to accept LOGON commands and pass them to TIOC for processing.

To start TCAM, the system operator must enter the START command as follows: START TCAM, (where TCAM is the name of a procedure that executes the TCAM MCP).

After TCAM has been started, the system operator must enter the MODIFY command to activate TIOC as a subtask of TCAM:

```
modify tcam,ts=start
```

If a parmlib member other than IKJPRM00 is to be used for TIOC parameters, the member name should be included on the MODIFY command. For example:

```
modify tcam,ts=start,ikjprm01
```

For additional information pertaining to the section on IKJPRM00, see **System Programming Library: Initialization and Tuning.**

To terminate all time-sharing users' connections with the system, the system operator must issue the MODIFY command:

```
modify tcam,ts=stop
```

For more information on START and MODIFY commands, see **System Commands.**

If the users of the TSO EDIT command require the ability to save their data sets with the SAVE subcommand, the installation must ensure that the DASD volume(s) to which the users will save are mounted with the attribute of "STORAGE".

Initializing TSO/VTAM Time Sharing

Before a terminal user can log on to TSO/VTAM time sharing, both VTAM and the terminal control address space (TCAS) must be active in the system.

The system operator must enter the START command to start VTAM. After VTAM has been started, the system operator must enter the START command to activate TCAS. TCAS accepts logons from TSO/VTAM users and creates an address space for each user.

When a user logs on, the VTAM terminal I/O coordinator (VTIOC) is initialized. VTIOC controls the movement of data between TSO and VTAM. Parmlib member TSOKEY00 (or an alternate member) contains parameters that are used during VTIOC initialization. If a member other than TSOKEY00 is to be used, the member name may be included on the START command or in the cataloged procedure invoked by the START command. For a description of TSOKEY00 see **System Programming Library: Initialization and Tuning.**

The system operator uses the MODIFY command to modify TSO/VTAM time sharing. The STOP command is used to stop TSO/VTAM time sharing. For more information on the START, MODIFY, and STOP commands as they pertain to TSO/VTAM time sharing see **System Commands.**

Writing the Procedure That Starts TSO/VTAM Time Sharing

The installation must write a cataloged procedure for starting TSO/VTAM time sharing, and include it either in SYS1.PROCLIB or in an installation-defined procedure library. The cataloged procedure must contain the following statements:

PROC
to name the cataloged procedure and, optionally, to assign default values to symbolic parameters defined in the procedure.

EXEC
to identify the program, IKTCAS00, to be executed.

PARMLIB DD
to identify the parmlib data set and member that contain TSO/VTAM time-sharing parameters. A symbolic parameter can be used for specifying the member name. If it is used, a default value must be specified in the PROC statement. When TSO/VTAM is started, the symbolic parameter receives either the value specified by the system operator on the MEMBER operand of the START command or, if MEMBER is not specified, the default value specified on the PROC statement.

PRINTOUT DD
to identify where the time-sharing parameters that are used should be listed.

A sample procedure for starting TSO/VTAM time sharing is:

```
//TSO       PROC    MBR=TSOKEY00
//STEP1     EXEC    PGM=IKTCAS00,TIME=1440
//PARMLIB   DD      DSN=SYS1.PARMLIB(&MBR),DISP=SHR,FREE=CLOSE
//PRINTOUT  DD      SYSOUT=A,FREE=CLOSE
```

The PROC statement assigns the name TSO to the cataloged procedure, which means that the system operator will enter START TSO to start TSO/VTAM time sharing. The PROC statement also designates a default parmlib member name, TSOKEY00. The EXEC statement specifies that program IKTCAS00 will be executed. It also specifies (TIME=1440) that the execution will not have a time limit. The PARMLIB DD statement identifies the parmlib and member that contain time sharing parameters. Specifying &MBR allows the system operator to use the MEMBER operand of the START command to specify a member name; if MEMBER is not specified, TSOKEY00 will be used. The PARMLIB DD statement also specifies (DISP=SHR) so that the parmlib data set can be used simultaneously by another job, and that it should be deallocated when it is closed (FREE=CLOSE). The PRINTOUT DD statement specifies that the time-sharing parameters used by TSO/VTAM should be written to the device corresponding to class A, and that the output data set should be deallocated when it is closed.

Building Translation Tables for TSO/VTAM Users

Translation tables allow TSO/VTAM users to internally replace characters that are not available on a keyboard with characters that are available. For example, the characters "<", ">", and " | ", which are not available on correspondence keyboards, can be represented by other characters that are available, such as "[", "]", and "!". The CHAR and TRAN operands of the TERMINAL command are used to specify replacement characters. (The TERMINAL command invokes the STTRAN macro instruction to set up the translation tables.)

Specifying character translation causes installation-written translation tables or default (IBM-supplied) translation tables to be used. Default translation tables are a part of the TSO/VTAM programs; they translate each character to itself. The different combinations of the CHAR and TRAN operands that can be specified by users are:

- CHAR (*characters*) alone. This causes a copy of the default translation tables (in the user's storage) to be updated according to the *characters* specified. The updated tables are used to translate all inbound and outbound characters, for as long as the terminal session lasts or until NOCHAR or NOTRAN is specified.
- TRAN (*name*) alone. This causes a copy of the translation tables located in *name* to be used to translate all inbound and outbound characters, for as long as the terminal session lasts or until NOTRAN is specified.
- CHAR (*characters*) and TRAN (*name*). This causes a copy of the translation tables located in *name* to be updated according to the *characters* specified. The updated tables are used to translate all inbound and outbound characters, for as long as the terminal session lasts or until NOCHAR or NOTRAN is specified.

When translation tables are in use, input translations take place after TSO/VTAM translates the line code to EBCDIC characters, and output translations take place before TSO/VTAM translates the EBCDIC characters to line code.

Follow these steps to build translation tables:

1. Code a pair of translation tables, one for input (to TSO) and one for output (to the terminal), with each pair in a control section. Each control section must consist of a fullword containing the address of the output table, followed by a 256-byte EBCDIC

table (on a fullword boundary) for translating the inbound code, followed by a 256-byte EBCDIC table for translating the outbound code. The tables must be formatted according to the rules for the TRANSLATE instruction (refer to **IBM System/370 Principles of Operation**). Translation of numbers and uppercase letters is not allowed.

2. Assemble the translation tables.
3. Link-edit the translation tables into a load module library. One CSECT is allowed per member.
4. Place a JOBLIB DD or STEPLIB DD statement, containing the name of the load module library, into a LOGON cataloged procedure that the user will specify when logging on.

Figure 8 shows translation tables that translate the characters "[" (represented by X'AD'), "]" (represented by X'BD'), and "!" (represented by X'5A') to "<" (represented by X'4C'), ">" (represented by X'6E'), and " | " (represented by X'4F'). (To help find these characters in the example they are printed in bold type.) All other characters are translated to themselves. Assuming these tables are located in a member named TRTAB1, and the name of the data set containing the member was specified in a LOGON cataloged procedure when the user logged on, the user would specify:

```
terminal tran (trtab1)
```

to use them.

```
TRTAB1   CSECT
OUTADR   DC    A(OUTTAB)
INTAB    DS    0F
         DC    X'000102030405060708090A0B0C0D0E0F'    0X
         DC    X'101112131415161718191A1B1C1D1E1F'    1X
         DC    X'202122232425262728292A2B2C2D2E2F'    2X
         DC    X'303132333435363738393A3B3C3D3E3F'    3X
         DC    X'404142434445464748494A4B4C4D4E4F'    4X
         DC    X'505152535455565758594F5B5C5D5E5F'    5X
         DC    X'606162636465666768696A6B6C6D6E6F'    6X
         DC    X'707172737475767778797A7B7C7D7E7F'    7X
         DC    X'808182838485868788898A8B8C8D8E8F'    8X
         DC    X'909192939495969798999A9B9C9D9E9F'    9X
         DC    X'A0A1A2A3A4A5A6A7A8A9AAABAC4CAEAF'    AX
         DC    X'B0B1B2B3B4B5B6B7B8B9BABBBC6EBEBF'    BX
         DC    X'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'    CX
         DC    X'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'    DX
         DC    X'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'    EX
         DC    X'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'    FX
OUTTAB   DS    0F
         DC    X'000102030405060708090A0B0C0D0E0F'    0X
         DC    X'101112131415161718191A1B1C1D1E1F'    1X
         DC    X'202122232425262728292A2B2C2D2E2F'    2X
         DC    X'303132333435363738393A3B3C3D3E3F'    3X
         DC    X'404142434445464748494A4BAD4D4E5A'    4X
         DC    X'505152535455565758595A5B5C5D5E5F'    5X
         DC    X'606162636465666768696A6B6C6DBD6F'    6X
         DC    X'707172737475767778797A7B7C7D7E7F'    7X
         DC    X'808182838485868788898A8B8C8D8E8F'    8X
         DC    X'909192939495969798999A9B9C9D9E9F'    9X
         DC    X'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'    AX
         DC    X'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'    BX
         DC    X'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'    CX
         DC    X'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'    DX
         DC    X'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'    EX
         DC    X'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'    FX
         END
```

Figure 8. An Example of a CSECT Containing Translation Tables

Writing Installation Exits for the SUBMIT Command

The TSO SUBMIT command allows a terminal user to initiate a batch job. The SUBMIT command processor writes the user-specified data set(s), consisting of JCL statements and input data, into a job entry subsystem internal reader. Through an exit routine, an installation can approve, reject, or modify the JCL statements being submitted.

When the first JOB statement is read (or generated), the SUBMIT command processor invokes the exit routine. As a default, only JOB cards (and continuations) are presented to the exit routine. The routine can dynamically indicate additional types of JCL statements that it wishes to inspect as the statements are read from the input data set(s). The routine can delete or change the current statement, and can generate continuation and/or new statements to

follow the current one. The routine can also send a message to the user's terminal and can request a response. In addition, the exit routine can verify jobname and userid, and can cancel a SUBMIT request.

IBM-Supplied Exit Routine

If an installation-written exit routine is not supplied, an IBM-supplied exit receives control. The IBM-supplied exit does not perform JCL checking. It is entered once for each SUBMIT command (JOB statement); it turns off all switches that cause it to receive control, and sets the return code to zero.

Installation-Written Exit Routine

The installation-written SUBMIT exit routine must be link-edited in SYS1.LINKLIB as an independent module named IKJEFF10, with reusable, reenterable, and refreshable link-edit characteristics. The SUBMIT command processor issues LOAD and CALL macro instructions for the exit routine and passes control in key 1 (scheduler key) and in supervisor state for protection purposes. (Thus a terminal user cannot have a revised version of the exit routine in a LOGON procedure's step library.) The exit routine must follow standard linkage conventions.

In the course of SUBMIT command processing, the exit routine may be passed each JCL statement that is submitted. By setting return codes and switches and by passing parameters to the SUBMIT processor, the routine can control the statements passed to it. The return codes and switches also determine subsequent calls to the exit routine. The return codes (to be set in register 15 before the exit routine returns control to the SUBMIT processor) are:

0 Continue (that is, process the current statement and read the next).

4 Reinvoke the exit routine to obtain another statement (that is, process the current statement and invoke the exit routine again so that it can insert a statement).

8 Display message IKJ56283I (message text is supplied by the exit routine) at the terminal and invoke the exit routine. The exit routine must obtain the message text area and may free it when reentered.

12 Display prompting message IKJ56280A (message text is supplied by the exit routine) at the terminal, obtain a response, and invoke the exit routine. (If the user has specified NOPROMPT on a PROFILE command or uses a CLIST without the PROMPT keyword, this will cause the SUBMIT processor to issue a message and terminate the SUBMIT command.) The SUBMIT command processor must obtain and free the reply text area. The exit routine must obtain the message text area and may free it when reentered.

16 Terminate the SUBMIT command. (The exit routine should first use return code 8 to issue a message to the user.)

Undefined return codes cause the SUBMIT processor to issue an error message and terminate.

Upon entry, register 1 contains the address of a pointer to an eight-word parameter list. The system programmer can issue the IKJEFFIE mapping macro instruction to format this parameter list and to assign equates for return codes by coding:

```
ikjeffie ietype=submit
```

As a result, two assembler DSECTs named IEDSECTD and IESUBCTD are created (see **Data Areas**, (microfiche), for the format of each DSECT). After establishing

addressability with each DSECT, the system programmer can refer to the DSECT fields by name. The contents of the parameter list are:

Word 1
: contains the address of the current statement. If word 1 is zero, it indicates a request for the exit routine to supply the next JCL statement (return code from previous call of the exit routine was 4). To delete the current statement, the exit routine must set this word to zero. The exit routine can also change the statement and issue return code 0 or 4 to have the statement processed.

Word 2
: contains the address of a message to be displayed on the user's terminal, supplied in conjunction with return code 8 or 12. If word 2 is non-zero on entry, the return code from the previous call to the exit routine was 8 or 12. The exit routine must obtain the message buffer and may free it when reentered. If word 2 is zero, no message is present (the previous return code was 0 or 4, or this is the first call). The format of the message is "LLtext" , where LL is a two-byte field containing the length of the message text area (including the LL field). The maximum text length is 246 bytes.

Word 3
: contains the address of the response from the terminal user if the exit routine's previous return code was 12. When this field is non-zero after calling the exit routine, the SUBMIT command processor frees the buffer. The format of the response is "LLtext", where LL is a two-byte field containing the length of the reply (including the LL field).

Word 4
: contains the address of userid. The userid is eight characters left-justified, padded with blanks.

Word 5
: contains the address of the control switches, which are contained in a fullword. Byte 0 specifies under what conditions the SUBMIT command processor will call the exit routine:

Byte	Bit	Meaning
0	0	Call exit routine for JOB card
	1	EXEC
	2	DD
	3	Command
	4	Null
	5	/*X in card columns 1-3 (JES2 control card - with /* in columns 1 and 2 and a nonblank character in column 3)
	6	//* in columns 1-3 (comment card or JES3 control card)
	7	//* in columns 1-3 and a nonblank character in column 4 (JES3 control card)

By default, the SUBMIT command processor enters the exit routine for JOB cards only. The exit routine can change the setting of these bits to control when it will be entered. For example, the exit routine can set bit 3 to request control when operator commands (or PROC or PEND statements) are found in the submitted JCL.

Byte 1 (if non-zero) indicates the card column where the operand field begins. For example, if the operand field begins in column 16, byte 1 contains hexadecimal 10. A value in byte 1 is supplied for all statement types.

The operand field is interpreted as follows for JES2 and JES3 control statements and comment statements:

```
/*JES2CTLSTMT OPERAND
//* OPERAND
//*COMMENT OPERAND
//*JES3CTLSTMT OPERAND
```

The first statement is interpreted as a JES2 control statement. (Bit 0 of byte 3 is on.)

The last three statements are interpreted as comment statements or JES3 control statements. (Bit 1 of byte 3 is on.)

Bytes 2 and 3 identify the current statement to the exit routine.

Byte	Bit	Meaning
2	0	JOB statement
	1	EXEC
	2	DD
	3	Command
	4	Null
	5	Operand to be continued
	6	Statement to be continued
	7	Statement is a continuation
3	0	/*X in columns 1-3 (JES2 control card with /* in columns 1 and 2 and a nonblank character in column 3)
	1	//* in columns 1-3 (comment card or JES3 control card)

If bit 3 in byte 2 is on, the current JCL statement has // in columns 1 and 2 but has not been recognized as a JOB, EXEC, or DD statement. The SUBMIT command processor assumes that the // statement is an operator command entered into the input stream.

Continuation is supported for all statement types except comment statements, JES2 control statements, and JES3 control statements. A comma as the last character of the operand, or a nonblank character in column 72 indicates continuation. If a statement has a comma as the last character of the operand, bits 5 and 6 of byte 2 are on. If a statement has a nonblank character in column 72, and the last character of the operand is not a comma, bit 6 of byte 2 is on.

When comment statements appear between continuation statements, all of the following bits are set:

- Continuation (bits 5 and 6 of byte 2, or bit 6 of byte 2 alone)
- Statement type for the statement being continued (one of bits 0 through 4 of byte 2)
- Comment statement (bit 1 of byte 3)

The exit routine is not passed the preceding JCL statements if they are in a DD DATA (or DD *, for /*X cards) input data stream.

Word 6
: reserved for the exit routine's use. The first time the SUBMIT command processor calls the routine, word 6 is initialized to zeros. The routine can use the word for counters or switches. The value is not changed between calls.

Word 7
: contains the address of reconstructed LOGON job accounting information for the user. The SUBMIT command processor inserts this information into generated JOB cards.

Word 8
: contains the address of a halfword that contains the length of the job accounting information.

Note: Normally an installation exit routine issues a message by setting a return code, putting a message address in word 2 of the parameter list, and returning control to SUBMIT command processor. If the exit routine wants to issue a message with a second level via PUTLINE or PUTGET, it must issue a MODESET macro instruction to change to key 0. After issuing the message, the routine must issue MODESET again to change back to key 1.

Writing Installation Exits for the OUTPUT, STATUS, and CANCEL Commands

An installation-written exit routine can control the conditions under which OUTPUT, STATUS, and CANCEL commands will be allowed. The exit routine can restrict a terminal user from obtaining status for a job, from canceling another terminal user's job, or from receiving the output of another terminal user's job. The exit routine can restrict a terminal user from directing a job's output to a specific data set or from changing a job's output class. Also, the exit routine can send a message to the user's terminal and can request a response. In determining whether to allow one of the commands to execute, the exit routine can verify the userid, jobname, and jobid. The exit routine is not entered for a STATUS command with no operands.

IBM-Supplied Exit Routine

If an installation-written exit routine is not supplied, an IBM-supplied exit (common to all three command processors) receives control. The IBM-supplied exit routine allows a terminal user to obtain the status of any job in the system. However, it restricts a terminal user from canceling any jobs other than his own. It also restricts a terminal user from obtaining the output of any jobs other than his own. The IBM-supplied exit routine checks the userid specified on the LOGON command against the jobname or jobnames entered on the CANCEL or OUTPUT command. (The jobname must equal the userid plus at least one character for CANCEL, and must be the userid or must start with the userid for OUTPUT.) If the terminal user enters an invalid jobname, the IBM-supplied exit routine sets up an error message to be issued by the appropriate command processor and tells the command processor via return code to ignore the CANCEL or OUTPUT request for that jobname. If any other jobnames are listed on the same command, the IBM-supplied exit routine processes them in order.

Installation-Written Exit Routine

The installation-written exit routine for OUTPUT, STATUS, and CANCEL commands is also common to all three command processors. The exit routine determines via a command code (in word 7 of the parameter list) which command processor is invoking it. The installation-written routine must be link-edited in SYS1.LINKLIB as an independent module named IKJEFF53, with reusable, reenterable, and refreshable link-edit characteristics. All three command processors issue LOAD and CALL macro instructions for the exit routine and pass control in key 1 (scheduler key) and in supervisor state for protection purposes. (A terminal user cannot have a revised version of the exit routine in a LOGON procedure's step library because only the system link list is used for the LOAD.) The exit routine must follow standard linkage conventions.

After determining the action it wishes to take, the exit routine indicates the action to the appropriate command processor by setting the return code in register 15. The return codes are:

0 Valid jobname; get the next jobname, and continue processing.

4 Display message IKJ56208A (message text is supplied by the exit routine), get a response, and call the exit routine again with the same jobname. If the terminal user has specified NOPROMPT on a PROFILE command or uses a CLIST without the PROMPT keyword, the command processor terminates and issues a message to the terminal.

8 Display message IKJ56208I (message text is supplied by the exit routine) and call the exit routine again with the same jobname. The IBM-supplied exit routine produces the message, "JOB jobname REJECTED - JOBNAME MUST BE YOUR USERID PLUS AT LEAST ONE CHARACTER".

12 Invalid jobname: cancel request for this job, then continue checking any other jobname on the command. The exit routine should first use return code 8 to issue a message.

16 Terminate the CANCEL, OUTPUT, or STATUS command. The exit routine should first issue an error message, using return code 8.

Undefined return codes cause the command processor to issue an error message and terminate.

Upon entry to the installation-written exit routine, register 1 contains the address of a ten-word parameter list. The system programmer can issue the IKJEFFIE mapping macro instruction to format this parameter list. For CANCEL or STATUS, issue:

```
ikjeffie ietype=canst
```

As a result, an assembler DSECT named IEDSECTD is created. For OUTPUT, issue:

```
ikjeffie ietype=output
```

As a result, two assembler DSECTs named IEDSECTD and IEOUTPLD are created (see **Data Areas**, (microfiche), for the format of each DSECT). After establishing addressability with each DSECT, the system programmer can refer to the DSECT fields by name.

The contents of the parameter list are:

Word 1
contains the address of the jobname.

Word 2
contains the address of a halfword that contains the length of the jobname.

Word 3
contains the address of the userid.

Word 4
contains the address of one byte that contains the length of the userid.

Word 5
contains the address of a message to be displayed on the terminal, supplied by the exit routine when the routine specifies return code 4 or 8. The format of a message is "LLtext" where LL is a two-byte field containing the length of the entire message (including the LL field). The maximum text length is 246 bytes. The exit routine must obtain and may free the message text area.

Word 6
contains the address of a response from the terminal user if the exit routine's previous return code was 4. The format of the response is "LLtext" where LL is a two-byte field containing the length of the entire reply (including the LL field). The caller of the exit routine obtains and frees the reply text area.

Word 7
contains the address of the one-byte caller command code. The command codes are:

0 STATUS command
4 CANCEL command
8 OUTPUT command

Word 8
contains the address of the jobid, if jobid was specified on the command. This word is zero if jobid was not specified.

Word 9
contains the address of a halfword that contains the jobid length. The length field is zero if jobid was not specified.

Word 10
(for OUTPUT command) contains the address of a list of pointers and bits reflecting the syntax entered by the terminal user. The total length of this list is five fullwords, with the following contents:

Word 1 —pointer to the first class parse descriptor entry (PDE) on the chain of PDEs. This word is zero if class was not specified on OUTPUT command.

Word 2 —pointer to the print-data-set-name PDE. This word is zero if the data set name was not entered.

Word 3 —pointer to the new class PDE. This word is zero if the new class was not entered.

Word 4 —pointer to the destination PDE. This word is zero if the destination not entered.

Word 5 —only the first one and one-half bytes (high-order) are used to reflect the user-entered syntax as follows:

'8000'	PAUSE (if off, assume NOPAUSE or not applicable)
'4000'	HOLD (if off, assume NOHOLD or not applicable)
'2000'	HERE
'1000'	BEGIN
'0800'	NEXT
'0400'	DELETE
'0200'	PRINT
'0100'	NEWCLASS
'0080'	KEEP (if off, assume NOKEEP or not applicable)
'0040'	DEST
'0020'	Reserved
'0010'	Reserved

The high-order bit of word 10 must be on to indicate the end of the parameter list.

Writing a LOGON Pre-Prompt Exit Routine

The LOGON command initiates a terminal session by supplying the system with certain basic information: userid, password, account number, procedure name, region size, and performance group. An installation can write an exit routine to specify these values for the LOGON command and thereby customize the LOGON procedure for that installation's terminal users. The exit routine can supply system and user attributes for the protected step control block (PSCB), generic group name, performance group, and default SYSOUT destination. It can also provide JCL statements to be used instead of the JOB and EXEC statements constructed by the LOGON Processor.

Note 1: The LOGON processor constructs a standard JOB card and, if SMF audit-exits are being taken, passes the JOB card directly to SMF. If a terminal user must insert installation-dependent information, a LOGON pre-prompt exit is required.

Installation Exit Routine Logic

The installation-written exit routine must be named IKJEFLD and must be link-edited with load module IKJEFLA in SYS1.LPALIB. Consult the output from stage 1 for correct linkage information. Use the system modification program (SMP) to perform this update. When the LOGON processor receives a LOGON command from a terminal user and before it opens the UADS, it passes control to the exit routine as a problem program. Note that when the exit routine is given control, the JSCBPASS bit is set to one to allow access to the UADS for additional user verification. The exit routine can use the I/O service routines through assembler macro instructions (STACK, PUTLINE, GETLINE, PUTGET). The macros require the addresses of the ECT and UPT, two of the parameters passed from LOGON processor.

When the exit routine receives control, register 13 points to a register save area. Register 1 points to a list of addresses, one address for each parameter. (See Figure 9 for the format of the address list. The parameters can be given any name; their meaning is determined by the order of appearance. The names used in the following explanation are for illustrative purposes only.) Four address list entries point directly to the data area. Each of the other entries points to a two-word descriptor: the address of the data area (four bytes), the maximum length of the data area (two bytes), and the current length of the data actually in the data area (two bytes).

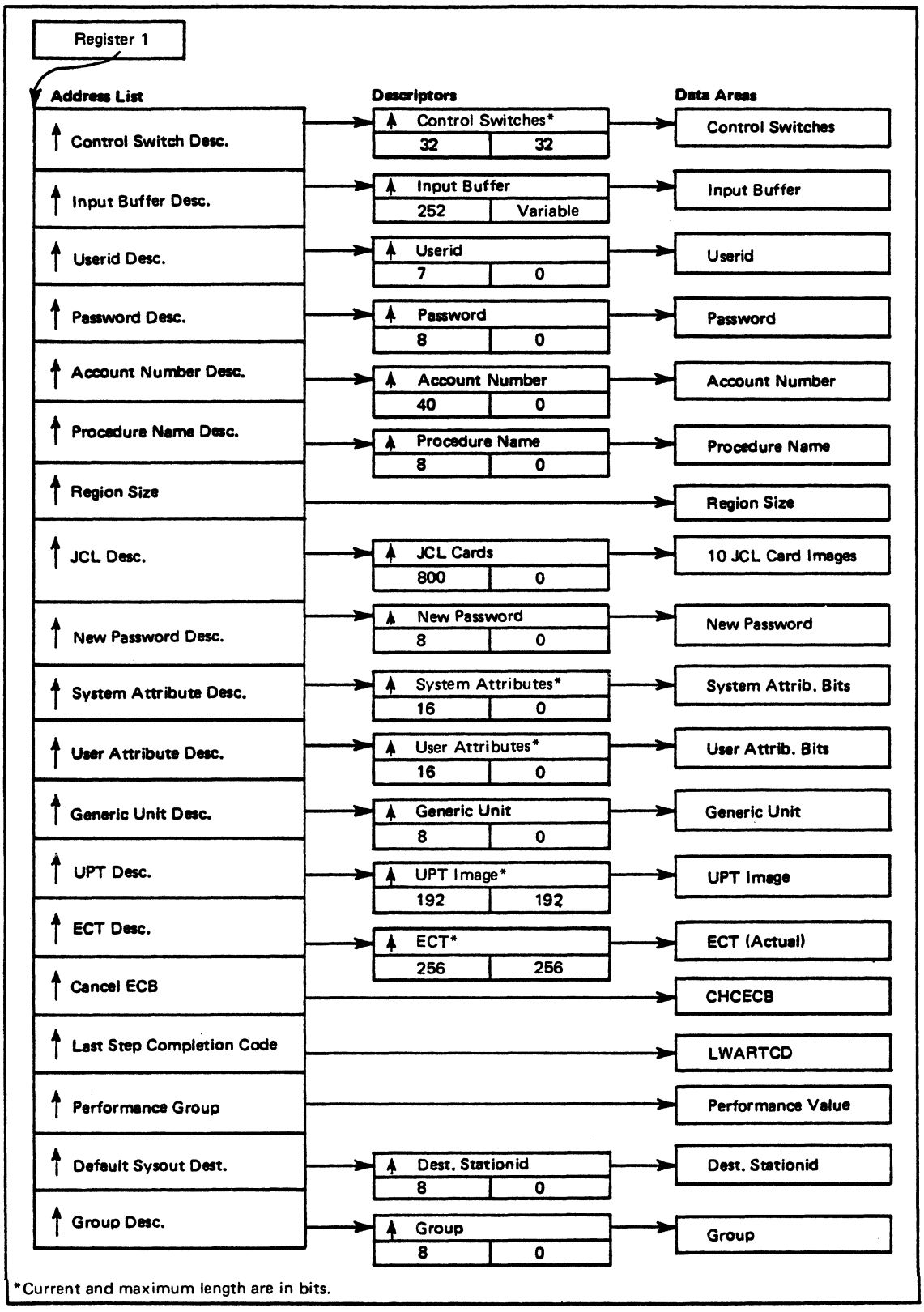


Figure 9. LOGON Pre-Prompt Parameters

The exit routine must move the proper data to the data area, update the current length field, and turn on the appropriate control switch bit, if it exists. The control switch bit indicates to the LOGON processor that the exit routine has passed the corresponding parameter in the data area. For example, if the JCL control switch bit is set to one, the JCL data area must be supplied.

The data address field, the maximum length field, and the addresses in the address list must not be altered. These fields are checked by the LOGON processor and must be the same upon return as they were upon entry. If they are altered, an appropriate error message is issued and the LOGON processor terminates.

Parameter Descriptions

Control Switches: The control switch bits set by IKJEFLD indicate to the LOGON processor that the exit routine has passed the corresponding parameter in the data area or indicate an action to be taken by the LOGON processor. The control switch bits set by the LOGON processor indicate to the exit routine that certain system conditions exist. The bit configuration is as follows:

Byte	Bit	Field Name
0	0	Userid ENQ fail
	1	Reserved
	2	Resource failure
	3	Disconnect
	4	Don't prompt
	5	No UADS
	6	JCL
	7	Reserved
1	0	System attributes
	1	User attributes
	2	Generic group
	3	UPT
	4	Don't ENQ userid
	5	Destination
	6	Abend
	7	Reserved

Notes:

- Bytes 2 and 3 are reserved.
- If the don't-prompt bit is set to one, the values for userid, password, account, procedure, region size, and performance group data areas must be supplied. If the no-UADS bit is set to one also, then system attributes, user attributes, generic group, and UPT data areas must be supplied.
- Maximum and current length are in bits.

Userid ENQ fail
: bit is set to one by the LOGON processor to tell IKJEFLD that the ENQ on the userid was unsuccessful, implying that the userid is in use.

Reserved
: This bit is reserved.

Resource failure
: bit is set to one by the LOGON processor to tell IKJEFLD that the LOGON processor was unable to obtain a resource other than the userid.

Disconnect
: bit is set to one by IKJEFLD to tell the LOGON processor to terminate the session. The LOGON processor sends no further message to the terminal.

Don't prompt
bit is set to one by IKJEFLD to tell the LOGON processor that IKJEFLD has supplied userid, password, procedure name, account number, region size, and, optionally, the performance group. The user cannot be prompted.

No UADS
bit is set to one by IKJEFLD to tell the LOGON processor not to look at the UADS to verify the userid, account number, etc. This bit is ignored if the don't-prompt bit is off.

JCL
bit is set to one by IKJEFLD to tell the LOGON processor that IKJEFLD has supplied a maximum of ten 80-byte JCL cards (in standard format).

System attributes
bit is set to one by IKJEFLD to tell the LOGON processor that IKJEFLD has supplied the system attribute bits.

User attributes
bit is set to one by IKJEFLD to tell the LOGON processor that IKJEFLD has supplied the user attribute bits.

Generic group
bit is set to one by IKJEFLD to tell the LOGON processor that IKJEFLD has supplied a generic group name.

UPT
bit is set to one by IKJEFLD to tell the LOGON processor that IKJEFLD has supplied the UPT.

Don't ENQ
bit is set to one by IKJEFLD to tell the LOGON processor not to ENQ on the userid, implying that more than one user can log on with the same userid. The No-UADS bit must also be set to one for this bit to have any effect.

Destination
set to one by IKJEFLD to tell the LOGON processor that IKJEFLD has supplied the default SYSOUT destination.

Abend
set to one by the LOGON processor's ESTAI retry routine to indicate to IKJEFLD that an abend has occurred and the LOGON processor is retrying one more time.

Input buffer: Contains the text portion of the input line entered from the terminal. It is obtained by the LOGON processor and passed to IKJEFLD. IKJEFLD may alter the input buffer. Maximum and current length are in bytes.

Userid: IKJEFLD passes a userid to the LOGON processor. The don't-prompt bit must be set to one in order for the LOGON processor to accept the userid. Maximum and current length are in bytes.

Password: IKJEFLD returns a password to the LOGON processor. The don't-prompt bit must be set to one in order for the LOGON processor to accept the password. Maximum and current length are in bytes.

Account: IKJEFLD passes an accounting string to the LOGON processor. The don't-prompt bit must be set to one in order for the LOGON processor to accept the accounting string. Maximum and current length are in bytes.

Procedure: IKJEFLD passes to the LOGON processor the name of a cataloged procedure containing JCL to define the resources needed by the terminal job. The don't-prompt bit must be set to one in order for the LOGON processor to accept the procedure name. Maximum and current length are in bytes.

Region size: IKJEFLD passes to the LOGON processor a fullword containing the region size, in hexadecimal, for the terminal job. The don't-prompt bit must be set to one in order for the LOGON Processor to accept the region size.

JCL: IKJEFLD provides JCL statements that define terminal job resources. This JCL is used instead of the JOB and EXEC statements constructed by the LOGON processor. The JCL field is 800 bytes (maximum) in length; therefore, ten card images (maximum) can be passed to the LOGON processor. The LOGON processor expects the JCL data to be in card image; each 80 bytes of the area should contain a full 80-byte JCL card in standard format. Updating the current length field tells the LOGON processor the number of cards being passed (80 x N cards). The JCL bit must be set to one by IKJEFLD for the LOGON processor to accept the JCL supplied. Maximum and current length are in bytes.

New password: For RACF, returns a new password to the LOGON processor, which will replace the current password. The don't-prompt bit must be set to one, in order for the LOGON processor to accept the new password. Maximum and current length are in bytes.

System attributes: IKJEFLD returns a value to the LOGON processor for the PSCBATR1 string in the PSCB. Either the system-attributes bit is set to one or don't-prompt and no-UADS bits are set to one. See **Debugging Handbook, Volume 4**, for a description of the PSCB. Maximum and current length are in bits.

User attributes: IKJEFLD returns a value to the LOGON processor for the PSCBATR2 string in the PSCB. Either the user-attributes bit is set to one or the don't-prompt and no-UADS bits are set to one. See **Debugging Handbook, Volume 4**, for a description of the PSCB. Maximum and current length are in bits.

Generic group: IKJEFLD returns a value to the LOGON processor for the PSCBGPNM field of the PSCB. The group name is initialized from the UADS by the LOGON processor unless the generic-group bit is set to one or the don't-prompt and no-UADS bits are set to one by IKJEFLD. See **Debugging Handbook Volume 4**, for a description of the PSCB. Maximum and current length are in bytes.

UPT: The LOGON processor passes a UPT containing binary zeros to IKJEFLD. This UPT can be updated and returned to the LOGON processor if the UPT bit is set to one or the don't-prompt and no-UADS bits are set to one. See **Debugging Handbook Volume 5,** for a description of the UPT. Maximum and current length are in bits.

ECT: Passed to IKJEFLD, may be modified by IKJEFLD, and will be used by the LOGON processor. This is the actual ECT built by the LOGON processor. IKJEFLD need not set any bit to tell the LOGON processor to accept the ECT. Maximum and current length are in bits.

Cancel ECB: The ECB used by the system for cancel processing. This ECB can be read but not altered by IKJEFLD. If cancel ECB is nonzero, the LOGON processor terminates the present session.

Last step completion: During a re-LOGON, this is a full word containing the completion code for the userid that was just logged off.

Performance group: IKJEFLD passes to the LOGON processor a fullword containing the performance group number, in hexadecimal, for the terminal job. If a performance group of zero is returned to the LOGON processor, then the LOGON processor will use the system default performance group. For further information concerning performance, refer to the **System Programming Library: Initialization and Tuning.**

Default SYSOUT destination: IKJEFLD returns to the LOGON processor the default SYSOUT destination for a userid. Maximum and current length are in bytes.

Group: For RACF, returns a group name to be used as the user's current connect group. The don't-prompt bit must be set to one, in order for the LOGON processor to accept the RACF group identification. Maximum and current length are in bytes.

Sample LOGON Pre-Prompt Exit Routine

A sample LOGON pre-prompt exit is shown in Figure 10. Besides performing housekeeping functions and satisfying linkage conventions, the exit routine supplies two JCL statements (JOB and EXEC) and updates the current length field to indicate that the data area contains 160 bytes of data. The exit routine also sets the JCL control switch bit to one to tell the LOGON processor that the JCL parameter is being passed.

```
IKJEFLD     CSECT
R0      EQU     0
R1      EQU     1
R5      EQU     5
R6      EQU     6
R7      EQU     7
RC      EQU     12
RD      EQU     13
RE      EQU     14
RF      EQU     15
        USING *,RF
        STM   RE,RC,12(RD)          SAVE REGISTERS
        L     R7,28(R1)             INITIALIZE REG SEVEN TO
        L     R6,0(R7)              POINT TO JCL DATA AREA
        MVC   0(80,R6),OURJCL       MOVE JOB CARD
        MVC   80(80,R6),OUREXEC     MOVE EXEC CARD
        MVC   6(2,R7),OURCNT        UPDATE CURRENT LENGTH
        L     R5,0(R1)              INITIALIZE REG FIVE TO
        L     R5,0(R5)              POINT TO CONTROL SWITCHES
        OI    0(R5),X'02'           TURN ON JCL SWITCH
        LM    RE,RC,12(RD)          RESTORE REGISTERS
        BR    14                    EXIT
OURJCL  DC    C'//IBMUSER JOB (D72598,9,OPR),IBM,MSGLEVEL=1,'
        DC    C'MSGCLASS=M,TIME=1439                       '
OUREXEC DC    C'//ST1 EXEC IKJACCNT                         '
        DC    C'                                            '
OURCNT  DC    H'160'          UPDATED CURRENT LENGTH
        END
```

Figure 10. Sample Listing of a LOGON Pre-Prompt Exit

Writing Installation Exits for the RENUM, MOVE and COPY Subcommands of EDIT

The RENUM subcommand of EDIT assigns a line number to each record in a data set that does not have line numbers; it also renumbers each record in a data set that has line numbers. The MOVE subcommand of EDIT moves lines in a data set from one location to another, renumbering the moved lines as necessary. The COPY subcommand of EDIT copies lines in a data set, renumbering the copied lines as necessary.

However, the RENUM, MOVE and COPY subcommand processors cannot handle certain data set types (for example, VSBASIC data sets) that can have embedded line references (using a line number as a destination or statement "label" in a statement that passes control, such as GOTO statements). Thus, an installation may want an exit routine that renumbers, moves, records and copies records in an in-storage data set, adjusting line references as necessary. An exit routine can also be used to flag a statement (for example, by adding a comment at the end of the statement) or to use a non-standard numbering scheme.

IBM-Supplied Exit Routine — VSBASIC Data Set Type

If an installation-written exit routine is not supplied for the VSBASIC program product, an IBM-supplied exit routine (supplied with the VSBASIC program product) receives control. The MOVE and COPY functions are not supported by the IBM-supplied exit. For a RENUM request, the exit routine renumbers the data set's records and scans the data set for statements that contain line references. When it encounters a line reference, the routine updates the reference to reflect the revised line number. Upon successful completion, the exit routine passes return code 0 to the subcommand processor. If it fails, the exit routine issues a message to the terminal user, sets the return code to 8, and returns control to the subcommand processor. The name of the IBM-supplied exit routine defaults to ICDQRNME during system generation.

Installation-Written Exit Routine

The RENUM, MOVE and COPY subcommand processors issue LOAD and CALL macro instructions for the exit routine, which can reside in a step library, in the LPA library, or in any data set in LNKLSTxx. (The exit routine's name must previously have been specified on the EDIT macro instruction during system generation as the DATEXIT value for the applicable data set type(s). The default is to only have a DATEXIT for the VSBASIC data set type.) The exit routine must follow standard linkage conventions. A work area is not passed to the exit routine; it must obtain and release any storage needed. If storage is obtained, it must be in subpool 1. The routine has the use of all TSO service routines in its processing.

The exit routine receives an in-storage data set as input. After processing the data set, the routine must return the data set to the subcommand processor. The subcommand processor copies the data set to a new utility data set to complete the operation.

The routine is expected to periodically check the EDIT command's attention ECB for function interruption. When an interrupt occurs, the exit routine must return control to the subcommand processor with return code 0, after releasing all resources it has obtained.

Upon any completion, the exit routine must release any resources it has obtained. On successful completion, the routine must return a pointer to the updated in-storage data set and its length, if applicable. It must also update the current line pointer (in the subcommand interface area). The exit routine indicates success or failure by the return code that it sets in register 15:

0 indicates successful completion or attention interrupt.

4 requests RENUM processing as if the exit routine were not available.

8 indicates function not performed, message sent to the terminal user.

Upon entry, register 1 contains the address of a parameter list:

Offset (Hex)	Length	Contents
0	4	UPT address
4	4	ECT address
8	4	EDIT attention ECB address
C	1	Flags:
		Bit 0 0 - Records have standard line numbers
		1 - Records do not have standard line numbers
		Bit 1 0 - Not RENUM function
		1 - RENUM function
		Bit 2 0 - Not MOVE function
		1 - MOVE function
		Bit 3 0 - Not COPY function
		1 - COPY function
		Bit 4 0 - Normal line range specification
		1 - '* COUNT' range notation
		Bit 5 0 - Fixed length records
		1 - Variable length records
D	3	Address (in storage) of data set
10	4	Address of subcommand interface
14	4	Address of data attribute parameters

The in-storage data set contains all records in the current EDIT utility data set in the format prescribed by the RECFM applying to the EDIT input data set. All variable record lengths are described by the standard header word (LL/00). Fixed record lengths are described by the data attribute parameters. All records by the data set are back-to-back and are contained in a single area of virtual storage, assigned from subpool 1.

The subcommand interface area, which contains all values in binary, consists of:

RENUM format:

Offset (Hex)	Length	Contents
0	4	Line number position
4	4	Length of line number
8	4	First line to be renumbered (0 indicates first line of data set)
C	4	Last line to be renumbered (X'7FFFFFFF' requests renumbering through end of data set)
10	4	Line number to be assigned to the first renumbered line
14	4	Increment to be used
18	4	Address of current line reference (updated by exit routine before it returns control to the RENUM subcommand processor)

MOVE/COPY format:

Offset (Hex)	Length	Contents
0	4	Line number position
4	4	Length of line number
8	4	First line of MOVE/COPY range (0 indicates first line in data set)
C	4	Last line of MOVE/COPY range (Bit 4 of flags = 1)
10	4	Line number of line preceding insertion point
14	4	Increment to be used
18	4	Current line pointer

The data attribute parameters consist of a two-word list:

Offset (Hex)	Length	Contents
0	4	Logical record length (in bytes)
4	4	Length of the data set (in bytes)

Data Set Types and Syntax Checking

This section tells the system programmer how to:

- Define a data set type
- Write a syntax checker for an installation-defined data set type
- Write an exit routine for an installation-written syntax checker

Data Set Types

An installation can define a data set type for the EDIT command in addition to the data set types supplied by IBM. The installation-defined data set types, along with those supplied by IBM, must be defined during system generation by using the EDIT macro instruction. (For the list of IBM-supplied data set types, see **TSO Command Language Reference.** For more information on how to specify the EDIT macro instruction during system generation, see **System Generation Reference.** The EDIT macro instruction builds a table of constants that describes the data set attributes. The EDIT command processor supports data sets that have the following attributes:

Data set organization	Sequential or partitioned
Record format	Fixed or variable
Logical record size	Less than or equal to 255 characters
Blocksize	User-specified, must be less than or equal to track length
Sequence number	For variable-length records, first eight characters
	For fixed-length records, last eight characters

Syntax Checking

Each IBM-supplied syntax checker is associated with one or more IBM-supplied data set types. If an installation defines its own data set type(s) for the EDIT command, the system programmer can write a syntax checker for each new data set type. Each syntax checker and its associated data set type must be defined during system generation by using the EDIT macro instruction (see **System Generation Reference**). Thus, when a terminal user enters lines of input to an installation-defined data set type, he can request that each line entered in input mode be immediately scanned for syntax errors.

Before a record is scanned by the appropriate syntax checker, the record is put into the terminal user's data set. If a syntax error is found in the record just entered by the user, EDIT displays an error message and switches from input mode to edit mode to enable the user to correct the mistake. The formats of the records passed to the syntax checkers are described in Figure 11.

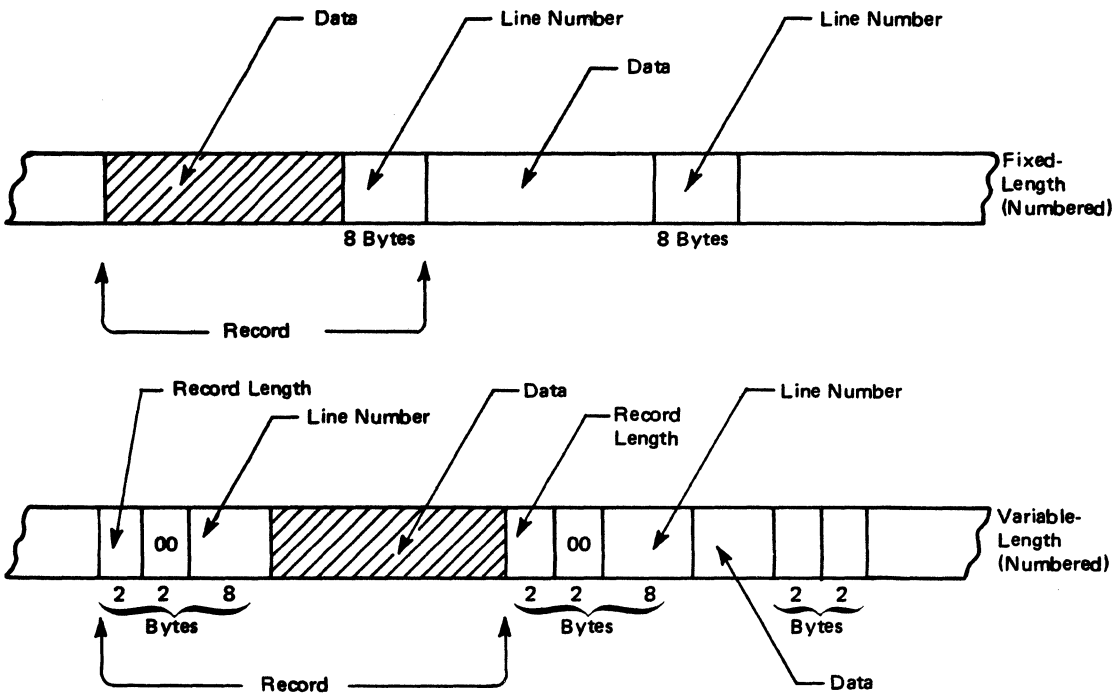


Figure 11. Format of Records Passed to Syntax Checkers

A standard interface is provided to enable the EDIT modules to invoke any available syntax checker. The EDIT modules that invoke syntax checkers, and the IBM-supplied syntax checkers are shown with the standard interface in Figure 12. An installation-written syntax checker can also use this interface, which consists of the syntax checker parameter list.

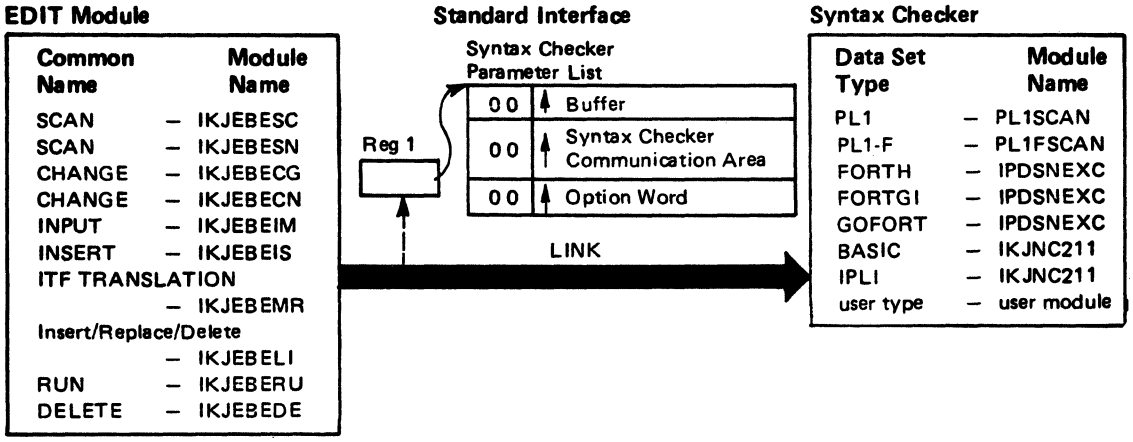


Figure 12. Interface Between EDIT Program and Syntax Checkers

The following figures describe the contents of the control blocks pointed to by the syntax checker parameter list.

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
0	0	C	1	Number of records in buffer (maximum—127); bit zero is set to 1 when the syntax checker has scanned all records in the buffer.
1	1	Chain	3	Address of next buffer; set to zero if this is last buffer in chain.
4	4	Record	Variable	Line or lines of source input data to be syntax checked; can be fixed- or variable-length, numbered or unnumbered.

Figure 13. Contents of the Buffer Control Block

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents	
				Setting	Meaning (Instructions to Syntax Checker)
0	0	None	1	bits 0-3	(where n=0 or 1).
				0nnn	First entry — obtain and initialize work area; if a buffer chain is supplied, the syntax checker will set the relative line number counter to zero.
				1n 1n	Last entry — release the work area and return; syntax checking is not performed.
				1000	Normal entry — set relative line number counter to zero; perform syntax checking.
				110n	Entry after return code 8 (error - buffer checking incomplete) — continue syntax checking.
				1001	Entry after return code 12 (complete statements have been checked, but last statement in input buffer is incomplete): if there is no more input (chain address of last buffer or buffer address is zero), syntax check the incomplete statement and return; if there is a new buffer chain, that is, more input (chain address or buffer address is not zero), resume syntax checking at the incomplete statement.
				bits 4-7	Reserved.
1	1	None	4	xxxx	Address of work area stored by syntax checker on first entry.
4	4	None	4	xxxx	Initial entry — maximum statement size specified at SYSGEN (if 0, checker assumes sufficient storage for largest legal statement is available); entry after return code 4 (error detected, syntax checking complete, second-level message present), or 8 (error detected, syntax checking incomplete) — address of error message area.
8	8	None	4	xxxx	Initial entry — Temporary work area; subsequent entries — address of second error message, if any.
12	C	None	4	xxxx	Temporary storage area used for GETMAIN.

Figure 14. Contents of the Syntax Checker Communication Area

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents		
				Setting	Meaning	Syntax Checker
0	0	None	1	X'00'	FORTRAN H level	FORTRAN
				X'03'	GOFORT	FORTRAN
				X'04'	FORTRAN G1	FORTRAN
				X'00'	IPLI level	IPLI
				X'01'	BASIC level	BASIC
				xxxx	Value of left source margin	PLIF
1	1	None	1	bits 0-5	Reserved	FORTRAN
				bit 6=1	FORTRAN G1/H Code and Go definition to be loaded on initial entry	FORTRAN
				=0	FORTRAN G1/H Code and Go definition not to be loaded on initial entry	FORTRAN
				bit 0=1	Entry from INPUT, Insert, linenum, *, CHANGE	IPLI or BASIC
				bit 1=1	Entry from DELETE	IPLI or BASIC
				bit 2=1	Entry from MERGE or RENUM	IPLI or BASIC
				bit 3=1	Translation already complete	IPLI or BASIC
				bit 4=1	Entry from RUN	IPLI or BASIC
				bit 5-7	Reserved	IPLI or BASIC
				xxxx	Value of right source margin	PLIF
2	2	None	1	xxxx	Record length of fixed-length records; binary zero, if variable-length records.	All
3	3	None	1	bit 0=0	CHAR 60	PLI or IPLI
				=1	CHAR 48	PLI or IPLI
				bit 1=0	Line-numbered data set	All
				=1	Data set not line-numbered	All
				bit 2	Reserved	All
				bit 3=0	Diagnose an incomplete statement	All
				=1	Delayed scan — return with code of 12 if last statement in input buffer is incomplete; immediate scan — possible incomplete statement in buffer.	All
				bit 4=0	Fixed-length records	All
				=1	Variable-length records	All
				bit 5=0	Standard form source input	All
				=1	Free form source input	All
				bit 6=0		
				=1		
				bit 7=0	SCAN or SCAN ON specified	All
				=1	NOSCAN or SCAN OFF specified	All

Figure 15. Contents of the Option Word

Writing Installation Exits for Syntax Checkers

For IBM-supplied data set types and associated syntax checkers, each syntax checker determines the attributes of the associated data set type by referring to information that EDIT initialization sets in the option word, a fullword in the syntax checker parameter list. However, for an installation's own data set type and syntax checker, EDIT initialization does not place the attribute information in the option word; the system programmer can write an exit routine to fill in the option word for the syntax checker according to information entered by a terminal user.

When a terminal user specifies the installation's data-set-type keyword on the EDIT command, he can also specify a subfield. The subfield can contain any alphameric data defined as valid, not exceeding 256 characters and not containing any blanks, tabulation characters, or commas. This information is passed to the exit routine to be interpreted and encoded into bytes 0 and 1 of the option word.

The exit routine name must be supplied during system generation as the USREXT value for the applicable data set type(s) on the EDIT macro instruction. The routine receives control from the EDIT command processor via a LINK macro instruction, and it must follow standard linkage conventions.

Upon entry, register 1 points to a three-word parameter list. The contents of the parameter list are:

Word 1

Address of the subfield parameter descriptor element (PDE) with the following format:

Bytes	Meaning
4	Address of the character string (zero, if the character string is omitted)
2	Length of the character string
1	Flag bits: High-order bit set to 0, the parameter is not present. High-order bit set to 1, the parameter is present. The other bits are unused.
1	Reserved

For more information on this PDE, see **TSO Guide to Writing a TMP or CP** for a description of the IKJIDENT PDE under "Format of the PDES Returned by Parse."

Word 2

Address of bytes 0 and 1 of the syntax checker option word. The installation-written syntax checker can assign its own meanings for the bit settings.

Word 3

Address of the command processor parameter list (CPPL) passed to the EDIT command processor from TMP. The format of the CPPL is:

Bytes	Field name	Meaning
4	CPPLCBUF	The address of the command buffer.
4	CPPLUPT	The address of the user profile table (UPT).
4	CPPLPSCB	The address of the protected step control block (PSCB).
4	CPPLECT	The address of the environment control table (ECT).

The exit routine uses this information to access the ECT and UPT to invoke Parse or a TMP service routine.

The exit routine is expected to update bytes 0 and 1 of the option word and to issue a return code: 0 if successful, 4 if unsuccessful.

Adding Subcommands to the EDIT Command

When a terminal user enters an EDIT subcommand, the Edit controller (load module IKJEBEMA) invokes the appropriate subcommand processor. IKJEBMA9, a CSECT within load module IKJEBEMA, is reserved as a table of installation-written EDIT subcommand names. Thus, if an installation requires an editing function not provided by the IBM-supplied EDIT subcommands, the system programmer can write one or more subcommand processors and add their names to this table by using the IKJEBEST macro instruction. The operands of the IKJEBEST macro instruction indicate the name of the subcommand, the abbreviation for the subcommand name, the name of the module that processes the subcommand, and the CSECT=USER operand. For example:

```
IKJEBEST  (SWTCH,SW,XXXSWTCH),CSECT=USER
```

To specify multiple subcommand processors via IKJEBEST, the format is:

```
IKJEBEST    (subcmd1,abbr1,mod-name1)[,...],CSECT=USER
```

Note: The names associated with an installation-written subcommand must not be the same as the names associated with any IBM-supplied subcommands.

When the IKJEBEST macro instruction is assembled, the CSECT=USER operand causes the resulting object module to be named IKJEBMA9. The system programmer must link-edit the IKJEBMA9 module with the IKJEBEMA load module, performing a CSECT-replacement operation for IKJEBMA9 object module. At that point, the installation-written EDIT subcommands are available.

Macro IKJEBEST can be obtained from PVTMACS or included in your macro library by coding the following:

```
            MACRO
            IKJEBEST     &CSECT=IBM
            LCLA      &A,&B,&C,&D,&E
            LCLA      &F
            LCLC      &CNAME,&SCNAME,&ABBR,&LDMOD,&LABEL,&LABEL1,&LABEL2,&
            AIF       ('&CSECT' NE 'IBM').CONT0
&CNAME      SETC      'IKJEBMA8'      DEFINE CSECT NAME FOR IBM TABLE.
IKJEBMA8    CSECT
            ENTRY     MA8IP002
            ENTRY     MA8LI002
            AGO       .CONT1
.CONT0      ANOP
            AIF       ('&CSECT' NE 'USER').ERROR2
&CNAME      SETC      'IKJEBMA9'      DEFINE CSECT NAME FOR USER TABLE.
IKJEBMA9    CSECT
.CONT1      ANOP
&A          SETA      N'&SYSLIST
            AIF       (&A EQ 0).END
&B          SETA      1
&F          SETA      1
.CONT2      ANOP
&C          SETA      N'&SYSLIST(&B)
            AIF       (&C LT 2 OR &C GT 3).ERROR1
&E          SETA      K'&SYSLIST(&B,&C)
&D          SETA      &E-1
.*  THE FOLLOWING FLAGGED INSTRUCTIONS WERE ADDED TO PROVIDE
.*  UNIQUE LABELS, EVEN IF MODULES HAVE IDENTICAL LAST TWO
.*  CHARACTERS IN ENTRY POINT NAMES.  THE LABELS FOR MODULES
.*  IKJEBELI AND IKJEBEIP ARE UNCHANGED. SINCE THEY ARE
.*  REFERENCED WITHIN IKJEBEMA.
            AIF       ('&CSECT' NE 'IBM').CONT10
            AIF       ('&SYSLIST(&B,&C)'(&D,&E) EQ 'LI'OR     X
                       '&SYSLIST(&B,&C)'(&D,&E) EQ 'IP' ).CONT11
.CONT10     ANOP
&LABEL1     SETC      '&CNAME'(6,8).'@'.'&F'
&F          SETA      &F+1
&LABEL2     SETC      '&CNAME'(6,8).'@'.'&F'
&F          SETA      &F+1
            AGO       .CONT12
.CONT11     ANOP
&LABEL1     SETC      '&CNAME'(6,8).'&SYSLIST(&B,&C)'(&D,&E).'001'
&LABEL2     SETC      '&CNAME'(6,8).'&SYSLIST(&B,&C)'(&D,&E).'002'
.CONT12     ANOP
&SCNAME     SETC      '&SYSLIST(&B,1)'
            SPACE 2
            DC        AL1(&LABEL1-*-1) LENGTH OF SUBCOMMAND NAME.
            DC        C'&SCNAME' SUBCOMMAND NAME.
&LABEL1     EQU       *
            DC        AL1(&LABEL2-*-1) LENGTH OF ABBREVIATION.
            AIF       (K'&SYSLIST(&B,2) EQ 0).CONT5
&ABBR       SETC      '&SYSLIST(&B,2)'
            DC        C'&ABBR' ABBREVIATION FOR SUBCOMMAND.
.CONT5      ANOP
&LABEL2     EQU       *
&LDMOD      SETC      '&SYSLIST(&B,&C)'
            DC        CL8'&LDMOD' LOAD MODULE NAME.
            AIF       (&B EQ &A).END
&B          SETA      &B+1
            AGO       .CONT2
.END        ANOP
            SPACE     2
            DC        AL1(255)        END OF TABLE MARKER.
            MEXIT
.ERROR1     MNOTE     12,'INVALID TABLE ENTRY'
            MEXIT
.ERROR2     MNOTE     12,'INVALID KEYWORD VALUE'
            MEND
```

Authorized Program Execution

If suitable installation access lists in CSECTs IKJEFTE2 and IKJEFTE8 are link edited with the terminal monitor program, then TSO users can execute authorized and nonauthorized programs within a single TSO session.

If the installation does not supply names in lists APFCTABL (in IKJEFTE2) and APFPTABL (in IKJEFTE8), then only nonauthorized programs can be executed. This restriction on execution of authorized programs includes the utility programs IEBCOPY, IEHMOVE, IEHATLAS, IEHINITT, and IEHPROGM; and, additionally, DFDSS. (However, TSO users can execute those programs by using the SUBMIT command.)

The IBM-supplied lists for APFCTABL and APFPTABL contain blank entries which inhibit the execution of APF-authorized programs. The APFCTABL list contains the names of authorized command processors executed by the TMP, and the APFPTABL list contains the names of authorized programs to be executed by CALL. The modules that are attached for these names must be link-edited with APF authorization. If a name does not appear in these lists, the program is attached without authorization. If a program is to be executed by both the TMP and CALL, then its name must appear in both lists.

The format of the list is a sequence of eight-character command name entries. This list is terminated by an entry consisting of eight blanks. Command name entries of less than eight characters must be left-justified and padded to the right with blanks to fill the eight-character entry.

The first entry to be examined by the TMP in either IKJEFTE2 or IKJEFTE8 will be that entry associated with the respective ENTRY name APFCTABL or APFPTABL. If a command has an abbreviation, it must appear as a separate entry. A null list consists of just the final eight blanks.

For example:

If commands R1USER with abbreviation R1 and P3SRCH are to be executed with authorization, then the list should look like:

```
          ENTRY     APFCTABL
IKJEFTE2  CSECT
          DC        CL8'IKJEFTE2'
          DC        CL8'  76.133'   DATE MAY CHANGE
APFCTABL  DC        CL8'R1USER  '
          DC        CL8'R1      '
          DC        CL8'P3SRCH  '
          DC        CL8'        '
          END
```

If an installation wishes to allow access to IEBCOPY through CALL, then the list should look like:

```
          ENTRY     APFPTABL
IKJEFTE8  CSECT
          DC        CL8'IKJEFTE8'
          DC        CL8'  76,133'   DATE MAY CHANGE
APFPTABL  DC        CL8'IEBCOPY '
          DC        CL8'        '
          END
```

The lists in APFCTABL and APFPTABL must contain only the eight-character strings. The installation can reserve extra space by additional terminal blank strings. Nonblank entries following a blank entry are not examined.

IBM modules IKJEFTE2 and IKJEFTE8 can be replaced by link-editing user modules with these names into TMP load module IKJEFT02 in SYS1.LPALIB. Consult the output from stage 1 for correct link edit information. Any program which depends upon a job step environment such as the TMP should not be placed in the lists.

Global Resource Serialization

An installation can place a single version of both SYS1.UADS and SYS1.BRODCAST on a shared DASD and access each one from any system in a multi-system complex by using global resource serialization (that is, the resources - SYS1.UADS and SYS1.BRODCAST - may be globally shared). However, to ensure that an installation can evaluate the applicability of global resource serialization in their TSO environment before using it, the minor names (SYS1.UADS and SYS1.BRODCAST) of both data sets are included in the default SYSTEMS exclusion resource name list, as distributed by IBM (that is, the resources are *excluded* from global sharing). In the process of evaluation, an installation should also do advance planning to investigate and measure:

- resource requirements (e.g., that required to merge multiple versions of the two data sets into a single version of each and test the new versions)
- performance implications (e.g., one version of each data set accessed by all users versus n versions of the same data sets each accessed by a subset of those users)

If an installation decides to use global resource serialization to allow global sharing of SYS1.UADS and SYS1.BRODCAST, it should be aware of the following considerations.

When the SEND command processor writes a new record to SYS1.BRODCAST, it does a sequential directory search when searching for the mail chain for a particular user. An exclusive ENQ is issued for each directory record. After the ENQ is issued, an I/O request is issued to read the directory record, which is then scanned for the entry for a particular user. If there is no entry for the user, a DEQ is issued. The process of ENQ/READ/DEQ of directory records is repeated until an entry is found for the user.

The number of records in the directory is a function of the number of users defined in the UADS. Therefore, the number of ENQ/READ/DEQ operations involved may be significant.

LISTBC does a sequential directory search when searching for the mail chain for a particular user and uses the same ENQ/READ/DEQ process as SEND to find the entry for the user. Therefore, the time to process a LISTBC command (or to complete a LOGON) could be increased.

Advantages:

1. Only two data sets to maintain; rather than 2n where n is the number of systems in a complex.
2. A user can logon from any system in a complex to allow a better workload balance.
3. For foreground initiated background jobs, a user who specifies NOTIFY will always receive the job-ended message regardless of which system in a complex processed the job.

Requirements:

1. Merge all existing versions of SYS1.UADS and SYS1.BRODCAST into a single version of each data set.
2. Modify the resource name lists, as distributed by IBM, as follows:
   a. Delete the minor names SYS1.UADS and SYS1.BRODCAST from the distributed default SYSTEMS exclusion resource name list.
   b. Add the major name SYSIKJUA as a generic entry in the SYSTEM inclusion resource name list (for SYS1.UADS sharing).
   c. Add the major name SYSIKJBC as a generic entry in the SYSTEM inclusion resource name list (for SYS1.BRODCAST sharing).

(Refer to **System Programming Library: User Exits** for information regarding the contents of, and how to modify, the resource name lists.)

Part 2: Reference — TSO Commands

This section describes the two TSO commands, ACCOUNT and OPERATOR, that should be totally restricted in use to system programmers and installation-approved personnel. These commands are completely described in this book.

Coding the Commands

The notation used to define the command syntax and format in this publication is described in the following paragraphs.

1. The set of symbols listed below is used to define the syntax, but never use them in the actual command.

hyphen	-
underscore	_
braces	
brackets	[ ]
ellipsis	...

   The special uses of these symbols are explained in the following paragraphs.

2. Use uppercase letters, numbers, and the set of symbols listed below in an actual command exactly as shown in the command syntax.

apostrophe	'
asterisk	*
comma	,
equal sign	=
parentheses	()
period	

3. Lowercase letters, and symbols appearing in a command syntax represent variables for which you substitute specific information in the actual command.

   Example: If *name* appears in a command syntax, substitute a specific value (for example, ALPHA) for the variable when you enter the command.

4. Hyphens join lower-case words and symbols to form a single variable.

   Example: If *member-name* appears in the command syntax, substitute a specific value (for example, BETA) for the variable in the actual command.

5. An underscore indicates a default option. If you select an underscored alternative, you need not specify it when you enter the command.

   Example: The representation

   $$\underline{\begin{matrix}A\\B\end{matrix}}\\C$$

   indicates select A or B or C; however, if you select B, you need not specify it because it is the default option.

6. Braces group related items, such as alternatives.

   Example: The representation

   ALPHA=({ A / B / C } ,D)

   indicates choose one of the items enclosed within the braces. If you select A, specify ALPHA=(A,D).

7. Brackets also group related items; however, everything within the brackets is optional and may be omitted.

   Example: The representation

   ALPHA=([A / B / C] ,D)

   indicates choose one of the items enclosed within the brackets or omit all of the items within the brackets. If you select only D, specify ALPHA=(,D).

8. An ellipsis indicates that the preceding item or group of items can be repeated more than once in succession.

   Example:

   ```
   ALPHA[,BETA]...
   ```

   indicates that ALPHA can appear alone or can be followed by ,BETA any number of times in succession.

9. Alphameric characters: unless otherwise indicated, an alphameric character is one of the following:

   - alphabetic: A-Z
   - numeric: 0-9
   - national: $ # @

Continuation Lines

Continue a command or subcommand, except the SLIP subcommand of OPERATOR, on one or more lines by following this rule:

Use either a hyphen (minus sign) or a plus sign as the last character on the line you wish to continue. If you use a plus sign, precede it by at least one blank to prevent the concatenation of character strings from line to line. (The plus sign causes TSO to delete leading delimiters (blanks, commas, tabs, and comments on the continuation line.)

Continue the SLIP subcommand of OPERATOR on one or more lines by following this rule:

Use a blank as the last character on the line you wish to continue.

You can end a line of input anywhere except:

- an equal sign and its preceding keyword must appear on the same line
- the binary indicator: *(b)* in the DATA keyword must appear on the same line.
- the complete keyword must appear on the same line

Delimiters

For all subcommands, except the SLIP subcommand of OPERATOR, the following rule on the use of delimiters applies: unless otherwise indicated, use a blank (or blanks) as the delimiter between a subcommand and a following parameter, and between parameters.

For the SLIP subcommand of OPERATOR, the following rule applies: blanks are not allowed except between SLIP and SET, MOD, or DEL.

Parameter Definitions

Two types of parameters are described under ACCOUNT and OPERATOR: positional and keyword. Positional parameters must appear in a command exactly as shown in the command syntax. Keyword parameters may appear in any order.

ACCOUNT Command

Use the ACCOUNT command and subcommands to create and to update the entries in the user attribute data set (UADS) and, simultaneously, the broadcast data set (SYS1.BRODCAST). This command can be executed as either a time-sharing or a batch job. Basically, the UADS is a list of terminal users who are authorized to use TSO. The UADS contains information about each of the users. The information in the UADS is used to regulate access to the system. SYS1.BRODCAST can contain notices and mail for all userids which are defined to it.

The syntax of the ACCOUNT command is:

ACCOUNT

The SYS1.UADS data set must be allocated as SHR prior to using the ACCOUNT command. You cannot accomplish any work with the ACCOUNT command until you use a subcommand to define the operation that you want to perform. The subcommands and the operations that they define are:

ADD	Add new entries to the UADS and SYS1.BRODCAST; add new data to existing entries.
CHANGE	Change data in specified fields of UADS entries; change userids in SYS1.BRODCAST.
DELETE	Delete entries or parts of entries from the UADS; delete userids from SYS1.BRODCAST.
END	Terminate the ACCOUNT command.
HELP	Obtain help from the system (not available for batch jobs).
LIST	Display the contents of an entry in the UADS.
LISTIDS	Display the user identifications for all entries.
SYNC	Build a new SYS1.BRODCAST data set and synchronize it with the UADS.

ADD Subcommand of ACCOUNT

Use the ADD subcommand to create new userids for prospective users of TSO. As you create a new userid, a corresponding entry is created in the UADS and SYS1.BRODCAST for that user. For each new userid that you create, the system builds a "typical" user profile in the user profile table (UPT) for that user.

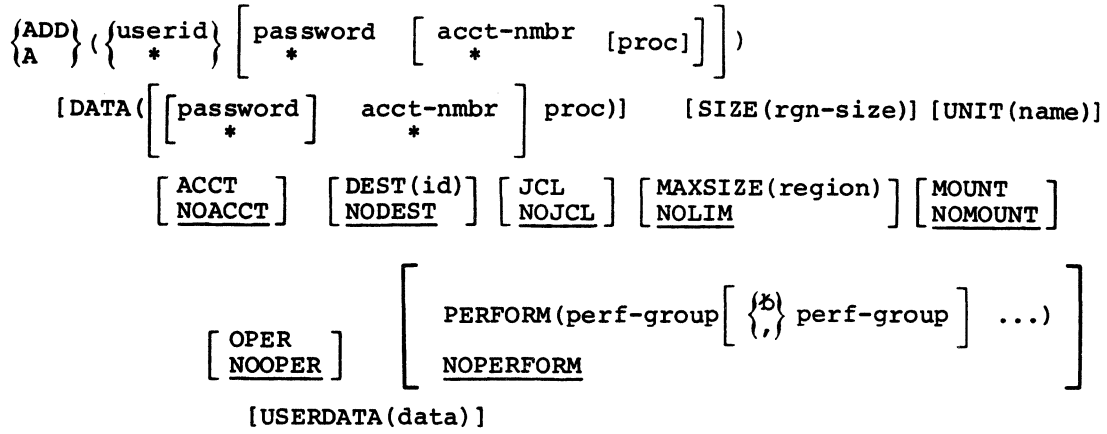
You can also use ADD to add additional data to an existing entry in the UADS. Do not use ADD to change any existing data in a UADS entry; use the CHANGE subcommand instead.

When adding a new entry to the UADS, you can also select the following options for the new user:

- The region size that he can request at logon
- The authority to use the ACCOUNT command
- The authority to use the OPERATOR command
- The authority to use the SUBMIT, STATUS, CANCEL, and OUTPUT commands
- The authority to specify performance groups at logon
- The authority to specify dynamic allocation requests for volume mounting
- The authority to specify remote work stations for SYSOUT data sets
- The installation-defined data to be added to the entry

When making additions to the UADS, ADD ensures that no duplications will exist in the UADS structure. If an item to be added is found to already exist at the specified location in an entry, no addition takes place.

The syntax of the ADD subcommand of ACCOUNT is:



- The first parameter (enclosed within parentheses) is a positional parameter; all others are keyword parameters.
- To create a new entry in the UADS and SYS1.BRODCAST, specify:
  A (*userid* *password* or * *acct-nmbr* or * *proc*)
- When you create a new entry, an asterisk (*) indicates a null field; that is, passwords and/or account numbers are not supported under that *userid*. Any subsequent explicit specification in either the positional or DATA parameter of *password* and/or *acct-nmbr* for that *userid* is invalid.
- If you create a new entry with explicit specification in the positional parameter for *password* and/or *acct-nmbr*, the specification of an asterisk (*) for the corresponding item in the DATA parameter is invalid.
- If you specify less than four items in the positional parameter, information is to be added to an existing entry (or entries); and you must specify those items omitted from the positional parameter in the DATA parameter.
- The specification of SIZE and/or UNIT is valid only if you specify *proc* in either the positional or DATA parameter.

userid
the user identification of a new entry in the UADS and SYS1.BRODCAST or an existing entry in the UADS to which information is to be added. (Note: If the specified *userid* already exists in SYS1.BRODCAST, all messages for that *userid* are deleted; the *userid* is not.)

value: 1-7 alphameric characters, beginning with an alphabetic or a national character

*
information is to be added to all existing entries in the UADS

password
a password that is part of a new entry or a password under the indicated userid(s) in an existing entry (or entries)

value: 1-8 alphameric characters

*
a null field in a new entry, or passwords are not supported under the indicated userid(s) in an existing entry (or entries), or all the passwords under the indicated userid(s) in an existing entry (or entries)

acct-nmbr
an account number that is part of a new entry or an account number under the indicated password(s) under the indicated userid(s) in an existing entry (or entries)

value: 1-40 alphameric characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character

*
a null field in a new entry, or account numbers are not supported under the indicated userid(s) in an existing entry (or entries), or all the account numbers under the indicated password(s) under the indicated userid(s) in an existing entry (or entries)

proc
a procedure name that is part of a new entry

value: 1-8 alphameric characters, beginning with an alphabetic character

DATA
information is to be added to an existing entry (or entries) in the UADS

password
a password or list of passwords to be added to an existing entry (or entries)

value: 1-8 alphameric characters

*
passwords are not supported under the indicated userid(s)

acct-nmbr
an account number or list of account numbers to be added to an existing entry (or entries). No more than 255 identical account numbers may exist under any one userid.

value: 1-40 alphameric characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character

*
account numbers are not supported under the indicated userid(s)

proc
a procedure name or list of procedure names to be added to an existing entry (or entries). No more than 255 identical procedure names may exist under any one userid.

value: 1-8 alphameric characters, beginning with an alphabetic character

When you specify a list of passwords and/or account numbers and procedure names, separate each item in the list by a comma or a blank and enclose each list within a separate set of parentheses embedded within the set required for the DATA parameter. If you specify only a list of procedure names, the embedded parentheses are optional.

SIZE
the minimum region size that will be assigned to a procedure if the user does not specify a region size at logon. If you do not specify SIZE or specify SIZE(0), the minimum region size available at logon is assigned to the user. If you specify in SIZE a minimum region size that is larger than MAXSIZE for the userid, SIZE is set equal to MAXSIZE. You can specify a SIZE parameter for each unique combination of password, account number, procedure name under a userid.

rgn-size
number of 1024-byte units of virtual storage for this user's private area

value: an integer in the range of 0-65535

UNIT
predefined specification of device use by a procedure. (Data sets allocated via the catalog are an exception. See the ALLOCATE command in the **TSO Command Language Reference.**) You can specify a UNIT parameter for each unique combination of password, account number, procedure name under a userid.

name
the name of a device or group of devices (for example, SYSDA)

value: 1-8 alphameric characters

- Use the following parameters only when you create a new entry in the UADS.

ACCT
the user is authorized to use the ACCOUNT command

NOACCT
the user is not authorized to use the ACCOUNT command

DEST
SYSOUT data sets, dynamically allocated by the user, are to be routed to a default destination. By allowing SYSOUT to be processed at the default destination, the user can eliminate ROUTE cards from his submitted batch jobs. The default destination can be overridden by the user through the use of the ALLOCATE, FREE, and other commands.

id
the default destination (remote work station)

value: 1-8 alphameric characters, beginning with an alphabetic or a national character

NODEST
the user must explicitly route his SYSOUT data sets for processing

JCL
the user is authorized to use the SUBMIT, STATUS, CANCEL, and OUTPUT commands

NOJCL
the user is not authorized to use the named commands

MAXSIZE
the maximum region size that the user may request at logon. If you do not specify MAXSIZE or specify MAXSIZE=0, NOLIM is assumed.

region
the number of 1024-byte units of virtual storage for the user's private area

value: an integer in the range of 0-65535

NOLIM
the user is not restricted to a maximum region size at logon

MOUNT
dynamic allocation requests for this userid are authorized to cause volume mounting as necessary. (Note: The volume request can be either explicit (for example, when the ALLOCATE command is issued) or implicit (for example, with commands that cause temporary space to be allocated).) No message is sent to the user indicating that operator action has been requested at the time the volume is mounted. The user will sit in a "locked out" condition at the terminal until the operator responds to the request. Therefore, the user should send a message to the operator prior to issuing the command requesting a particular volume.

NOMOUNT
dynamic allocation requests for this userid are not authorized to cause volume mounting

OPER
the user is authorized to use the OPERATOR command

NOOPER
the user is not authorized to use the OPERATOR command

PERFORM
the user is authorized to explicitly request a performance group (or groups) at logon

perf-group
the identification of the performance group (or groups)

value: an integer in the range of 1-255

Note: If the installation has written an installation control specification for TSO users, the following applies to the interpretation of the PERFORM parameter under the stated conditions:

- control performance group (PGN) specified
- optional control performance group (OPGN) not specified
  - If a user specifies a performance group at logon and that performance group is not valid under that userid, logon prompts the user for a valid performance group even though it will be ignored and the value of PGN assigned.
- both control performance group (PGN) and optional control performance group (OPGN) specified.
  - If a user specifies a performance group at logon and that performance group is not valid under that userid, logon prompts the user for a valid performance group. If the user then specifies a valid performance group that is equal to a specified OPGN value, the logon value is accepted; otherwise, the value of PGN is assigned.
  - If a user specifies a valid performance group (as specified in the UADS) at logon and that value does not equal a value specified for OPGN, the PERFORM parameter is ignored and the value of PGN is assigned.

NOPERFORM
an installation-defined performance group will be assigned to the user at logon. (The user is not authorized to explicitly request a performance group.)

USERDATA
installation-defined data is to be added under this userid. The two-byte field in the UADS is a four-digit hexadecimal number that represents the contents of data. The meaning of the field is defined by the user.

data
the data to be added

value: 4 EBCDIC characters (valid characters 0-9 and A-F)

Example 1

Operation: Add a new entry to the UADS and SYS1.BRODCAST.

```
add (warner1 xaybzc 32058 mylog) noacct nooper jcl -
maxsize(150) size(80) unit(sysda) userdata(1fa*) perform (1,5,6,2,4)
dest(deptout) mount
```

Example 2

Operation: Add a new password, account number, and procedure name to an existing entry in the UADS. Also include the region size requirements for the procedure.

```
add (warner1) data(mz3tii 7116166 amabala) size(20)
```

Example 3

Operation: Continuing Example 2, add a new account number and procedure name to an existing entry in the UADS.

```
add (warner1 mz3tii) data(288104 mylog) size(114) unit(sysda)
```

Example 4

Operation: Add a new procedure name, and the region size requirements for it, to all entries in the UADS.

```
add (* * *) data(mcqlg) size (73)
```

Example 5

Operation: Add a new account number and procedure name to all structures under an existing entry in the UADS.

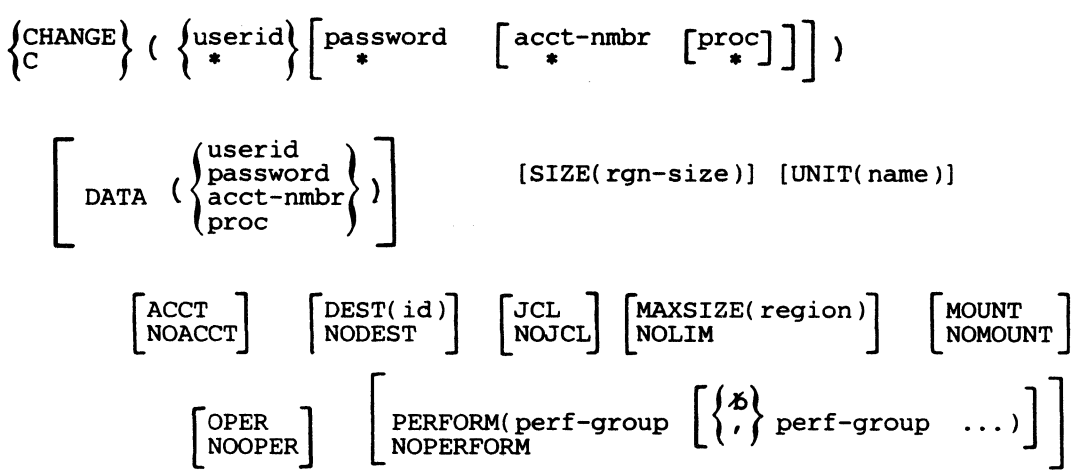
```
add (warner1 *) data(5707571 logproc) size(100)
```

CHANGE Subcommand of ACCOUNT

Use the CHANGE subcommand to change existing fields of data within entries in the UADS and userids in SYS1.BRODCAST.

When making changes to the UADS, CHANGE ensures that no identical (redundant) paths will exist in the UADS structure after the change operation. On the other hand, if an 'impossible merge' situation arises (identical procedure names associated with different data), CHANGE cannot determine which data to retain. Therefore, it terminates processing of the current structure and issues an explanatory message.

The syntax of the CHANGE subcommand of ACCOUNT is:



- The first parameter (enclosed within parentheses) is a positional parameter; all others are keyword parameters.
- To change a userid in the UADS and SYS1.BRODCAST, explicitly specify *userid* as the only item in the positional parameter and specify *userid* in the DATA parameter. (The specifications c(*) or c(*) data (*userid*) are invalid.)
- To change password(s), account number(s), or procedure name(s) in an entry (or entries) in the UADS, specify the item (either explicitly or as an asterisk (*)) as the final item in the positional parameter, and specify the corresponding item in the DATA parameter.
- If you created (with the ADD subcommand) an entry in the UADS with an asterisk (*) specification for password and/or account number, an explicit specification in either the positional or DATA parameter of *password* and/or *acct-nmbr* for the particular userid is invalid.
- If you change MAXSIZE for a userid, the specification of a region size smaller than an existing SIZE for any procedure under that userid is invalid.
- The specification of SIZE and/or UNIT is valid only if you specify *proc* or * in the positional parameter, or *proc* in the DATA parameter.
- If you specify in the SIZE parameter for a procedure a minimum region size larger than MAXSIZE for the userid, SIZE is set equal to MAXSIZE.

userid

the user identification in the UADS and SYS1.BRODCAST to be changed or the UADS entry to be changed

value: 1-7 alphameric characters, beginning with an alphabetic or a national character

Note: In SYS1.BRODCAST, the effect of

c $userid_1$ data *(*$userid_2$ *)*

is as follows:

- *If* $userid_1$ *and* $userid_2$ *do not exist, add* $userid_2$ *as a new entry.*
- *If* $userid_1$ *exists and* $userid_2$ *does not, delete* $userid_1$ *; add* $userid_2$ *as a new entry; chain messages for* $userid_1$ *to* $userid_2$ *.*
- *If* $userid_1$ *and* $userid_2$ *both exist, delete* $userid_1$ *; chain messages for* $userid_1$ *to* $userid_2$

*

all entries in the UADS are to be changed

password

the password to be changed or the password under the indicated userid(s)

value: 1-8 alphameric characters

*

all passwords are to be changed, or passwords are not supported under the indicated userid(s), or all the passwords under the indicated userid(s)

acct-nmbr

the account number to be changed or the account number under the indicated password(s) under the indicated userid(s)

value: 1-40 alphameric characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character

*

all account numbers are to be changed, or account numbers are not supported under the indicated userid(s), or all the account numbers under the indicated password(s) under the indicated userid(s)

proc

the procedure name to be changed or the requirements of the procedure are to be changed (indicated by the specification of SIZE and/or UNIT without the specification of DATA)

value: 1-8 alphameric characters, beginning with an alphabetic character

*

all procedure names are to be changed or the requirements of all procedures in the indicated entry (or entries) are to be changed (indicated by the specification of SIZE and/or UNIT without the specification of DATA)

DATA

data is to be changed in an existing entry (or entries)

userid

a user identification to replace an existing userid

value: 1-7 alphameric characters, beginning with an alphabetic or a national character

password

a password to replace an existing password(s)

value: 1-8 alphameric characters

acct-nmber

an account number to replace an existing account number(s)

value: 1-40 alphameric characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character

proc

a procedure name to replace an existing procedure name(s)

value: 1-8 alphameric characters, beginning with an alphabetic character

SIZE
the minimum region size that will be assigned to a procedure if the user does not specify a region size at logon. If you do not specify SIZE or specify SIZE(0), the minimum region size available at logon is assigned to the user. If you specify in SIZE a minimum region size that a larger than MAXSIZE for the userid, SIZE is set equal to MAXSIZE. You can specify a SIZE parameter for each unique combination of password, account number, procedure name under a userid.

rgn-size
number of 1024-byte units of virtual storage for the user's private area

value: an integer in the range of 0-65535

UNIT
predefined specification of device use by a procedure. (Data sets allocated via the catalog are an exception. See the ALLOCATE command in the **TSO Command Language Reference.**) You can specify a UNIT parameter for each unique combination of password, account number, procedure name under a userid.

name
the name of a device or group of devices (for example, SYSDA)

value: 1-8 alphameric characters

- Use the following parameters only when you change the attributes of a userid(s). All attribute changes must be explicitly specified.

ACCT
the user is authorized to use the ACCOUNT command

NOACCT
the user is not authorized to use the ACCOUNT command

DEST
SYSOUT data sets, dynamically allocated by the user, are to be routed to a default destination. By allowing SYSOUT to be processed at the default destination, the user can eliminate ROUTE cards from his submitted batch jobs. The default destination can be overridden by the user through the use of the ALLOCATE, FREE, and other commands.

id
the default destination (remote work station)

value: 1-8 alphameric characters, beginning with an alphabetic or a national character

NODEST
the user must explicitly route his SYSOUT data sets for processing

JCL
the user is authorized to use the SUBMIT, STATUS, CANCEL, and OUTPUT commands

NOJCL
the user is not authorized to use the named commands

MAXSIZE
the maximum region size that the user may request at logon. If you do not specify MAXSIZE or specify MAXSIZE=0, NOLIM is assumed

region
the number of 1024-byte units of virtual storage for the user's private area

value: an integer in the range of 0-65535

NOLIM
the user is not restricted to a maximum region size at logon

MOUNT
dynamic allocation requests for this userid are authorized to cause volume mounting as necessary. (Note: The volume request can be either explicit (for example, when the ALLOCATE command is issued) or implicit (for example, with commands that cause temporary space to be allocated).) No message is sent to the user indicating that operator action has been requested at the time the volume is mounted. The user will sit in a 'locked out' condition at the terminal until the operator responds to request. Therefore, the user should send a message to the operator prior to issuing the command requesting a particular volume.

NOMOUNT
dynamic allocation requests for this userid are not authorized to cause volume mounting

OPER
the user is authorized to use the OPERATOR command

NOOPER
the user is not authorized to use the OPERATOR command

PERFORM
the user is authorized to explicitly request a performance group (or groups) at logon

perf-group
the identification of the performance group (or groups)

value: an integer in the range of 1-255

Note: If the installation has written an installation control specification for TSO users, the following applies to the interpretation of the PERFORM parameter under the stated conditions:

- control performance group (PGN) specified
- optional control performance group (OPGN) not specified
  - If a user specifies a performance group at logon and that performance group is not valid under that userid, logon prompts the user for a valid performance group even though it will be ignored and the value of PGN assigned.
- both control performance group (PGN) and optional control performance group (OPGN) specified.
  - If a user specifies a performance group at logon and that performance group is not valid under that userid, logon prompts the user for a valid performance group. If the user then specifies a valid performance group that is equal to a specified OPGN value, the logon value is accepted; otherwise, the value of PGN is assigned.
  - If a user specifies a valid performance group (as specified in the UADS) at logon and that value does not equal a value specified for OPGN, the PERFORM parameter is ignored and the value of PGN is assigned.

NOPERFORM
an installation-defined performance group will be assigned to the user at logon. (The user is not authorized to explicitly request a performance group.)

USERDATA
installation-defined data is to be added under this userid. The two-byte field in the UADS is a four-digit hexadecimal number that represents the contents of data. The meaning of the field is defined by the user.

data
the data to be added

value: 4 EBCDIC characters (valid characters 0-9 and A-F)

Example 1

Operation: Change an account number for an entry in the UADS and authorize the user to issue the ACCOUNT and OPERATOR commands.

```
change (slc05 aox3p se29705) data(2e26705) acct oper
```

Example 2

Operation: Authorize all users to issue the SUBMIT, CANCEL, STATUS, and OUTPUT commands.

```
change (*) jcl
```

The asterisk in the first positional parameter position specifies that all user identities are considered valid for the operation of this subcommand.

Example 3

Operation: Change the user identification for an entry in the UADS.

```
change (warner) data(renwar)
```

Example 4

Operation: Change the name of a procedure for an entry that consists of a user identification, a procedure name, and attributes (passwords and account numbers are not supported under the indicated userid).

```
change (jal95 * * oldproc) data(newproc)
```

Example 5

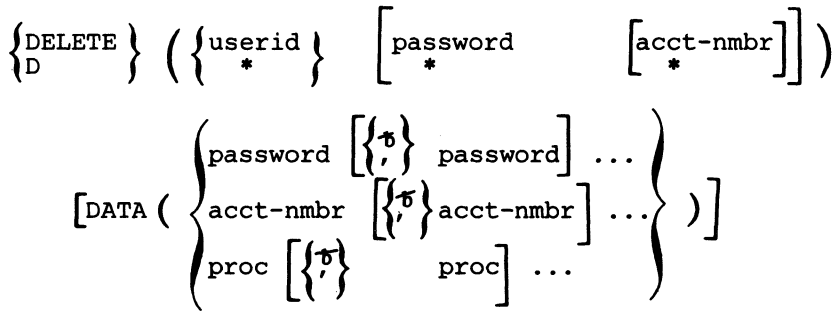
Operation: Change the default destination for an entry in the UADS.

```
change (ceh01) dest(rmt1)
```

DELETE Subcommand of ACCOUNT

Use the DELETE subcommand to delete data from the UADS and userids from SYS1.BRODCAST. Each terminal user has an entry in the UADS; and each entry contains several items of data. The data that you want to delete may be a part of an existing entry, or may be an entire existing entry.

The syntax of the DELETE subcommand of ACCOUNT is:



- The first parameter (enclosed within parentheses) is a positional parameter.
- To delete an entire entry in the UADS and a userid (with associated messages) in SYS1.BRODCAST, specify:
  d(*userid*)
- To delete all entries in the UADS and all userids (and all messages) in SYS1.BRODCAST, specify:
  d(*)
- When you delete an item in an entry in the UADS, all lower-level items under that item are also deleted.
- When you delete the only password, account number, or procedure name under a userid, the userid is also deleted.
- The explicit specification of the same item in both the positional and DATA parameters is invalid.
- If you created (with the ADD subcommand) an entry in the UADS with an asterisk (*) specification for password and/or account number, an explicit specification in either the positional or DATA parameter of *password* and/or *acct-nmbr* for the particular userid is invalid.

userid
the entry in the UADS and the userid in SYS1.BRODCAST to be deleted or the entry in the UADS from which data is to be deleted

value: 1-7 alphameric characters, beginning with an alphabetic or a national character

*
all entries in the UADS and all userids and all messages in SYS1.BRODCAST are to be deleted or some data in all the entries in the UADS is to be deleted

password
the password to be deleted or the password under the indicated userid(s)

value: 1-8 alphameric characters

*
passwords are not supported under the indicated userid

acct-nmber
the account number to be deleted or the account number under the indicated password under the indicated userid(s)

value: 1-40 alphameric characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character

*
account numbers are not supported under the indicated userid

DATA
data is to be deleted from an existing entry in the UADS

password
the password or list of passwords to be deleted

value: 1-8 alphameric characters

acct-nmbr
the account number or list of account numbers to be deleted

value: 1-40 alphameric characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character

proc
the procedure name or list of procedure names to be deleted

value: 1-8 alphameric characters, beginning with an alphabetic character

Example 1

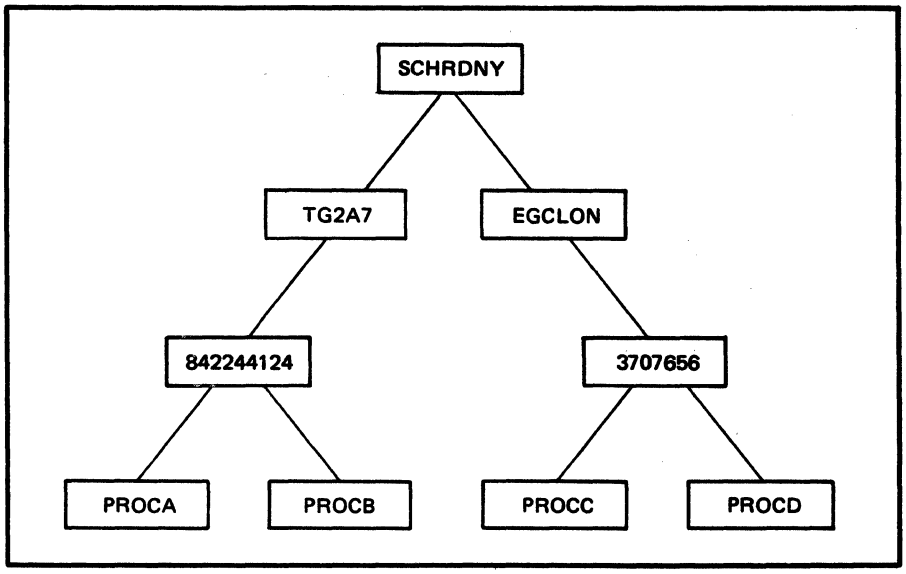
Operation: Delete an entire entry from the UADS.

```
delete (early08)
```

Example 2

Operation: Delete a procedure name from an entry in the UADS having the following structure.

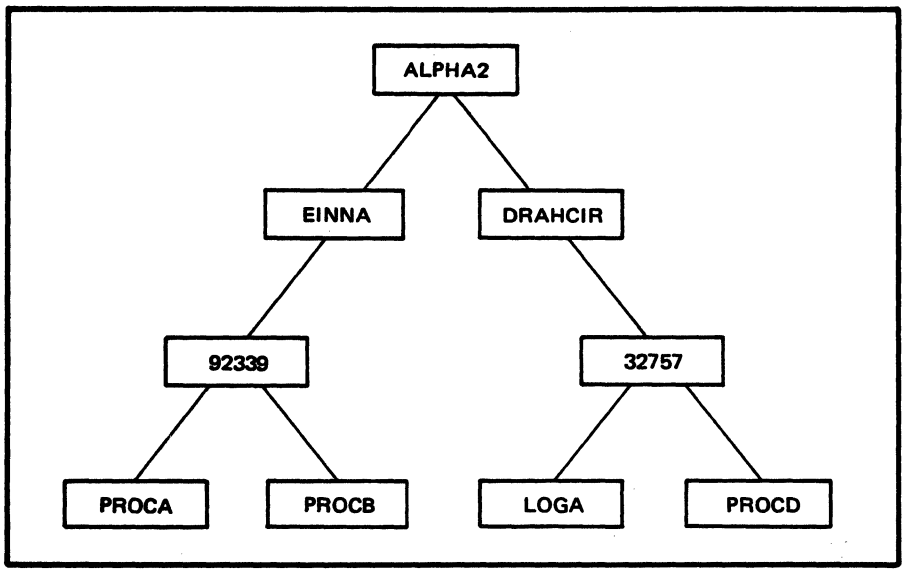
```
delete (schrdny egclon 3707656) data(procd)
```



Example 3

Operation: Delete an account number and all procedure names under that account number from an entry in the UADS having the following structure.

```
delete (alpha2 drahcir 32757)
```



END Subcommand of ACCOUNT

Use the END subcommand to terminate operation of the ACCOUNT command. After entering the END subcommand, you may enter new commands.

The syntax of the END subcommand of ACCOUNT is:

```
END
```

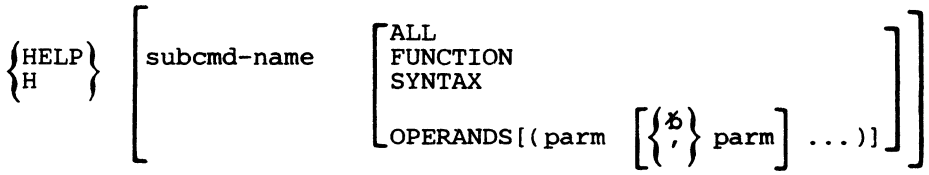
HELP Subcommand of ACCOUNT

Use the HELP subcommand to find out how to use the ACCOUNT subcommands. When you enter the HELP subcommand, the system responds by printing out explanatory information at your terminal. You may request:

- A list of available subcommands
- An explanation of the function, syntax, and parameters (both positional and keyword) of a specific subcommand

The HELP subcommand actually causes the system to execute a function of the HELP command; therefore, you may consult the discussion of the HELP command in **TSO Command Language Reference**.

The syntax of the HELP subcommand of ACCOUNT is:



- If you specify HELP with no parameters, a list of available subcommands of ACCOUNT is displayed at your terminal.

subcmd-name
the subcommand you want clarified

value: any valid subcommand of ACCOUNT

ALL
a description of the function, syntax, positional parameters, and keyword parameters of the subcommand is displayed

FUNCTION
a description of the function of the subcommand is displayed

SYNTAX
a description of the proper syntax of the subcommand is displayed

OPERANDS
a description of the positional and keyword parameters of the subcommand is displayed

parm
only a description of the indicated keyword parameter(s) of the subcommand is displayed

value: any valid keyword parameter of the subcommand

Example 1

Operation: Have a list of available subcommands displayed at your terminal.

```
help
```

Example 2

Operation: Obtain all available information about the ADD subcommand.

```
h add
```

Example 3

Operation: Have a list of the positional and keyword parameters for the CHANGE subcommand displayed at your terminal.

```
h change operands
```

Example 4

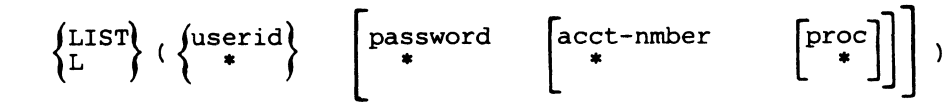
Operation: Have a list of the indicated keyword parameters for the ADD subcommand displayed at your terminal.

```
h add operands (data mount userdata)
```

LIST Subcommand of ACCOUNT

Use the LIST subcommand to display entries in the UADS or to display fields of data from within particular entries.

The syntax of the LIST subcommand of ACCOUNT is:



- The parameter enclosed within parentheses is positional.
- If you created (with the ADD subcommand) an entry in the UADS with an asterisk (*) specification for password and/or account number, an explicit specification in the positional parameter of *password* and/or *acct-nmbr* is invalid.

userid
the userid or the UADS entry to be listed

value: 1-7 alphameric characters, beginning with an alphabetic or a national character

*
all userids or all UADS entries are to be listed

password
the password to be listed or the password under the indicated userid(s)

value: 1-8 alphameric characters

*
all passwords are to be listed, or passwords are not supported under the indicated userid(s), or all the passwords under the indicated userid(s)

acct-nmbr
the account number to be listed or the account number under the indicated password(s) under the indicated userid(s)

value: 1-40 alphameric characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character

*
all account numbers are to be listed, or account numbers are not supported under the indicated userid(s), or all the account numbers under the indicated password(s) under the indicated userid(s)

proc
the procedure name to be listed

value: 1-8 alphameric characters, beginning with an alphabetic character

*
all procedure names are to be listed

Implicit specifications:

- The specification of L (*userid*) implies L (*userid* * * *)
- The specification of L (*userid password*) implies L (*userid password* * *)
- The specification of L (*userid password acct-nmbr*) implies
  L (userid password acct-nmbr *)

The LIST processor generates the necessary asterisks and that information is also listed.

Example 1

Operation: List the contents of the UADS.

```
list (*)
```

Example 2

Operation: List all of a particular entry in the UADS.

```
list (wrrid)
```

Example 3

Operation: List all of the account numbers under a specific password for a particular entry.

```
list (wrrid roolf *)
```

Example 4

Operation: List all references to a specific procedure for all entries.

```
l (* * * proc01)
```

LISTIDS Subcommand of ACCOUNT

Use the LISTIDS subcommand to have a list of the user identifications in the UADS displayed at your terminal.

The syntax of the LISTIDS subcommand of ACCOUNT is:

{LISTIDS / LISTI}

Example 1

Operation: List all user identifications in the UADS.

```
listids
```

SYNC Subcommand of ACCOUNT

When the UADS is created, use the SYNC subcommand to build a new SYS1.BRODCAST data set and synchronize it with the UADS. (All userids from the UADS are entered into SYS1.BRODCAST.)

If you use SYNC when the UADS exists, all messages (MAIL) are deleted from SYS1.BRODCAST.

SYNC also formats the NOTICES section of SYS1.BRODCAST to reserve room for the maximum number of messages. (The maximum number of messages is specified at system generation in the SCHEDULR macro.)

The syntax of the SYNC subcommand of ACCOUNT is:

```
SYNC
```

OPERATOR Command

Use the OPERATOR command (along with its subcommands) to regulate and maintain TSO from a terminal. The authority to use OPERATOR is normally given to personnel responsible for system operation. When an entry is created (or changed) in the UADS, OPER is specified for the userid.

The OPERATOR command is fully supported only for terminals that have the transmit-interruption capability; that is, this command is supported only for those terminals for which the BREAK parameter of the TERMINAL command is valid.

The syntax of the OPERATOR command is:

```
{OPERATOR}
{OPER    }
```

The OPERATOR command, through the use of its subcommands, allows the terminal user to control TSO as follows:

CANCEL	Cancel a terminal session.
DISPLAY	Display summary or detailed information about users and jobs, the time of day and the date, and summary or detailed information about SLIP traps.
END	Terminate the OPERATOR command (thereby removing the user's terminal from OPERATOR mode).
HELP	Get a list of the subcommands of the OPERATOR command, along with the function, syntax, and parameters of the subcommands.
MONITOR	Monitor both terminal and background job activities within the system. Informational messages will be displayed.
SEND	Send a message to other terminal users or operators.
SLIP	Control SLIP/PER (serviceability level indication processing/program event recording).
STOPMN	Terminate the monitoring operations of the MONITOR subcommand; the display of information messages at the user's terminal will be stopped.

CANCEL Subcommand of OPERATOR

Use the CANCEL subcommand to terminate the current activities of a TSO user. When you use the CANCEL command to terminate a session, accounting information will be presented to the user.

The syntax of the CANCEL subcommand of OPERATOR is:

```
{CANCEL}      {userid          }
{C     } U= {*LOGON*,A=asid  } [,DUMP]
```

- If a user is currently logged on, the specification of *userid* is sufficient to terminate that user.
- If a user is attempting to logon and the logon has not completed or cannot complete, the system rejects the command:

  c u=*userid*

  In this case, issue

  d ts a

  In the display, note the 'userids' shown as *LOGON* and their corresponding ASIDs. Then issue

  c u=*logon*, a=*asid*

  to terminate a particular user.
- If a logon completes before the CANCEL command takes effect, the system rejects the command:

  c u=*logon*, a=*asid*

  In this case, reissue

  c u=*userid*

  to terminate the user.

userid
the user identification of a logged on TSO user whose session you are terminating

value: 1-7 alphameric characters, beginning with an alphabetic or a national character

LOGON
the 'user identification' of a TSO user attempting to logon

asid
an address space identifier

value: 1-4 hexadecimal digits

DUMP
an abnormal-end-of-job storage dump is taken. (The dump is printed on the system output device.)

Example 1

Operation: Terminate a TSO user's session with a dump.

```
c   u=slcid,dump
```

Example 2

Operation: Terminate a TSO user attempting to logon.

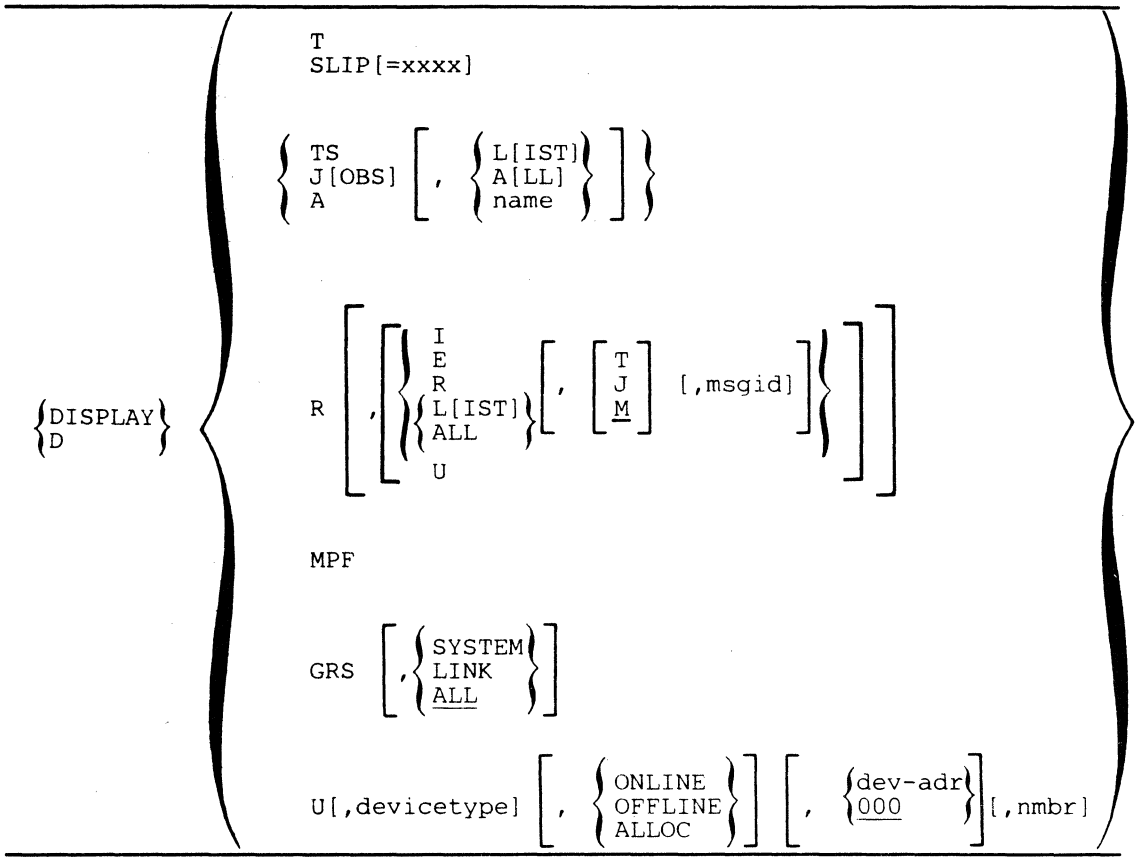
```
c   u=*logon*,a=002F
```

DISPLAY Subcommand of OPERATOR

Use the DISPLAY subcommand to display:

- Summary or detailed information about users, jobs, address spaces, and devices
- The time of day and the date
- The status of the message processing facility (MPF)
- Information about the status of the global resource serialization complex
- Summary or detailed information about SLIP traps

The syntax of the DISPLAY subcommand of OPERATOR is:



- The specification of a d a, d u, d mpf, or d grs subcommand is valid only if you have TSO/E installed.
- The specification of a d ts, d j, or d a subcommand displays exactly the same information.
- All parameters on a d r or d u subcommand are positional. Therefore, if a parameter is not specified, its absence must be indicated by a comma. The following specifications illustrate the rule:

d r,i,,*msgid*	d u,,alloc
d r,,t	d u,tp,,,*nmbr*
d r,,,*msgid*	d u,,,,*nmbr*

T

display the local time of day and the date; and the Greenwich Mean Time (GMT) of day and the date.

SLIP

display summary information about all the SLIP traps in the system. (The information consists of the trap ids and whether a trap is enabled or disabled.)

xxxx
display detailed information about the SLIP trap identified by xxxx. (See the SLIP subcommand for details about xxxx.)

value: 1-4 alphameric characters

TS
display the number of active batch jobs, started tasks (MOUNT commands in execution are treated as started tasks), TSO users currently logged on, active system address spaces (e.g., master, global resource serialization, auxiliary, etc.), active initiators; and if TSO/VTAM is running, the number of users logged on, and the maximum number allowed to be logged on, under TSO/VTAM.

LIST or L
include in the display a list of TSO userids currently logged and the status of each address space.

ALL or A
include in the display a list of TSO userids currently logged on, and, for each address space, also include:

- status
- ASID
- program event recording (PER) active indicator
- number of step-must-complete requests
- performance group number
- domain number
- CPU affinity
- accumulated CPU time
- elapsed time since logon

name
include in the display only those specified TSO userid(s) currently logged on, and, for each address space, also include:

- status
- ASID
- program event recording (PER) active indicator
- number of step-must-complete requests
- performance group numbers
- domain number
- CPU affinity
- accumulated CPU time
- elapsed time since logon

value: 1. 1-7 alphameric characters (If the specified TSO userid is LIST, L, ALL, or A, enclose the userid in parentheses.)

2. 1-6 alphameric characters followed by an asterisk (All TSO userids beginning with the specified alphameric character(s) are included in the display.)

JOBS or J
display the number of active batch jobs, started tasks (MOUNT commands in execution are treated as started tasks), TSO users currently logged on, active system address spaces (e.g., master, global resource serialization, auxiliary, etc.), active initiators; and if TSO/VTAM is running, the number of users logged on, and the maximum number allowed to be logged on, under TSO/VTAM.

LIST or L
include in the display, for each active batch job and started task, the jobname, stepname, procedure stepname, V=R region boundaries, and the status of each address space.

ALL or A
include in the display, for each active batch job and started task, the jobname, stepname, procedure stepname, V=R region boundaries, and, for each address space, also include:

- status
- ASID
- program event recording (PER) active indicator
- number of step-must-complete requests
- performance group number
- domain number
- CPU affinity
- accumulated CPU time
- elapsed time since initiation

name
for each specified batch job or started task, or a specified system address space, include in the display

- for each specified active batch job and started task
  - jobname
  - stepname
  - procedure stepname
  - V=R region boundaries
- for the specified active system address space
  - jobname
  - stepname
  - procedure stepname
- for each of the specified active address spaces
  - status
  - ASID
  - program event recording (PER) active indicator
  - number of step-must-complete requests
  - performance group number
  - domain number
  - CPU affinity
  - accumulated CPU time
  - elapsed time since initiation

value: 1. 1-8 alphameric characters (If the specified jobname/started task is LIST, L, ALL, or A, enclose the name in parentheses.)

2. 1-7 alphameric characters followed by an asterisk (All jobnames/started tasks beginning with the specified alphameric character(s) are included in the display.)

3. specific name of a system address space, e.g.,

   MASTER

   GRS

   ALLOCAS

A

display the number of active batch jobs, started tasks (MOUNT commands in execution are treated as started tasks), TSO users currently logged on, active system address spaces (e.g., master, global resource serialization, auxiliary, etc.), active initiators; and if TSO/VTAM is running, the number of users logged on, and the maximum number allowed to be logged on, under TSO/VTAM.

LIST or L

include in the display a list of TSO userids currently logged on and, for each active batch job and started task, the jobname, stepname, procedure stepname, V=R region boundaries, and the status of each address space.

ALL or A

include in the display

- a list of TSO userids currently logged on
- for each active batch job and started task
  - jobname
  - stepname
  - procedure stepname
  - V=R region boundaries
- for each active system address space
  - name (e.g., *MASTER*, GRS, etc.)
  - stepname
  - procedure stepname
- for every active address space
  - status
  - ASID
  - program event recording (PER) active indicator
  - number of step-must-complete requests
  - performance group number
  - domain number
  - CPU affinity
  - accumulated CPU time
  - elapsed time since logon or initiation

name

for each specified TSO userid, batch job, or started task, or a specified system address space, include in the display

- a list of specified TSO userids currently logged on
- for each specified active batch job and started task
  - jobname
  - stepname
  - procedure stepname
  - V=R region boundaries
- for the specified active system address space
  - name
  - stepname
  - procedure stepname
- for each of the specified active address spaces
  - status
  - ASID
  - program event recording (PER) active indicator
  - number of step-must-complete requests
  - performance group number
  - domain number
  - CPU affinity
  - accumulated CPU time
  - elapsed time since logon or initiation

value: 1. TSO userids

a. 1-7 alphameric characters (If the specified TSO userid is LIST, L, ALL, or A, enclose the userid in parentheses.)

b. 1-6 alphameric characters followed by an asterisk (All TSO userids beginning with the specified alphameric character(s) are included in the display.)

2. Jobnames/started tasks

a. 1-8 alphameric characters (If the specified jobname or started task is LIST, L, ALL, or A, enclose the name in parentheses.)

b. 1-7 alphameric characters followed by an asterisk (All jobnames and started tasks beginning with the specified alphameric character(s) are included in the display.)

3. Specific name of a system address space, e.g.,

MASTER

GRS

ALLOCAS

R

display the number of:

- messages awaiting replies
- outstanding immediate action messages (descriptor codes 1 and 2)
- outstanding eventual action messages (descriptor codes 3 and 11)
- outstanding mount requests
- outstanding operator interventions required
- and the status (active or not active) of the action message retention facility.

I
include in the display the message id and the message text of all outstanding immediate action messages.

E
include in the display the message id and the message text of all outstanding eventual action messages.

R
include in the display the message id and the message text of all messages awaiting replies.

LIST or L
include in the display:

- the message id and message text of all
  - outstanding immediate action messages
  - outstanding eventual action messages
  - messages awaiting replies
- the unit numbers of all devices
  - with outstanding mount requests
  - awaiting operator intervention

ALL
the same inclusion specified under LIST.

omitted operand
the same inclusion specified under LIST.

T
include in the display the time a message was issued and the job id of the issuer.

J
include in the display the job id of the issuer of a message.

M
do not include in the display either the time a message was issued or the job id of the issuer.

msgid
limit the amount of information in the display (for example, suppress message ids and accompanying text) to that identified by the value of *msgid*.

value: 1-8 alphameric characters

U
display the unit numbers of all devices with outstanding mount requests and of all devices awaiting operator intervention.

MPF
display information concerning message suppression, and color and highlighting options

- if message suppression is active, display the two-character identifier (xx) and the contents of the MPFLSTxx member of SYS1.PARMLIB that indicates those messages currently being suppressed and not being displayed on the operator's console.
- if message suppression is inactive, the display indicates that fact.
- if installation-defined color and highlighting options are in effect, display the two-character identifier (xx) and the contents of the MPFLSTxx member of SYS1.PARMLIB that defines the options
- if default color and highlighting options are in effect, display the identifier DF and the default options.

GRS
display both system and CTC information about the current global resource serialization complex

SYSTEM
display only system information (For each system in the complex, the display includes system name, state, and communication status.)

LINK
display only CTC information (For each CTC assigned to global resource serialization on this system, the display includes the device address, status, and target system name.)

ALL
display both system and CTC information about the current global resource serialization complex

U
display status information about all device types, including non-supported devices (those specified at system generation in the IODEVICE macro with the DUMMY= parameter)

devicetype
display status information about particular device types

value: as indicated in the following list

CTC - channel-to-channel adapters

TP - communications equipment

GRAPHIC - graphic devices

TAPE - magnetic tape units

DASD - direct access storage devices

UR - unit record devices

ALL - equivalent to specifying d u

ONLINE
include in the display only online devices

OFFLINE
include in the display only offline devices

ALLOC
include in the display the jobname and ASID of each job to which a device is presently allocated

omitted operand
include in the display only online and offline devices

dev-adr
include in the display only devices whose numbers are equal to or greater than *dev-adr.*

value: 3 hexadecimal digits

nmbr
include in the display only a specific number of devices

value: 1-4 decimal digits

- If neither *nmbr* nor ALLOC is specified, *nmbr* defaults to 100.
- If *nmbr* is not specified, but ALLOC is, *nmbr* defaults to 8.

Descriptor Code Meanings

- Action messages with descriptor code 1 - an uncorrectable error has occurred and the operator must restart the system or a major subsystem.
- Action messages with descriptor code 2 - the operator must perform an action immediately; the issuing task waits until the requested action is performed.
- Action messages with descriptor code 3 - the operator must perform an action eventually; the issuing task does not wait for the completion of the action.
- Action messages with descriptor code 11 - the operator must perform a critical action eventually; the issuing task does not wait for the completion of the action.

Example 1

Operation: Display the number of, and a list of, the TSO users currently logged on.

```
d   ts,list
```

Example 2

Operation: Display the time of day and the date.

```
d   t
```

Example 3

Operation: Display detailed information about SLIP trap 502X.

```
d   slip=502x
```

Example 4

Operation: Display -

- the number of messages awaiting replies
- the number of outstanding immediate and eventual action messages
- the number of outstanding mount requests
- the number of outstanding operator interventions required
- the status of the action message retention facility
- the specified message ids and message texts

```
d r,,,iee
```

Example 5

Operation: Display -

- the number of messages awaiting replies
- the number of outstanding immediate and eventual action messages
- the number of outstanding mount requests
- the number of outstanding operator interventions required
- the status of the action message retention facility
- the specified message ids and message texts for the outstanding immediate action messages

```
d r,i,,iee76
```

END Subcommand of OPERATOR

Use the END subcommand to terminate operation of the OPERATOR command. After entering the END subcommand, you may enter new commands.

The syntax of the END subcommand of OPERATOR is:

END

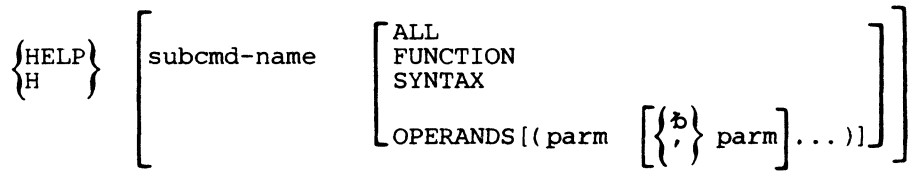
HELP Subcommand of OPERATOR

Use the HELP subcommand to find out how to use the OPERATOR subcommands. When you enter the HELP subcommand, the system responds by printing out explanatory information at your terminal. You may request:

- A list of available subcommands
- An explanation of the function, syntax, and parameters of a specific subcommand

The HELP subcommand actually causes the system to execute a function of the HELP command; therefore, you may consult the discussion of the HELP command in **TSO Command Language Reference**, if you desire more detailed information.

The syntax of the HELP subcommand of OPERATOR is:



- If you specify HELP with no parameters, a list of available subcommands of OPERATOR is displayed at your terminal.

subcmd-name
the subcommand you want clarified

value: any valid subcommand of OPERATOR

ALL
a description of the function, syntax, positional parameter, and keyword parameters of the subcommand is displayed

FUNCTION
a description of the function of the subcommand is displayed

SYNTAX
a description of the proper syntax of the subcommand is displayed

OPERANDS
a description of the positional and keyword parameters of the subcommand is displayed

parm
a description of only the indicated keyword parameter(s) of the subcommand is displayed

value: any valid keyword parameter of the subcommand

Example 1

Operation: Have a list of available subcommands displayed at your terminal.

```
help
```

Example 2

Operation: Obtain available information about a particular subcommand.

```
h monitor
```

Example 3

Operation: Have a list of the parameters for a particular subcommand displayed at your terminal.

```
h display operands
```

MONITOR Subcommand of OPERATOR

Use the MONITOR subcommand to monitor terminal activities and job activities within the system. Informational messages will be displayed. The content of the messages will pertain to the type of information indicated by the parameter included with the MONITOR subcommand. The system will continue to issue these informational messages until halted by a STOPMN subcommand or until you terminate the OPERATOR command.

The syntax of the MONITOR subcommand of OPERATOR is:

JOBNAMES
the name of each job is displayed both when the job starts and terminates, and unit record allocation is displayed when the job step starts. If a job terminates abnormally, the jobname appears in the diagnostic message; the message "jobname ENDED" does not appear.

SESS
the userid is displayed whenever a terminal session is initiated or terminated. If a terminal session terminates abnormally, the userid appears in the diagnostic message. If a terminal session is canceled, the message "user LOGGED OFF" does not appear.

T
the local time of day is displayed in the following format:

hh.mm.ss

The variable are:

hh-hours (00-23)
mm-minutes (00-59)
ss-seconds (00-59)

Note: After the initial specification (by any user) of the T parameter in the MONITOR subcommand, all subsequent users of MONITOR receive the time of day at their terminals, whether or not they specify T.

STATUS
the names and volume serial numbers of data sets with dispositions of KEEP, CATLG, or UNCATLG are displayed whenever the data sets are freed

Example 1

Operation: Have the system notify you whenever a terminal session begins or ends.

```
monitor sess
```

Example 2

Operation: Have displayed at your terminal the name of each job when the job starts and when it terminates. Also have the time displayed with the jobname.

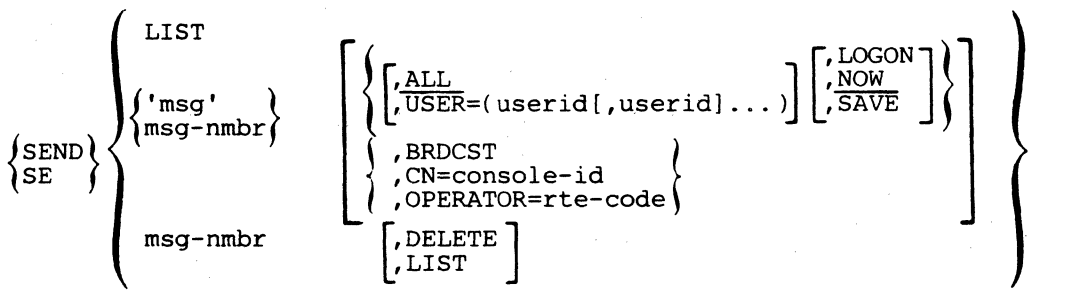
```
mn jobnames,t
```

SEND Subcommand of OPERATOR

Use the SEND subcommand to send a message to one or more terminal users, to save a message in the SYS1.BRODCAST data set, to list, delete, or send a specified message from the notices section of SYS1.BRODCAST, and to list all messages in the notices section of SYS1.BRODCAST. Messages sent via the SEND subcommand will have the characters OPER appended to the message text before it is directed to a user terminal.

Messages are also sent to the console operator and other terminals in operator mode. The characters specified in the CN parameter are appended to a message sent to a console operator.

The syntax of the SEND subcommand of OPERATOR is:



- The maximum length of the operand of the SEND subcommand is restricted to 124 characters.

LIST
: a list of all messages stored in the notices section of SYS1.BRODCAST is displayed at your terminal. (Each message displayed is preceded by a system-assigned number.)

'msg'
: the message to be sent. Enclose the text of the message within single quotation marks, and ensure that it is a one-line message. If you want a quotation mark as part of the message, enter two quotation marks in the original text.

  value: variable-length character string

msg-nmbr
: the identification number of a message in the notices section of SYS1.BRODCAST. (The identification number is system-assigned.) If you specify *msg-nmbr* with no other operands, the associated message in the notices section is sent.

  value: an integer

ALL
: the message is to be sent to all terminal users.

USER
the message is to be sent to the indicated terminal user(s)

userid
the user identification of one or more terminal users who are to receive the message. (The maximum number of userids allowed is 20.)

value: 1-7 alphameric characters, beginning with an alphabetic or a national character

LOGON
the message is sent immediately to the terminal user(s) logged on and accepting messages; otherwise,

- users logged on, but not receiving messages, receive it upon requesting messages.
- if you specify ALL, the message is stored in the notices section of SYS1.BRODCAST; sent to every user as he logs on and requests messages; and retained in SYS1.BRODCAST until you delete it.
- if you specify USER, the message is stored in the mail section of SYS1.BRODCAST; sent to each indicated user as he logs on and requests messages; and deleted by the system after all indicated users have received it.

NOW
the message is sent immediately

- if you specify ALL, the message is sent to all terminal users currently logged on, and then deleted by the system
- if you specify USER, the message is sent to the indicated terminal user(s) currently logged on. If any indicated terminal user is not logged on, you are notified; and the system deletes the message.

SAVE
the message is stored in the appropriate section of SYS1.BRODCAST. The message is not sent immediately, even to those terminal users currently logged on and receiving messages.

- if you specify ALL, the message is stored in the notices section of SYS1.BRODCAST, and assigned an identification number by the system. The identification number is displayed at your terminal. The message is sent to terminal users as they log on and request messages; and retained in SYS1.BRODCAST until you delete it.
- if you specify USER, the message is stored in the mail section of SYS1.BRODCAST, and sent to the indicated terminal user(s) as they log on and request messages. After the last indicated user has received the message, it is deleted by the system.

BRDCST
the message is queued to all active operator consoles

CN
the message is queued to a particular operator console

console-id
the identification number of an operator console. If you specify CN=0 or an invalid id, the message is sent to the master console. (You can use the console configuration message issued at IPL for reference to determine the valid and appropriate ids.)

value: an integer in the range of 0-99

OPERATOR
the message is queued to the console associated with the routing code

rte-code
the identification number of a functional console

value: as shown in the following table

Routing Code	Console
1	Master console action
2	Master console information
3	Tape pool
4	Direct access pool
5	Tape library
6	Disk library
7	Unit record pool
8	Teleprocessing control
9	System security
10	System error/maintenance
11	Programmer information
12	Emulators
13	Reserved for your use
14	Reserved for your use
15	Reserved for your use
16	Reserved for future expansion

DELETE
the message identified by *msg-nmbr* is deleted from the notices section of SYS1.BRODCAST

LIST
the message identified by *msg-nmbr* is displayed at your terminal

Example 1

Operation: Send a message to all terminal users currently logged on.

```
send 'tso to shut down at 9:55 p.m. est 9/14/70'
```

Example 2

Operation: Send a message to two particular terminal users currently logged on.

```
send 'your acct no. invalid after this session',user=(heus75,jul65)
```

Example 3

Operation: Delete a message.

```
send 8, delete
```

Example 4

Operation: Have all messages displayed at your terminal.

```
send list
```

SLIP Subcommand of OPERATOR

Use the SLIP subcommand to control SLIP (serviceability level indication processing), a diagnostic aid designed to intercept or trap certain system events. You can indicate what kinds of events you want trapped and what the system should do when these events occur.

The kinds of events you can intercept are:

- Program event recording (PER) events
  - Instruction fetch PER interruption
  - Successful branch PER interruption
  - Storage alteration PER interruption
- Error events
  - Paging error
  - Dynamic address translation error
  - Machine check associated software error
  - Address space termination error
  - SVC 13 issued by a task
  - SVC error
  - Program check interruption
  - Restart interruption

When one of these events occurs, you can take one of the following actions:

- Request an SVC dump tailored specifically to your needs
- Cause a GTF trace record to be written
- Suppress dumps (for error events only)
- Ignore the event

The PER and error events you can trap are quite general, and you probably do not want to take one of those actions each time such an event occurs. To narrow the scope of SLIP processing, qualify the event by specifying exactly what state the system must be in when the error or PER event happens for the action to occur. The system checks each specified condition to see if it corresponds to the system condition at the time of the error or PER interruption. The conditions you specify serve as filters to screen out those events you are not interested in. When conditions specified are the same as those in the system, a match occurs. When conditions specified are not the same as those in the system, a no-match occurs. Only when all the conditions you specify match will your action be taken. Among the conditions you can specify are:

- The type of error the system is processing
- The system mode at the time of the error or PER interruption
- A user or system completion code associated with the error
- The name of a job that must be in control at the time of the error or PER interruption

- The name of the job step program that must be in control at the time of the error or PER interruption
- The module name or address range where the error or PER interruption must occur
- The address space that must be in control at the time of the error or PER interruption
- The contents of specific storage locations and/or registers at the time of the error or interruption

If you do not specify a particular condition, then the system makes no checks for that condition.

There are three types of SLIP subcommands:

- SLIP SET subcommand defines SLIP traps.
- SLIP MOD subcommand enables or disables previously defined SLIP traps.
- SLIP DEL subcommand deletes previously defined SLIP traps.

For more information about designing an effective SLIP trap, see Diagnostic Techniques.

PER Monitoring

You can control in which address space, or spaces, PER is active through the specifications of ASID, JOBNAME, and MODE=HOME. The following matrix illustrates the effects of different combinations of specifications. (Note: For information concerning cross memory services and the definitions of particular address spaces, for example, HOME or PRIMARY, refer to **SPL: System Macros and Facilities, Volume 1.**)

ASID	JOBNAME	MODE = HOME	Effect
NO	NO	NO	PER active in all address spaces
NO	YES	NO	PER active in any address space in which the specified job is running
NO	NO	YES	PER active in any address space in which a unit of work is dispatched
NO	YES	YES	PER active in any address space in which the specified job is dispatched
YES	NO	NO	PER active only in the specified address space(s)
YES	YES	NO	PER active in any of the specified address spaces in which the specified job is running
YES	NO	YES	PER active in any of the specified address spaces in which a unit of work is dispatched
YES	YES	YES	PER active in any of the specified address spaces in which the specified job is dispatched
•NO - parameter not specified •YES - parameter specified			

Parameter Relationships

SLIP SET parameters fall into six functional groups: trap-type parameters, event filter parameters, action related parameters, trap control parameters, dump and trace tailoring parameters, and specialized parameters.

The trap-type parameters are IF, SA, and SB. Each defines a specific type of PER interruption trap. Omitting all trap-type parameters also has meaning: defines an error detection (or non-PER) trap.

The event filter parameters are ADDRESS, ASID, ASIDSA, COMP, DATA, ERRTYP, JOBNAME, JSPGM, LPAMOD, MODE, NUCMOD, PVTMOD, RANGE, and REASON. Event filter parameters define the scope of the event, or events, the trap is to monitor.

The action related parameter is ACTION. The operands of ACTION (IGNORE, NODUMP, NOSUP, NOSVCD, NOSYSA, NOSYSM, NOSYSU, RECOVERY, SVCD, TRACE, and TRDUMP) specify what you want done when the trap matches.

The four trap control parameters are DISABLE, ENABLE, MATCHLIM, and PRCNTLIM. They control the operation of the trap by indicating whether the trap is active; how many times the trap should match and produce the desired action before it is automatically disabled; or what percentage of the system can be used when monitoring a PER trap.

The dump tailoring parameters are ASIDLST, LIST, SDATA, and SUMLIST. The trace tailoring parameter is TRDATA. The parameters enable you to tailor the contents of a dump or a trace record.

The specialized parameters are DEBUG, END, ID, and RBLEVEL. DEBUG is used to diagnose a SLIP trap that is apparently not working according to your specifications; it indicates that you want some trap information recorded each time the trap is checked rather than each time it matches. END marks the end of a SLIP SET command. ID assigns an identifier to a trap. RBLEVEL indicates which Request Block the system is to use for error detection traps.

Indirect Addressing Used with SLIP

Indirect addressing used with SLIP is similar to that used with TSO TEST except:

- unlimited levels of indirection are permitted
- symbols are not allowed
- absolute addresses are not followed by a period
- address modifiers must be hexadecimal

You can use indirect addresses with the following SLIP command parameters: DATA, LIST, SUMLIST, and TRDATA. The addresses refer to the address space in which the event occurs unless an address space identifier is specified. For DATA, SUMLIST, and TRDATA, the storage areas referred to must be paged in. If they are paged out, they will be paged in only if the trap is non-PER and the system at the time of error was unlocked, enabled, and without any EUT FRRs; otherwise they are ignored. For LIST, the storage areas referred to when resolving the indirect address must be paged in (except for the non-PER, unlocked, enabled, and no EUT FRR case); but the storage areas to be dumped will be paged in if they are paged out.

The elements of an indirect address are:

1. a direct address: 1-8 hexadecimal digits optionally followed by one or more displacements.
2. a displacement: a plus (+) or minus (-) sign followed by 1-4 hexadecimal digits.
3. a general purpose register: xR where x is an integer in the range 0-15.
4. an indirection indicator or pointer: a percent sign (%) indicating a 24-bit address or a question mark (?) indicating a 31-bit address. A pointer is always 4 bytes long. (The high-order byte is ignored for 24-bit addresses.)

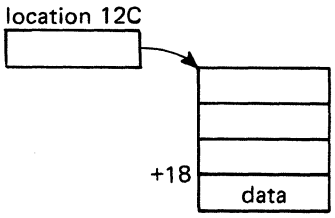
An indirect address is either of the following forms:

```
{ direct-address {%} [ % ]...[[displacement]...[ % [ % ]...]]...  }
{                {?} [ ? ]    [                  ? [ ? ]    ]]    }
{ xR             {%} [ % ]...[[displacement]...[ % [ % ]...]]...  }
{                {?} [ ? ]    [                  ? [ ? ]    ]]    }
```

The following expressions illustrate some indirect addresses.

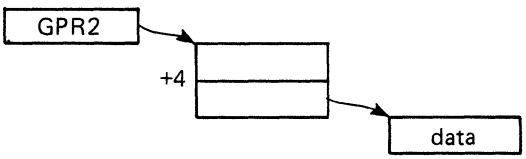
```
12C%+4+8+C
```

Graphically:



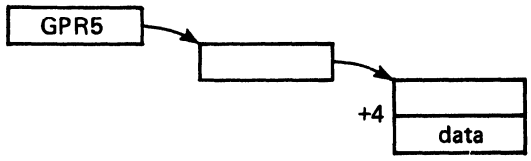

```
2R?+4?
```

Graphically:



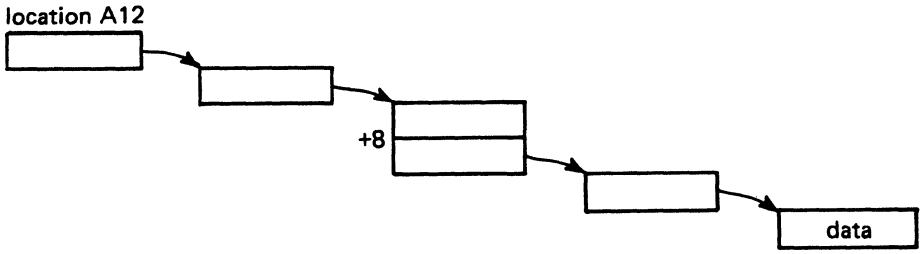
5R%%+4

Graphically:

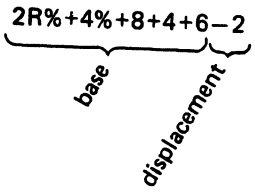


A12%?+8?

Graphically:



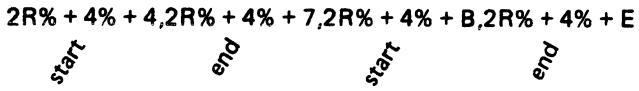
Each complete address is composed of two parts: the base and the displacement. The base is defined as everything except the last displacement. For example, the address 2R%+4%+8+4+6-2 has the following component parts:



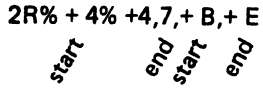
If a complete address is entered without a final + or - displacement, a +0 is assumed. For example, 2R%+4% is treated as 2R%+4%+0.

The following discussion applies to the LIST, SUMLIST, and TRDATA parameters when multiple *start,end* specifications are made.

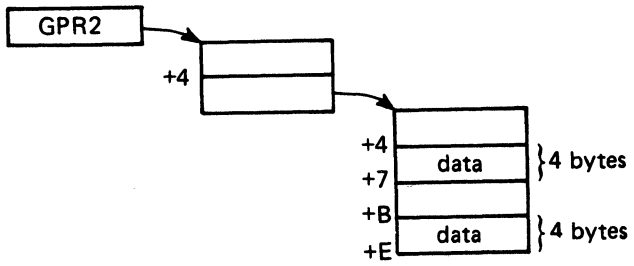
After entering the first complete address (direct or indirect), you can use a form of shorthand for subsequent addresses. The first address establishes the base address. Subsequent addresses are written as plus or minus displacements from the base and are separated by commas. For example, the following set of addresses



can be written using the shorthand form as



Graphically:



Note that the shorthand form allows you to use control block offsets directly from the **Debugging Handbook** without performing any calculations.

The following discussion applies to the DATA parameter when multiple *target* specifications are made.

After entering the first target address (direct or indirect), you can use a form of shorthand for subsequent target addresses. The first target address establishes the base address. Subsequent target addresses are written as plus or minus displacements from the base. For example, the following

```
2R% + 4,EQ,A24,2R% + 8,NE,B66
target operator preval target operator preval
```

can be written using the shorthand form as

```
2R% + 4,EQ,A24, +8,NE,A66
target operator preval target operator preval
```

However, if you specify the first target as the contents of a general purpose register, the shorthand form is invalid. For example, the following specification is invalid.

```
2R,EQ,C12,+6,NE,D01
```

Once the base address is established, subsequent target addresses may include both the shorthand form and the contents of a general purpose register. For example, the following specification is valid.

```
2R%+4,NE,D11,5R,EQ,15R,+6,GT,C10
```

An address space identifier may be specified as a prefix to a direct or indirect address. If the address space identifier is not specified, the address refers to the address space in which the event occurs. Once an identifier is specified, that address and subsequent addresses within the same keyword are fetched from the address space associated with the identifier, until another identifier is specified. An address space identifier may be:

1. 1-4 hexadecimal digits representing an

   explicit ASID

2. one of the following symbolics

   HASID or H - home address space

   PASID or P - primary address space

   SASID or S - secondary address space

   CURRENT or CU - current address space

The following expressions illustrate the use of address space identifiers:

```
6.
A.12C%+8%+4
4B.6R%+C%
PASID.12R%+4%+1C
```

In the following expression:

```
6R%+4,+7,SASID.8R%+C,+F,7R%,+1F,+30,+37,H.21C%+10,13
```

* The address space used to resolve the 6R%+4,7 addresses is the address space in which the event occurred.
* The address space used to resolve the 8R%+C,+F,7R%,+1F,+30,+37 addresses is the secondary address space at the time event occurred.
* The address space used to resolve the 21C%+10,13 addresses is the home address space at the time the event occurred.

Note: The specification of an address space identifier as a prefix to the shorthand form of an indirect address is invalid; for example,

```
7R%,+1F,PASID.+30,+37
```

is an invalid specification.

Parameter Descriptions

The major positional and keyword parameters are described in alphabetical order. The subparameters are described under the major parameters in alphabetical order.

ACTION or A
the action that is to occur when a trap matches.

IGNORE
the system is to resume normal processing when the trap matches

NODUMP
the following dumps are suppressed

- SVC dumps requested by ESTAE and FRRs
- all SYSABEND, SYSUDUMP, and SYSMDUMP dumps

Note:

1. When you specify ACTION=NODUMP, ensure that the SLIP trap is specific. If the trap is too general, you might suppress dumps needed for other problems. For example, if you specify only a system completion code, all dumps for that code are suppressed. However, if you specify both a completion code and a jobname, other jobs that abend with that completion code produce dumps.

2. If a second error occurs during processing for an event with ACTION=NODUMP specified, any dump requested for the second error is also suppressed. You can determine if a second error occurred by checking both the job output messages and SYS1.LOGREC output. If either one indicates more than one abend, a second error occurred. If you need a dump for the second error, disable the SLIP trap that specifies ACTION=NODUMP and rerun the failing job.

NOSUP

1. override the action of dump analysis and elimination (DAE) in suppress mode and do *not* suppress any duplicate SVC or SYSMDUMP dumps.
2. override the specification of dump suppression requested by a user ABDUMP predump exit and do *not* suppress any SYSUDUMP or SYSABEND dumps.

NOSVCD
suppress only SVC dumps requested by ESTAE and FRRs when the trap matches.

NOSYSA
suppress only requested SYSABEND dumps when the trap matches.

NOSYSM
suppress only requested SYSMDUMP dumps when the trap matches.

NOSYSU
suppress only requested SYSUDUMP dumps when the trap matches.

RECOVERY
force PER traps to initiate recovery processing for the interrupted process after the specified action is taken. (System completion code 06F is generated.)

Note: Use the RECOVERY keyword carefully to avoid unexpected results. Before using RECOVERY, be thoroughly familiar with MVS recovery principles. In particular, make sure that recovery procedures exist at the point where you are forcing recovery processing. Know what the recovery routines will do under the circumstances in which you are forcing recovery processing.

SVCD
schedule an SVC dump when the trap matches. (If an address space is failing and you did not specify a list of ASIDs to be dumped, SLIP tries to dump the failing address space. If the failing address space cannot be dumped, SLIP dumps the home address space.) The dump can be tailored using the keywords ASIDLST, LIST, SDATA, and SUMLIST.

The specification of SVCD always overrides the duplicate dump suppression action requested through DAE.

TRACE
create a SLIP GTF trace record when the trap matches. The record may be written to external storage or maintained in virtual storage according to the GTF options selected. (For detailed information on GTF functions, refer to **SPL: Service Aids.**) If you do not specify TRDATA, a SLIP standard trace record is created when the trap matches. (For TRACE to be active, GTF with the SLIP option must be active.)

TRDUMP
create a SLIP GTF trace record each time a trap matches and schedule an SVC dump when the trap is disabled or deleted. The dump can be tailored using the keywords ASIDLST, LIST, SDATA, and SUMLIST. If you do not specify TRDATA, a SLIP standard trace record is created when the trap matches. (For TRDUMP to be active, GTF with the SLIP option must be active.)

The specification of TRDUMP always overrides the duplicate dump suppression action requested through DAE.

TRDATA or TD
tailor the type and contents of a SLIP GTF trace record

STD
create a standard SLIP GTF trace record when the trap matches

REGS
collect the contents of the 16 GPRs into the SLIP GTF trace record when the trap matches

asid
an address space identifier (If *asid* is not specified, current is assumed.)

value: 1. 1-4 hexadecimal digits (The value specified must not exceed the maximum value set by your installation.)

2. as indicated in the following list:

HASID or H	-home address space
PASID or P	-primary address space
SASID or S	-secondary address space
CURRENT or CU	-current address space

start,end
collect the contents of the address range, or ranges, from the address space, or spaces, into the SLIP GTF trace record when the trap matches. The address range can be:

1. a virtual address (direct address)

value: 1-8 hexadecimal digits for *start* and *end*. (The ending address must be greater than or equal to the starting address.)

2. an indirect address

value: refer to the paragraph 'Indirect Addressing Used with SLIP'. (The ending address must be greater than or equal to the starting address.)

Each address range must not be larger than 65,535 bytes. If any range exceeds 65,535 bytes, no data is written in the trace record but a zero-length indicator is to indicate the error.

ASIDLST or AL
the address space, or spaces, to be dumped when the trap matches.

n
an address space identifier. (The maximum number of identifiers allowed is 15.)

value: 1. 1-4 hexadecimal digits. (A value of 0 indicates the home address space. The value specified must not exceed the maximum value set by your installation.)

2. as indicated in the following list:

HASID or H	-home address space
PASID or P	-primary address space
SASID or S	-secondary address space
CURRENT or CU	-current address space
LLOC	-locked address space

LIST or LS
the address range, or ranges, from the address space, or spaces, to be included in an SVC dump when the trap matches

asid
an address space identifier (If *asid* is not specified, current is assumed.)

value: 1. 1-4 hexadecimal digits (The value specified must not exceed the maximum value set by your installation.)

2. as indicated in the following list:

HASID or H	-home address space
PASID or P	-primary address space
SASID or S	-secondary address space
CURRENT or CU	-current address space

start,end
the starting and ending addresses. The address range can be:

1. a virtual address (direct address)

value: 1-8 hexadecimal digits for *start* and *end*. (The ending address must be greater than or equal to the starting address.)

2. an indirect address

value: refer to the paragraph 'Indirect Addressing Used with SLIP'. (The ending address must be greater than or equal to the starting address.)

Two error conditions may arise when using LIST. The first involves the resolution of an indirect address. If, for any reason, an indirect address cannot be converted to a direct address (for example, a page fault occurs while retrieving a pointer or registers are unavailable for conversion), the characters *RC=4* are dumped instead of the address range requested to indicate that the address pair was not successfully converted. The second condition occurs when a pair of indirect addresses are successfully converted but the second address (ending address) is less than the first address (starting address) of the pair. The characters *A1>A2* are dumped instead of the address range requested (and SDUMP is prevented from abending).

SDATA or SD
the system control information to be included in an SVC or summary dump when the trap matches

option
an area of storage or type of dump

value: as indicated in the following list:

ALLNUC	- all of the DAT-on and DAT-off nuclei
ALLPSA	- prefix storage area for all CPUs
CSA	- common storage area
GRSQ	- global resource serialization queues
LPA	- link pack area
LSQA	- local system queue area
NOALLPSA or NOALL	- not ALLPSA
NOSQA	- not SQA
NOSUMDUMP or NOSUM	- not SUMDUMP
NUC	- only the non-page protected part of the DAT-on nucleus
PSA	- prefix storage area of dumping CPU
RGN	- entire private area
SQA	- system queue area
SUMDUMP or SUM	- summary dump function
SWA	- scheduler work area
TRT	- GTF or supervisor trace data

- For ACTION=SVCD, if SDATA is not specified, the following is assumed:

  ```
  SDATA=(ALLPSA,CSA,LPA,NUC,RGN,SQA,SUM,TRT)
  ```

- For ACTION=TRDUMP, if SDATA is not specified, the following is assumed:

  ```
  SDATA=(TRT)
  ```

- If any SDATA options are explicitly specified, any alterations to those options specified on the CHNGDUMP command are ignored. However, if CHNGDUMP is set with the NODUMP option, no dump is produced when the trap matches. (For more detailed information relative to SDATA, CHNGDUMP, and SDUMP, refer to **MVS Diagnostic Techniques.**)

SUMLIST or SL
the address range, or ranges, from the address space, or spaces, to be included in a summary dump when the trap matches. (If SDATA=(NOSUMDUMP) is specified, the specification of SUMLIST is invalid.)

asid
an address space identifier (If *asid* is not specified, current is assumed.)

value: 1. 1-4 hexadecimal digits (The value specified must not exceed the maximum value set by your installation.)

2. as indicated in the following list:

HASID or H	-home address space
PASID or P	-primary address space
SASID or S	-secondary address space
CURRENT or CU	-current address space

start,end
an address range. The address range can be:

1. a virtual address (direct address)

value: 1-8 hexadecimal digits for *start* and *end*. (The ending address must be greater than or equal to the starting address.)

2. an indirect address

value: refer to the paragraph 'Indirect Addressing Used with SLIP'. (The ending address must be greater than or equal to the starting address.)

Two error conditions may arise when using SUMLIST. The first involves the resolution of an indirect address. If, for any reason, an indirect address cannot be converted to a direct address (for example, a page fault occurs while retrieving a pointer or registers are unavailable for conversion), the characters *RC=4* are dumped instead of the address range requested to indicate that the address pair was not successfully converted. The second condition occurs when a pair of indirect addresses are successfully converted but the second address (ending address) is less than the first address (starting address) of the pair. The characters *A1>A2* are dumped instead of the address range requested (and SDUMP is prevented from abending).

ADDRESS or AD
the event must occur at a virtual address, or within a range of virtual addresses, to satisfy the match test

start
a virtual address (1-byte range)

values: 1-8 hexadecimal digits

start,end
a virtual address range

value: 1-8 hexadecimal digits for *start* and *end* (The ending address must be greater than or equal to the starting address.)

- For more information on choosing the virtual address or address range relative to the environment, refer to *Diagnostic Techniques*.

ASID or AS
the event must occur within an address space, or spaces, to satisfy the match test. For storage alteration PER traps, this keyword refers to the address space, or spaces, from which the instructions are fetched. Refer to the paragraph 'PER Monitoring' for additional information.

id
an address space identifier. (The maximum number of ids allowed is 16.)

value: 1-4 hexadecimal digits. (The value specified must not exceed the maximum value set by your installation.)

ASIDSA or ASA
the storage being altered must reside within an address space, or spaces, to satisfy the match test.

asid
an address space identifier (The maximum number of ids allowed is 16.)

value: 1. 1-4 hexadecimal digits (The value specified must not exceed the maximum value set by your installation.)

2. as indicated in the following list:

HASID or H	-home address space
PASID or P	-primary address space
SASID or S	-secondary address space
CURRENT or CU	-current address space

COMP or C
a system or user completion code to be associated with an error. If you specify a set of codes, the occurrence of any one satisfies the match test.

hhh
a system completion code or a set of system completion codes

value: 1. 3 hexadecimal digits (a unique code)

2. 0-2 hexadecimal digits and 1-3 occurrences of X (a set of codes), valid specifications - XxX, xXX, xxX, XXx, xXx, Xxx, XXX (For example, X11 means 011, 111, ..., F11)

Note: If you specify any of the following system completion codes, the match test always fails:

```
11A,12E,15D,15F,200,212,279,25F,282,402,42A,57D,6FC,700,72A,A00,B00,
```

Most of the preceding codes occur originally as a program check (0C4) and are converted to the indicated code by the system. SLIP can detect the original code (0C4), but not the converted code. To specify a program check, use COMP=0C4 or ERRTYP=PROG. To avoid satisfying the match test for all program checks, specify a program name, module name, or other qualifier.

In addition, the specification of 13E or 33E prevents a trap match because those completion codes occur for any active subtasks associated with a task that is abending. The secondary abends occur for the purpose of clean-up only and are not detected by SLIP.

Udddd
a user completion code or a set of user completion codes

value: 1. 4 decimal digits (a unique code)

2. 0-3 decimal digits and 1-4 occurrences of X (a set of codes), valid specifications -

UdddX	UXXdd	UddXd	UXXXX
UddXX	UXXXd	UXXdX	UXdXd
UdXXX	UdXdd	UXdXX	UdXdX
UXddd	UdXXd	UXddX	

Note: If any user completion code is changed in a user recovery routine with the SETRP macro instruction, specify the original completion code in the COMP keyword parameter.

REASON or RE
associate a reason code with the error

code
a reason code

value: 1. 1-8 hexadecimal digits (a unique code)

2. 1-7 hexadecimal digits and 1-7 occurrences of X. SLIP ignores the digit(s) of a reason code specified as an X.

Note: 1. The specification of REASON is valid only if the REASON parameter was coded on the ABEND or CALLRTM macro instruction.

2. If *code* is less than eight digits, it is padded on the left with zeroes. For example, REASON=4 is stored as 00000004; REASON=XX0X1C is stored as 00XX0X1C.

DATA or DA

logically compare the contents of a target location to a specified value. All comparisons must be successful to satisfy the match test. (If a DATA= target location is paged out, a no match is assumed. You are notified by message IEA413I, and a 'data unavailable' count is updated in the SCVA. For a PER trap, you are notified only the first time the data is unavailable. However, the 'data unavailable' count is readily available by displaying the trap or in the standard portion of a SLIP trace record.)

asid

an address space identifier (If *asid* is not specified, current is assumed.)

value: 1. 1-4 hexadecimal digits (The value specified must not exceed the maximum value set by your installation.)

2. as indicated in the following list:

HASID or H	-home address space
PASID or P	-primary address space
SASID or S	-secondary address space
CURRENT or CU	-current address space

target

the address of a storage location or a general purpose register (GPR). The target can be:

1. a virtual address (direct address)

   value: 1-8 hexadecimal digits

2. a GPR in the form *x*R

   x

   a register designation

   value: an integer in the range of 0-15

3. an indirect address

   value: refer to the paragraph 'Indirect Addressing Used with SLIP'

b

target modifier that indicates where a binary comparison is to start

value: 1. an integer in the range of 0-7 (storage locations)

2. an integer in the range of 0-31 (GPRs)

operator

a logical operator

value: as indicated in the following list:

EQ - equal
NE - not equal
GT - greater than
LT - less than
NG - not greater than (less than or equal to)
NL - not less than (greater than or equal to)

preval
the data to which the contents of *target* are to be compared

value: 1. binary digits - maximum length of 8 bits (*b* specified). (The comparison can be across a byte boundary, but not across a register boundary.)

2. hexadecimal digits -maximum length of 4 bytes, right justified (*b* not specified)

- The length of *preval* determines the length of the compare except for the hexadecimal compare of a register. In that case, *preval* is right justified, padded with zeros, and the compare length is the entire register.

DEBUG
provides information to allow you to determine why a trap you set is not working as you expected. (For DEBUG to be active, GTF with the SLIP option must be active.)

Each time the trap is tested a trace record, containing the standard SLIP trace data plus two bytes of match/no match bit indicators, is written. Each bit corresponds to a possible test made to determine a match for the trap. If a test is successful, the corresponding indicator is set to 0. If a test is unsuccessful, the corresponding indicator is set to 1. After the first unsuccessful test, no further tests are made and all the remaining indicators are set to 0. For a description of the SLIP DEBUG trace record and the bit indicators, refer to **SPL: Debugging Handbook, Volume 1.**

DISABLE or D
a defined SLIP trap is initially disabled

ENABLE or EN
a defined SLIP trap is initially enabled

END or E
the end of a SLIP command

ERRTYP or ER
an error condition must occur to satisfy the match test. If you specify more than one error condition, the occurrence of any one satisfies the match test. If you do not specify ERRTYP, ALL is assumed.

type
an error condition

value: as indicated in the following list:

ALL	-	all of the following error conditions
ABEND	-	task issued SVC13
DAT	-	dynamic address translation error
MACH	-	software error caused by machine check
MEMTERM	-	abnormal address space termination
PGIO	-	paging I/O error
PROG	-	program check interruption
REST	-	restart interruption
SVCERR	-	SVC error (issuing an SVC while holding a lock, executing disabled, or executing in SRB mode)

ID
an identifier is to be assigned to a trap. If you do not specify ID, the system assigns a unique four-character identifier beginning with 0001. You are notified of the system-assigned identifier by message IEE727I.

xxxx
trap identifier

value: 1-4 alphameric characters

IF
monitor an instruction fetch PER trap

JOBNAME or J
the initiated job, started task, or TSO session that must be in control to satisfy the match test. Refer to the paragraph 'PER Monitoring' for additional information.

j-name
the jobname, started task id, or TSO userid. (The *jobname* is the one specified on the JOB statement; the started task id is the *procname* specified on the START command; the TSO userid is the *userid* specified on the LOGON command.)

value: 1. 1-8 alphameric characters, beginning with an alphabetic or a national character (jobname and started task id)

2. 1-7 alphameric characters, beginning with an alphabetic or a national character (TSO userid)

JSPGM or JS
the job step program that must be in control to satisfy the match test. If you specify JSPGM for an error trap and any address space abnormally terminates, a no-match condition for the trap occurs.

js-name
the job step program name. (This name is the one specified on the EXEC statement in the PGM=*program-name* parameter.)

value: 1-8 alphameric characters, beginning with an alphabetic or a national character

LPAMOD or L
the event must occur within a link pack area load module to satisfy the match test. If *start* and *end* are not specified, the monitored range is the entire module. If only *start* is specified, the monitored range is one byte at the offset 'start'. If both *start* and *end* are specified, the monitored range is between the offset 'start' and the offset 'end'.

mod-name
the module name

value: 1-8 alphameric characters

start
the offset into the module (1-byte range)

value: 1-6 hexadecimal digits

start,end
the starting and ending offsets into the module

value: 1-6 hexadecimal digits for *start* and *end*. (The value specified for *end* must be greater than or equal to the value specified for *start*.)

- For more information on choosing the *start* and *end* offsets, refer to *Diagnostic Techniques*.

MATCHLIM or ML

a SLIP trap is automatically disabled after the specified number of matches. If you specify ACTION=SVCD for a PER trap and do not specify MATCHLIM, 1 is assumed. For all other traps, if you do not specify MATCHLIM, no limit is assumed; that is, no MATCHLIM test is made.

When the specified number of trap matches occurs, an enabled trap is disabled and message IEA411I is issued. If you specified ACTION=TRDUMP, an SVC dump is scheduled.

m

number of trap matches

value: integer in the range of 1-65535

MODE or M

the system must be in a particular mode, or modes, to satisfy the match test. If you do not specify MODE, ALL,ANY is assumed

cond

a system mode

value: as indicated in the following list:

HOME	-	executing in the home (dispatched) address space
ALL	-	all of the following modes
DIS	-	physically disabled for I/O and external interruptions
GLOC	-	holding any global lock
GLOCSD	-	holding a global suspend lock
GLOCSP	-	holding a global spin lock
LLOC	-	holding a local lock
LOCK	-	holding any lock
PKEY	-	problem program key (key 8 or higher)
PP	-	problem program
RECV	-	recovery routine in control (RECV is an invalid specification for all PER traps.)
SKEY	-	system key (key 0-7)
SRB	-	SRB mode
SUPER	-	supervisor state
SUPR	-	supervisor control mode (any bit set in PSASUPER)
TCB	-	TCB mode
TYP1	-	type 1 SVC in control

ANY

any mode specified must occur to satisfy the match test, except when HOME is specified along with other modes and ANY. In that case, the unit of work must have been executing in the home address space when the event occurred and at least one of the other specified modes must occur to satisfy the match test

EVERY

every mode specified must occur to satisfy the match test

NUCMOD or N

the event must occur within a load module in the nucleus to satisfy the match test. If *start* and *end* are not specified, the monitored range is the entire module. If only *start* is specified, the monitored range is one byte at the offset 'start'. If both *start* and *end* are specified, the monitored range is between the offset 'start' and the offset 'end'.

mod-name

the module name

value: 1-8 alphameric characters

start
the offset into the module (1-byte range)

value: 1-6 hexadecimal digits

start, end
the starting and ending offsets into the module

value: 1-6 hexadecimal digits for *start* and *end*. (The value specified for *end* must be greater than or equal to the value specified for *start*)

- For more information on choosing the *start* and *end* offsets relative to the environment, refer to :cit.Diagnostic Techniques:ecit..

PRCNTLIM or PL
the percentage limit of system processing that can be devoted to monitoring PER traps. At least 33.55 seconds must have elapsed since the first PER interruption before a trap is disabled because of this limit. If you do not specify PRCNTLIM for a non-ignore PER trap, 10 is assumed

When the processing limit is surpassed, the non-ignore enabled PER trap is disabled and message IEA411I is issued. If you specified ACTION=TRDUMP, an SVC dump is scheduled.

p
the percentage limit

value: an integer in the range of 1-99. (The value computed to test PRCNTLIM is an approximation. In addition, the value computed is truncated to an integer before the test is made.)

Note: Use caution in specifying a percentage limit of 99 because no percent limit checking is performed.

PVTMOD or P
the event must occur within a private area load module to satisfy the match test. If *start* and *end* are not specified, the monitored range is the entire module. If only *start* is specified, the monitored range is one byte at the offset 'start'. If both *start* and *end* are specified, the monitored range is between the offset 'start' and the offset 'end'.
If PVTMOD is specified for an error trap, certain conditions cause the match test to fail when executing in non task mode.

- local lock not held or obtained prior to the search of the CDE chain
- MEMTERM error
- DAT error

In addition, if the private area load module is not executing in the home (dispatched) address space, the match test will fail.

mod-name
the module name

value: 1-8 alphameric characters

start
the offset into the module (1-byte range)

value: 1-6 hexadecimal digits

start,end
the starting and ending offsets into the module

value: 1-6 hexadecimal digits for *start* and *end*. (The value specified for *end* must be greater than or equal to the value specified for *start*.)

- For more information on choosing the *start* and *end* offsets relative to the environment, refer to *Diagnostic Techniques*.

RANGE or RA
the event must occur at a virtual address, or within a range of virtual addresses, to satisfy the match test. If you specify IGNORE in the ACTION keyword parameter for a PER storage alteration trap, the specification of RANGE is invalid.

start
a virtual address (1-byte range)

value: 1-8 hexadecimal digits

start,end
a virtual address range

value: 1-8 hexadecimal digits for *start* and *end*. (No test is made to ensure that *end* is greater than *start* because the specification of a starting address greater than an ending address is valid. (See **IBM System/370 Principles of Operation** for detailed information.)

RBLEVEL or RB
the registers for use in resolving indirect addresses and the PSW for use by LPAMOD, PVTMOD, ADDRESS, and MODE are taken from a particular RB. (RBLEVEL applies only to unlocked task mode errors.) If the RB specified by RBLEVEL is not found, a no-match condition for the trap exists.

ERROR
the PSW is obtained from the RB prior to the SVC 13 (ABEND) RB (RB2 in Figure 16). The registers are obtained from the SVC 13 RB (RB1 in Figure 16).

PREVIOUS
the PSW and registers are obtained from one RB prior to the one used in ERROR. (PSW from RB3 in Figure 16; registers from RB2 in Figure 16.)

NOTSVRB
the PSW is obtained from the most recent non-SVRB; the registers from the associated SVRB. (For example in Figure 16, if RB1, RB2, and RB3 are SVRBs, the PSW is obtained from RB4 and the registers from RB3.)

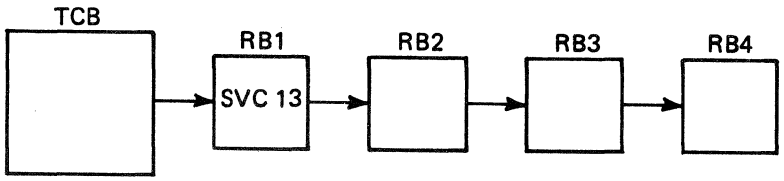


Figure 16. RB Structure

SA
monitor a storage alteration PER trap

SB
monitor a successful branch PER trap

SET
define a SLIP trap. If IF, SA, or SB are not specified, a non-PER trap (error trap) is defined and monitored.

Example 1

Operation: Setting a SLIP trap using the instruction fetch PER event.

```
sl set,if,en,action=svcd,range=(cd3100),end
```

Example 2

Operation: Setting a SLIP trap using the storage alteration PER event.

```
sl set,sa,en,action=svcd,range=(cd3010,cd3013),
data=(cd3010,eq,00000000),end
```

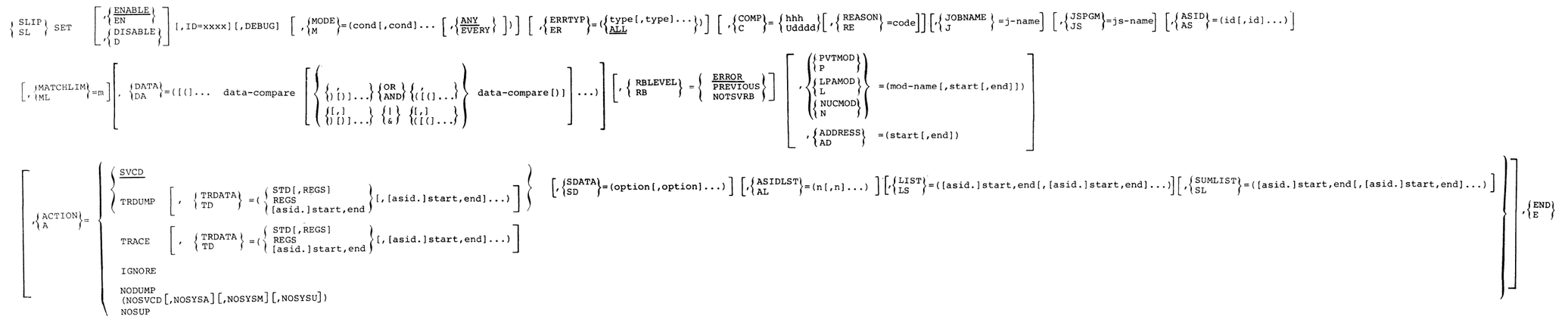
Example 3

Operation: Setting a SLIP trap to obtain a dump with queue elements and control blocks.

```
sl set,id=50cx,a=svcd,comp=x30,errtyp=abend,jspgm=comrtn,
sdata=(sqa,rgn,trt,sum),end
```

The syntax of the SLIP Subcommand of OPERATOR is

- Setting a SLIP Error (non-PER) Trap



- SET and END are positional parameters; all others are keyword parameters.
- NOSVCD, NOSYSA, NOSYSM, or NOSYSU may be specified in any order; for example,

  ACTION = (NOSYSU)
  ACTION = (NOSYSA,NOSVCD)
  ACTION = (NOSYSM,NOSYSU,NOSYSA,NOSVCD)

  are all valid specifications.
- In the DATA parameter, the elements of data-compare are: [asid.] target [(b)], operator, preval
- In the DATA parameter, a maximum of 16 levels of parentheses are allowed; that is, no more than 16 unmatched left parentheses may appear in a DATA parameter specification.

- **Setting a PER Trap for Instruction Fetch or Successful Branch**

```
{SLIP} SET,{IF} [,{ENABLE }] [,ID=xxxx] [,DEBUG] [,{MODE}=(cond[,cond]... [,{ANY  }])] [,{JOBNAME}=j-name] [,{JSPGM}=js-name] [,{ASID}=(id[,id]...)] [,{MATCHLIM}=m] [,{PRCNTLIM}=p]
{SL  }     {SB}   {EN     }                    {M   }                    {EVERY}       {J      }          {JS   }           {AS  }                {ML      }      {PL      }
                 [,{DISABLE}]
                   {D      }

[,{DATA}=([asid.]target[(b)],operator,preval[,[asid.]target[(b)],operator,preval]...)] { ,{RANGE }=(start[,end])            }
  {DA  }                                                                              {  {RA    }                          }
                                                                                      { ,{LPAMOD}=(mod-name[,start[,end]]) }
                                                                                      {  {L     }                          }

[ ,{ACTION}= { (SVCD[,RECOVERY])                                                                    } [,{SDATA}=(option[,option]...)] [,{ASIDLST}=(n[,n]...)] [,{LIST}=([asid.]start,end[,[asid.]start,end]...)] [,{SUMLIST}=([asid.]start,end[,[asid.]start,end]...)] ] ] ,{END}
   {A     }  { (TRDUMP[,RECOVERY]) [,{TRDATA}=({STD[,REGS]         }[,[asid.]start,end]...)]        }   {SD   }                          {AL     }              {LS  }                                      {SL     }                                                   {E  }
             {                       {TD    }  {REGS               }                                 }
             {                                 {[asid.]stat,end    }                                 }
             { (TRACE[,RECOVERY])  [,{TRDATA}=({STD[,REGS]         }[,[asid.]start,end]...)]        }
             {                       {TD    }  {REGS               }                                 }
             {                                 {[asid.]stat,end    }                                 }
             { (IGNORE[,RECOVERY])                                                                   }
             { RECOVERY                                                                              }
```

- SET, IF, SB, and END are positional parameters; all others are keyword parameters.
- Only one non-ignore PER trap may be enabled at any one time. If you attempt to set an enabled non-ignore PER trap while one is already enabled, the trap is defined, forced to the disabled state, and message IEE740I is issued. If you attempt to enable a non-ignore PER trap while one is already enabled, the request is denied and message IEE741I is issued.
- If you do not specify RECOVERY in conjunction with another parameter, the use of the indicated parentheses is optional.

- **Setting a PER Trap for Instruction Fetch or Successful Branch**

```
{SLIP|SL} SET,{IF|SB}[,{ENABLE|EN|DISABLE|D}][,ID=xxxx][,DEBUG]|[,{MODE|M}=(cond[,cond]...[,{ANY|EVERY}])][,{JOBNAME|J}=j-name][,{JSPGM|JS}=js-name][,{ASID|AS}=(id[,id]...)][,{MATCHLIM|ML}=m][,{PRCNTLIM|PL}=p]

[,{DATA|DA}=([asid.]target[(b)],operator,preval[,[asid.]target[(b)],operator,preval]...)]
{,{RANGE|RA}=(start[,end])
 ,{LPAMOD|L}=(mod-name[,start[,end]])}

[,{ACTION|A}={(SVCD[,RECOVERY])
              (TRDUMP[,RECOVERY])[,{TRDATA|TD}=({STD[,REGS]|REGS|[asid.]stat,end}[,[asid.]start,end]...)]
              (TRACE[,RECOVERY])[,{TRDATA|TD}=({STD[,REGS]|REGS|[asid.]stat,end}[,[asid.]start,end]...)]
              (IGNORE[,RECOVERY])
              RECOVERY}
 [,{SDATA|SD}=(option[,option]...)][,{ASIDLST|AL}=(n[,n]...)][,{LIST|LS}=([asid.]start,end[,[asid.]start,end]...)][,{SUMLIST|SL}=([asid.]start,end[,[asid.]start,end]...)]
 ,{END|E}]
```

- SET, IF, SB, and END are positional parameters; all others are keyword parameters.
- Only one non-ignore PER trap may be enabled at any one time. If you attempt to set an enabled non-ignore PER trap while one is already enabled, the trap is defined, forced to the disabled state, and message IEE740I is issued. If you attempt to enable a non-ignore PER trap while one is already enabled, the request is denied and message IEE741I is issued.
- If you do not specify RECOVERY in conjunction with another parameter, the use of the indicated parentheses is optional.

- **Setting a PER Trap for Instruction Fetch or Successful Branch**

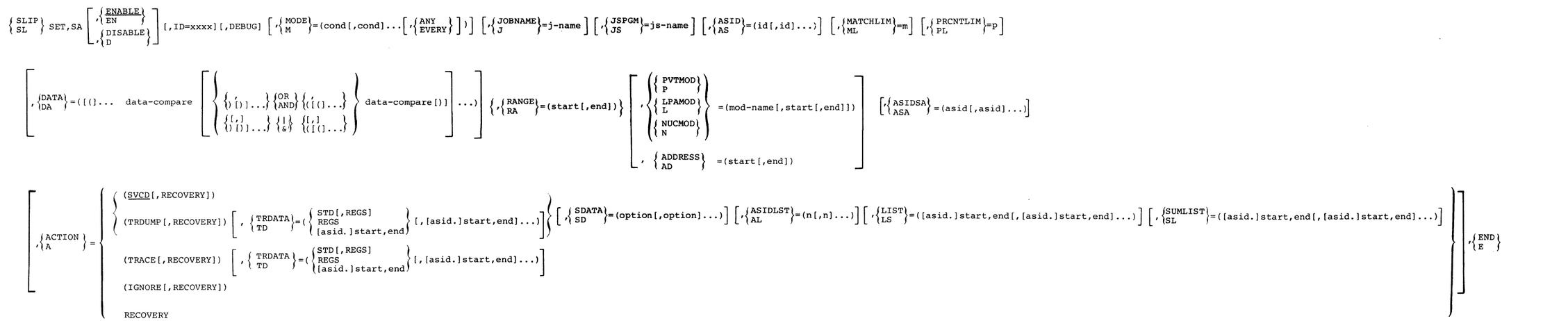
```
{SLIP|SL} SET,{IF|SB} [,{ENABLE|EN}|,{DISABLE|D}] [,ID=xxxx] [,DEBUG] [,{MODE|M}=(cond[,cond]... [,{ANY|EVERY}])] [,{JOBNAME|J}=j-name] [,{JSPGM|JS}=js-name] [,{ASID|AS}=(id[,id]...)] [,{MATCHLIM|ML}=m] [,{PRCNTLIM|PL}=p]

[,{DATA|DA}=([asid.]target[(b)],operator,preval[,[asid.]target[(b)],operator,preval]...)] {,{RANGE|RA}=(start[,end]) | ,{LPAMOD|L}=(mod-name[,start[,end]])}

[,{ACTION|A}={ (SVCD[,RECOVERY]) | (TRDUMP[,RECOVERY]) [,{TRDATA|TD}=({STD[,REGS]|REGS|[asid.]stat,end}[,[asid.]start,end]...)] } [,{SDATA|SD}=(option[,option]...)] [,{ASIDLST|AL}=(n[,n]...)] [,{LIST|LS}=([asid.]start,end[,[asid.]start,end]...)] [,{SUMLIST|SL}=([asid.]start,end[,[asid.]start,end]...)]
                    (TRACE[,RECOVERY]) [,{TRDATA|TD}=({STD[,REGS]|REGS|[asid.]stat,end}[,[asid.]start,end]...)]
                    (IGNORE[,RECOVERY])
                    RECOVERY
                  }]
                  ,{END|E}
```

- SET, IF, SB, and END are positional parameters; all others are keyword parameters.
- Only one non-ignore PER trap may be enabled at any one time. If you attempt to set an enabled non-ignore PER trap while one is already enabled, the trap is defined, forced to the disabled state, and message IEE740I is issued. If you attempt to enable a non-ignore PER trap while one is already enabled, the request is denied and message IEE741I is issued.
- If you do not specify RECOVERY in conjunction with another parameter, the use of the indicated parentheses is optional.

- Setting a PER Trap for Storage Alteration



- SET, SA, and END are positional parameters; all others are keyword parameters.
- Only one non-ignore PER trap may be enabled at any one time. If you attempt to set an enabled non-ignore PER trap while one is already enabled, the trap is defined, forced to the disabled state, and message IEE740I is issued. If you attempt to enable a non-ignore PER trap while one is already enabled, the request is denied and message IEE741I is issued.
- If you do not specify RECOVERY in conjunction with another parameter, the use of the indicated parentheses is optional.
- In the DATA parameter, the elements of data-compare are: [asid.] target [(b)], operator, preval.
- In the DATA parameter, a maximum of 16 levels of parentheses are allowed; that is, no more than 16 unmatched left parentheses may appear in a DATA parameter specification.

DEL
delete an existing SLIP trap, or traps. (If TRDUMP is active, an SVC dump is scheduled.)

MOD
change the status of an existing SLIP trap, or traps

ENABLE or EN
a disabled trap, or traps, is enabled

DISABLE or D
an enabled trap, or traps, is disabled. (If TRDUMP is active, an SVC dump is scheduled.)

ALL
the status of all traps is changed or all traps are deleted.

ID
the status of a trap is changed or a trap is deleted

xxxx
trap identifier

value: 1-4 alphameric characters

Example 1

Operation: Delete a previously defined trap.

```
sl del,id=50CB
```

Example 2

Operation: Enable a previously defined trap.

```
sl mod,en,id=61DB
```

- Enabling or Disabling Previously Defined Traps

```
{SLIP} MOD, {{ENABLE  }  }, {ALL      }
{SL  }      {{EN      }  }  {ID=xxxx }
            {{DISABLE }  }
            {{D       }  }
```

- All parameters are positional parameters.
- If more than one user of SLIP has defined traps in the system, their actions to enable or disable traps must be coordinated to prevent undesirable results. For example, you can unknowingly enable traps previously disabled by another user by issuing the following command:

  SLIP MOD, EN, ALL

  To find out if other traps have been set and what their status is, use the DISPLAY SLIP subcommand. Another way to avoid undesirable results is to enable or disable traps explicitly by specifying ID=.

  When a trap is enabled or disabled by a TSO userid other than the one who set the trap, the originator of the trap is informed of the trap's changing status and the userid responsible via message IEE727I.

- Deleting Previously Defined Traps

```
{SLIP} DEL, {ALL      }
{SL  }      {ID=xxxx }
```

- All parameters are positional parameters.
- If more than one user of SLIP has defined traps in the system, their actions to delete traps must be coordinated to prevent undesirable results. For example, you can unknowingly delete traps previously defined by another user by issuing the following command:

  SLIP DEL, ALL

  To find out if other traps have been set and what their status is, use the DISPLAY SLIP subcommand. Another way to avoid undesirable results is to delete traps explicitly by specifying ID=.

  When a trap is deleted by a TSO userid other than the one who set the trap, the originator of the trap is informed of the trap's changing status and the userid responsible via message IEE727I.

STOPMN Subcommand of OPERATOR

Use the STOPMN subcommand to terminate the monitoring operations of the MONITOR subcommand. This subcommand will halt the display of information at your terminal.

The syntax of the STOPMN subcommand of OPERATOR is:

```
{STOPMN}     {JOBNAMES}
{PM    }     {SESS    }
           {STATUS  }
```

JOBNAMES
the system stops displaying the names of jobs as they start and terminate

SESS
the system stops displaying TSO userids as terminal sessions are initiated and terminated

STATUS
the system stops displaying the names and volume serial numbers of data sets with dispositions of KEEP, CATLG, or UNCATLG when the data sets are freed

Example 1

Operation: Stop the display of the names of jobs as they start and terminate.

```
stopmn jobnames
```

Example 2

Operation: Stop the display of TSO userids as terminal sessions are initiated and terminated.

```
pm  sess
```

Index

GC28-1173-01

MVS/Extended Architecture
System Programming Library: TSO

GC28-1173-01

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

GC28-1173-01

Reader's Comment Form

Cut or Fold Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department D58, Building 920-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape | Please Do Not Staple | Fold and tap

IBM®

MVS/Extended Architecture SPL: TSO (File No. S370-39) Printed in U.S.A. GC28-1173-01

MVS/Extended Architecture
System Programming Library: TSO

GC28-1173-01

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: ***Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

GC28-1173-01

Reader's Comment Form

Cut or Fold Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department D58, Building 920-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape | Please Do Not Staple | Fold and tap

IBM®

GC28-1173-1

MVS/Extended Architecture SPL: TSO (File No. S370-39) Printed in U.S.A. GC28-1173-01