GC28-0629-1
File No. S370-39

Systems

OS/VS2 MVS System Programming Library: TSO

VS2 Release 3.7

See also
TSO Command Reference Manual
SC28-1134-

IBM

Second Edition (January, 1976)

This is a major revision of, and obsoletes GC28-0629-0. See the Summary of Amendments following the Contents. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition with Technical Newsletter GN28-2815 applies to release 3.7 of OS/VS2 and to all subsequent releases of OS/VS2 until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest **IBM System/370 Bibliography**, GC20-0001, for the editions that are applicable and current.

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 is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, Building 706-2, PO Box 390, Poughkeepsie, N.Y. 12602. Comments become the property of IBM.

IBM Technical Newsletter

This Newsletter No. GN28-2815
Date November 15, 1976

Base Publication No. GC28-0629-1
File No. S370-39

Prerequisite Newsletters None

OS/VS2 System Programming
Library: TSO

This Technical Newsletter provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be inserted and/or removed are:

Pages to be Removed	Attached Pages to be Inserted*
Cover - Edition Notice	Cover - Edition Notice
5 - 8	5 - 8
15 - 16	15 - 16
21 - 24	21 - 24
27 - 32	27 - 32.2
49 - 50	49 - 50
53 - 54	53 - 54
65 - 68	65 - 68
81 - 82	81 - 82

A change to the text or to an illustration is indicated by a vertical line to the left of the change.

*If you are inserting pages from different Supplements/Newsletters and *identical* page numbers are involved, always use the page with the latest date (shown at the top of the page). The page with the latest date contains the most complete information.

Summary of Amendements

This update reflects miscellaneous technical corrections to the base publication. The major changes are:

SUBMIT

SUBMIT support of comment statements and continuation statements is explained.

LOGON Installation Exit

The meaning of certain fields in the parameter list to the LOGON installation exit (IKJEFLD) is clarified.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications Development, Department D58, Building 706-2, PO Box 390, Poughkeepsie, New York 12602

Printed in U.S.A.

TSO/VTAM Newsletter

VS2.03.813

This Newsletter No. GN28-2654
Date May 28, 1976

Base Publication No. GC28-0629-1
File No. S370-39

Previous Newsletters None

OS/VS2 System Programming
Library: TSO

This TSO/VTAM Newsletter provides replacement pages for the subject publication. These replacement pages remain in effect unless specifically altered.

TSO/VTAM has a prerequisite: VTAM 2. Therefore, do *not* replace the indicated pages in the subject publication unless you previously installed VTAM 2.

Only replace Table of Contents; list of Figures, Illustrations, and Tables; and Index pages in the subject publication with pages from a Newsletter with a later date. (Newsletters with the same date contain equivalent information.)

Pages to be replaced/removed/inserted are:

3 - 6
9 - 12
21, 22 (includes 21.0 - 21.3)
41, 42
79 - 82

A change to the text or to an illustration is indicated by a vertical line to the left of the change.

Summary of Amendments

Sections have been added on initializing TSO/VTAM time sharing, writing the procedure that starts TSO/VTAM time sharing, and building translation tables for TSO/VTAM users. In addition, the system programmer is instructed to construct the TSOKEY00 member of SYS1.PARMLIB, and to write any editing, attention handling, and nonsupported device exit routines that are desired.

For a complete list of publications that support TSO/VTAM, see OS/VS2 MVS TSO/VTAM System Information, GC27-0046.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications Development, Department D58, Building 706-2, PO Box 390, Poughkeepsie, New York 12602

Printed in U.S.A.

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, which should be installation controlled.

References to multiple virtual storage (MVS) pertain to OS/VS2 Release 2 and subsequent releases.

Associated publications:

OS/VS2 System Programming Library: Initialization and Tuning Guide,GC28-0681

OS/VS2 System Programming Library: Job Management, GC28-0627

OS/VS2 System Programming Library: Supervisor, GC28-0628

OS/VS2 TCAM Programmer's Guide, GC30-2041

OS/VS2 Data Areas, SYB8-0606

OS/VS2 TSO Command Language Reference, GC28-0646

OS/VS2 System Programming Library: System Generation Reference GC26-3792

OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor, GC28-0648

OS/VS2 MVS System Programming Library: VTAM, GC28-0688

OS/VS2 MVS VTIOC and TCAS Logic, SY27-7269

IBM System/370 Principles of Operation, GA22-7000

Contents

Figures

Summary of Amendments for GC28-0629-1 as Updated by GN28-2815 OS/VS2 Release 3.7

This update reflects miscellaneous technical corrections to the base publication. The major changes are:

SUBMIT Command

SUBMIT support of comment statements and continuation statements is explained.

LOGON Installation Exit

The meaning of certain fields in the parameter list to the LOGON installation exit (IKJEFLD) is clarified.

Summary of Amendments for GC28-0629-1 OS/VS2 Release 3.7

Changes have been made throughout this publication to reflect a Service Update to OS/VS2 Release 3.7. In addition, pertinent technical and editorial changes have been made in the following areas:

Determining TSO User Size

Gives the search order used to obtain the region size that is to be allocated to a TSO user.

Adding Subcommands to the EDIT Command

Supplies the code that can be used to include IKJEBEST in your macro library.

Summary of Amendments for GC28-0629-0 OS/VS2 Release 3

TSO Services

Was taken in its entirety from Part 3 of **OS/VS2 System Programming Library: Job Management, Supervisor,** and TSO. Major changes and additions to this data are as follows:

TSO Allocation Suggestions

Offers suggestions for improving TSO allocations.

- Placement of user data sets in a LOGON Procedure.
- Carefully choosing the DYNAMNBR parameter value in the EXEC statement.

Space Allocation Using the EDIT Access Method

Explains the algorithms used in allocating EDIT's utility work data set space.

- Existing data sets.
- New data sets.

Writing Installation Exits for the RENUM Subcommand of EDIT

Expanded to cover the MOVE and COPY subcommands of EDIT.

Executing Authorized Programs Under TSO

Gives restrictions and exposures pertaining to the execution of authorized programs.

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, convert, 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 subcommand of EDIT.
- Write an exit for an installation-written syntax checker.
- Build translation tables for TSO/VTAM users.

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 **OS/VS TCAM Programmer's Guide**, GC30-2044.
- Writes the MCP cataloged procedure and includes it in SYS1.PROCLIB.
- Constructs the IKJPRM00 member of SYS1.PARMLIB to set terminal I/O coordinator (TIOC) parameters. See the discussion of IKJPRM00 in **OS/VS2 System Programming Library: Initialization and Tuning Guide**, GC28-0681.

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 **OS/VS2 System Programming Library: Initialization and Tuning Guide.**
- 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. Refer to **OS/VS2 MVS VTIOC and TCAS Logic**, SY27-7269.

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

- Writes the LOGON cataloged procedure(s) and includes them in SYS1.PROCLIB.
- Includes SYS1.CMDLIB in a LNKLSTxx member of SYS1.PARMLIB or in a LOGON cataloged procedure. For information on LNKLSTxx, see **OS/VS2 System Programming Library: Initialization and Tuning Guide.**
- Creates or converts the UADS and broadcast data sets.

Note: No BRDR procedure is needed for MVS, since the background reader for submitted jobs has been replaced with a direct interface from the SUBMIT command to a job entry subsystem internal reader (INTRDR).

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 **OS/VS2 SPL: Job Management.** 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 **OS/VS2 TCAM Programmer's Guide.**

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 multiple virtual storage 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 procedure.
2. Specify xx in the PROC==xx parameter for the &TSU job class in the JES2 initialization parameters. (See "JES2 Initialization" in **OS/VS2 SPL: Initialization and Tuning Guide.**)

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 **SPL: Supervisor** for details about IEALIMIT control.)
- The region size value specified for locations 1, 2, and 3 (in 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 F. 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 Used by the EDIT Access Method

The following paragraphs explain the algorithms used in allocating EDIT's utility work data set space when editing an existing data set or a new data set. These algorithms are used unless the EDIT utility work data sets are otherwise allocated by the installation. If the installation pre-allocates EDIT's work files, the EDIT Access Method Space management is not in effect.

When editing an Existing Data Set — Two times the ((total number of characters contained within the data set plus additional requested records (*) divided by the average block size of 4096) plus 2) gives you the initial space size in 4K blocks. Dividing the initial space size by two gives you the secondary space size.

Example

Known:

1. The data set contains 2000 records.
2. Each record is 30 characters in length.
3. The average block size if 4096.

```
2x ((160,000 + 0 ÷ 4096) + 2)=initial space (number of 4K blocks)
Initial space ÷ 2=secondary space (number of 4K blocks)
```

When Editing a New Data Set — A new data set has four blocks with the average block size of 4096 for the defaulted initial space and one half that amount for its secondary space. If it contains 80-character records it has a total capacity of approximately 1300 records.

sysedit — sysedit is a sample ddname for the &EDIT data set. When a DD statement is included for &EDIT, dynamic allocation recognizes that the &EDIT data set is permanently allocated. Thus, the &EDIT data set allows an installation to control the allocation of &EDIT utility data sets, and to control the direct access space used for these data sets.

sysedit2 — sysedit2 is a sample ddname for the EDIT2 data set. When a DD statement is included for &EDIT2, dynamic allocation recognizes that the EDIT2 data set is permanently allocated. Thus, the &EDIT2 data set allows an installation to control the allocation of EDIT utility data sets and to control the direct access space used for these data sets.

Sample DD statements (when used in LOGON procedure)

```
//SYSEDIT   DD  DSN=&EDIT,UNIT=SYSDA,SPACE=(1688,(50,20))
//SYSEDIT2  DD  DSN=&EDIT2,SPACE=(4096,(20,10)),UNIT=SYSDA
```

Creating, Converting, and Maintaining UADS and Broadcast Data Sets

This section tells the system programmer how to:

- create the UADS and broadcast data sets
- convert 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 UADS and broadcast data sets are available to the system. Both data sets have a specific format for MVS. The data sets are created in MVS format. The LOGON processor can handle either old or new formats of the UADS, but the data set must be in MVS format in order to modify or add an entry. The broadcast data set must be in MVS format. After the UADS and broadcast data sets are established, authorized individuals can update them by adding, changing, or deleting entries.

Neither the UADS nor broadcast data sets should be placed on a shared DASD device that is accessed by more than one CPU. Sharing of these data sets is not supported.

ACCOUNT Command 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 4: Reference — Macro Instructions and Commands" in this publication.

To permit creating and maintaining the UADS and broadcast data sets from a batch job, VS2 Release 2 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

The SYS1.UADS data set does not have to be reformatted before using TSO. However, you should be aware of the following points if you continue to use the old format:

- The ACCOUNT command will cause entries to be added to SYS1.UADS in the new format.
- The accounting fields are different in the new format. When the system resource manager updates accounting fields in SYS1.UADS, it assumes that SYS1.UADS exists in the new format.
- A user cannot use the MVS parameters of the ACCOUNT subcommands unless SYS1.UADS exists in the new format.

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.

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, which can be in MVS format or the format from previous releases. The SYSUADS DD statement specifies the output UADS, which will be in MVS format.

Content and Structure of UADS

The user attribute data set (UADS) is basically a list of terminal users who are authorized to 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.

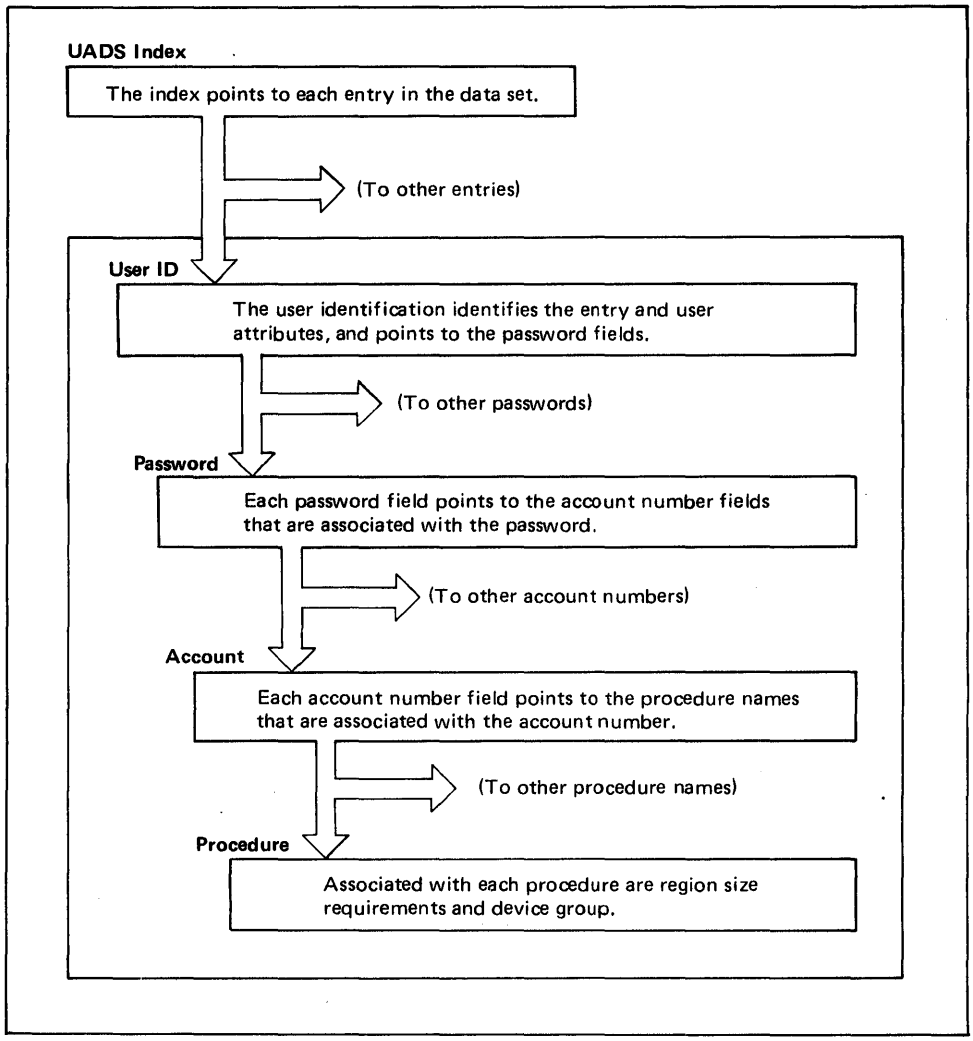


Figure 3. Organization of the UADS

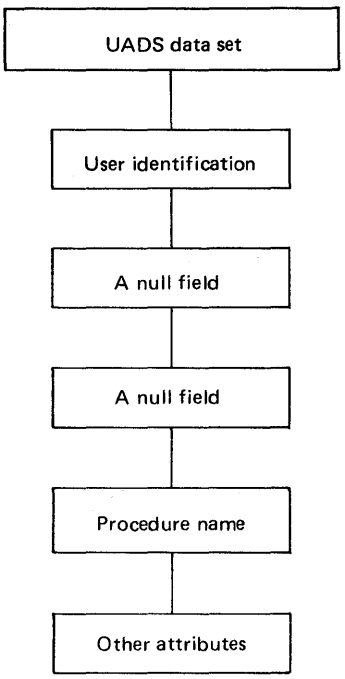


Figure 4. The Simplest Structure for a Typical UADS Entry

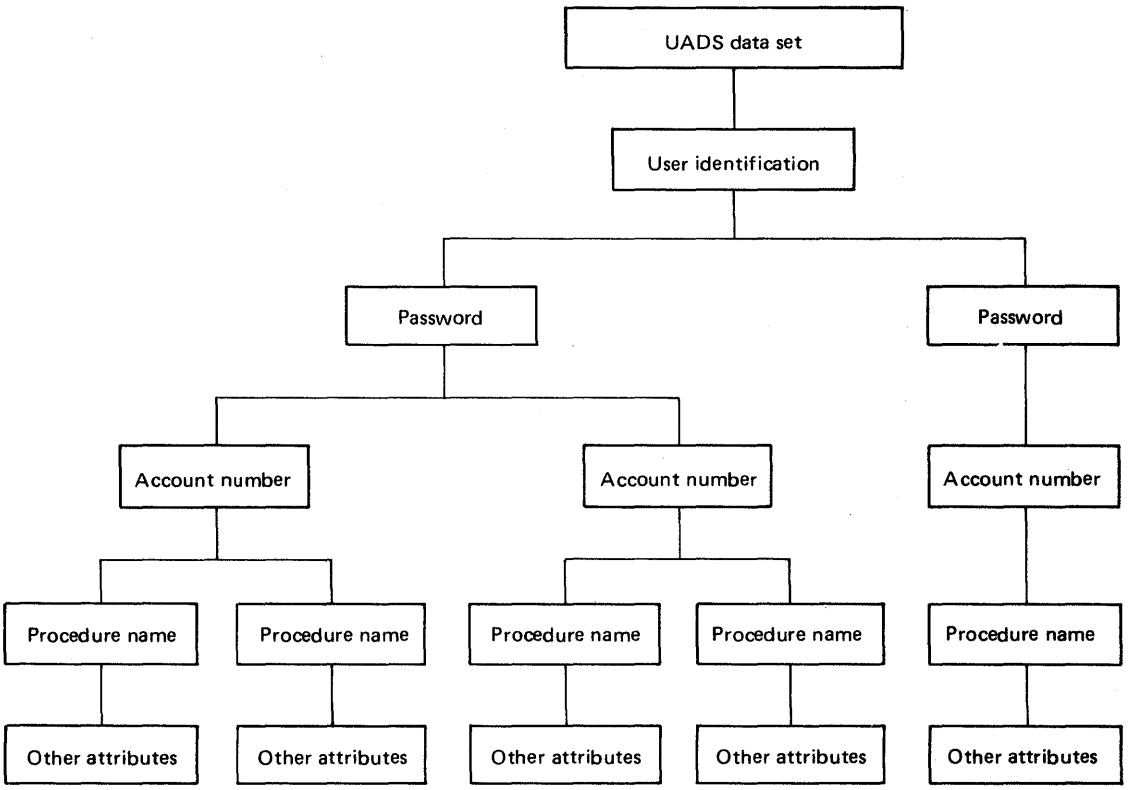


Figure 5. A Complex Structure for a Typical UADS Entry

Creating the UADS and Broadcast Data Sets From a Terminal

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

While running the newly-generated system, execute the ACCOUNT facility as a batch job. Use the SYNC subcommand of ACCOUNT to format a skeleton for the broadcast data set. Then, as each userid is added to UADS via the ADD subcommand of ACCOUNT, a corresponding entry is made in 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 sample listing showing the creation of UADS and broadcast with a batch job. An explanation of this JCL can be found under "Executing the TMP as a Batch Job".

```
//jobname   JOB     job card parameters
//          DD      EXEC    PGM=IKJEFT01
//SYSPRINT  DD      SYSOUT=A
//SYSUADS   DD      DSN=uads created during sysgen
//SYSLBC    DD      DSN=SYS1.BRODCAST,DISP=SHR
//SYSIN     DD      *
ACCOUNT
SYNC
ADD new userid
 .
 .
 .
DELETE (IBMUSER)
END
/*
```

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

Converting the UADS and Broadcast Data Sets

While running the newly-generated system, execute the ACCOUNT facility with a batch job. Use the UADSREFM program to create an MVS UADS containing the userids from the old format UADS. Use the SYNC subcommand of ACCOUNT to format an MVS 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 create a new UADS. Because a new UADS has been created, IBMUSER should be deleted. Figure 7 is a sample listing showing the conversion of UADS and broadcast. An explanation of this JCL can be found under "Executing the TMP as a Batch Job".

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

Figure 7. A Sample Listing, Showing the Conversion 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.

Executing the TMP as a Batch Job

The terminal monitor program (TMP), if invoked in the background, creates a TSO environment so that the TSO service routines ACCOUNT and its subcommands, PROFILE, and the Access Method Services utilities can function. Input for a GETLINE is obtained from the data set defined by the SYSIN DD statement. The TMP issues the STACK macro instruction to put the SYSIN data set on the input stack. Messages issued via PUTLINE are written to the data set defined by the SYSPRINT DD statement.

Multilevel messages are automatically written out as if a question mark(?) had been entered. The commands that are read from SYSIN are logged on SYSPRINT. Because of this, the commands and subcommands can be entered via SYSIN. Each command must begin on a separate card. Continuation is indicated according to "Continuation Lines" elsewhere in this book.

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

```
profile noprompt noprefix noline nochar
```

Since "noprefix" is defaulted, an unqualified data set name will not have a userid prefixed to it. If you would like to have a userid prefixed to the data-set-names specified on the Access Method Services utilities include the command:

```
profile prefix(userid)
```

at the beginning of the SYSIN stream.

The JCL used to run the TMP in a batch job is the:

1. EXEC statement with PGM=IKJEFT01 which should have a DYNAMNBR parameter if ACCOUNT will dynamically allocate SYS1.BRODCAST or SYS1.HELP data sets. (This is only for HELP subcommands). Optional for this statement is the PARM operand, which is interpreted as the first line of input from SYSIN.
2. SYSPRINT DD statement for commands and subcommands executed, plus messages.
3. SYSIN DD statement for data sets containing commands and subcommands.

Note: The SYSIN and SYSPRINT 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 SYSIN 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 above. 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 commmand. For security, the UADS should be password protected or a JCL exit should be written to limiit 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 Guide.**

To terminate all time-sharing users' connections with the system, the system operator must issue the MODIFY TCAM,TS=STOP.

For more information on START and MODIFY commands, see **Operator's Library: OS/VS TCAM.**

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 **OS/VS2 System Programming Library: Initialization and Tuning Guide**, GC28-0681.

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 **Operator's Library: OS/VS2 MVS System Commands**, GC38-0229.

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**, GA22-7000.) 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 7.1 shows translation tables that translate the characters "[" (X'AD'), "]" (X'BD'), and "!" (X'5A') to "<" (X'4C'), ">" (X'6E'), and "|" (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    0D
         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    0D
         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 7.1 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 linkage-edited in SYS1.LINKLIB as an independent module named IKJEFF10, with reusable, reenterable, and refreshable linkage-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 and cannot pre-load a revised version using the TEST command.) 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 *OS/VS2 Data Areas,* (microfiche) SYB8-0606, 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
Address of the current statement. If 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
Address of a message to be displayed on the user's terminal, supplied in conjunction with return code 8 or 12. If nonzero 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
Address of the response from the terminal user if the exit routine's previous return code was 12. When this field is nonzero 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
Address of userid. The userid is eight characters left-justified, padded with blanks.

Word 5
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 or JES3 (ASP) 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	Reserved

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 nonzero) 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 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

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

Address of reconstructed LOGON job accounting information for the user. The SUBMIT command processor inserts this information into generated JOB cards.

Word 8

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. (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 entered 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 linkage-edited in SYS1.LINKLIB as an independent module named IKJEFF53, with reusable, reenterable, and refreshable linkage-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 and cannot pre-load a revised version using the TEST command 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 next jobname, and continue processing.

4 Display message IKJ56208A (message text is supplied by the exit routine), get 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 *OS/VS2 Data Areas,* (microfiche) SYB8-0606, 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. Zero if class was not specified on OUTPUT command.

Word 2 — pointer to the print-data-set-name PDE. Zero if not entered.

Word 3 — pointer to the new class PDE. Zero if not entered.

Word 4 — pointer to the destination PDE. Zero if 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: 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. 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 8 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).

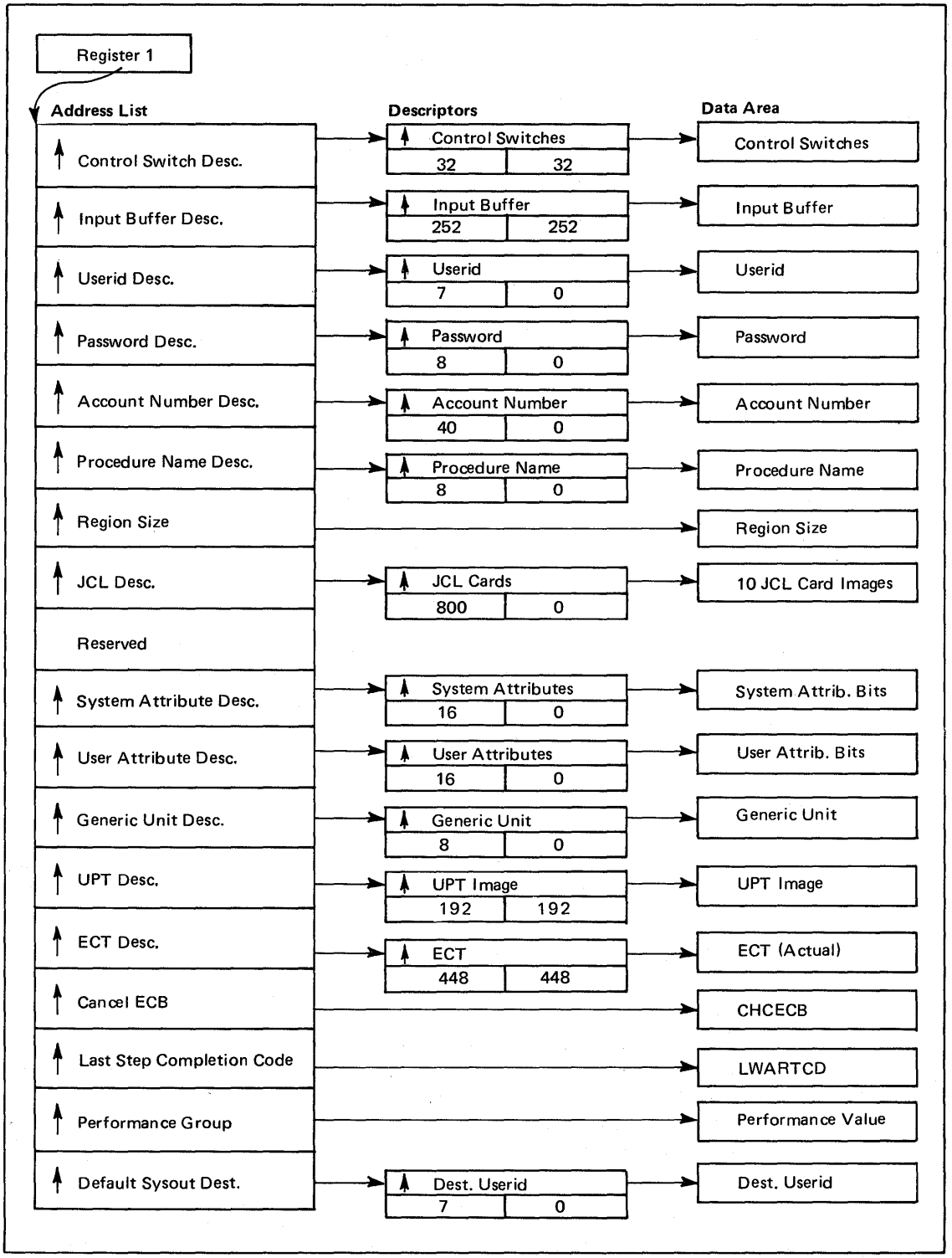


Figure 8. LOGON Pre-prompt Parameters

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 LOGON Processor. The control switch bits set by 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	Reserved
	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
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
Set to one by the LOGON Processor to tell IKJEFLD that LOGON Processor was unable to obtain a resource other than the userid.

Disconnect
Set to one by IKJEFLD to tell LOGON Processor to terminate the session. LOGON Processor sends no further message to the terminal.

Don't Prompt
Set to one by IKJEFLD to tell 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
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
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).

Reserved
This bit is reserved.

System Attributes
Set to one by IKJEFLD to tell the LOGON Processor that IKJEFLD has supplied the system attribute bits.

User attributes
Set to one by IKJEFLD to tell the LOGON Processor that IKJEFLD has supplied the user attribute bits.

Generic group
Set to one by IKJEFLD to tell the LOGON Processor that IKJEFLD has supplied a generic group name.

UPT
Set to one by IKJEFLD to tell the LOGON Processor that IKJEFLD has supplied the UPT.

Don't ENQ
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.

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. 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 region size 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.

Reserved: This parameter is reserved.

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 *OS/VS2 Data Areas,* (microfiche) SYB8-0606, 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 *OS/VS2 Data Areas,* (microfiche) SYB8-0606, 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 **OS/VS2 Data Areas**, 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 noUADS bits are set to one. See **OS/VS2 Data Areas**, 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 full word containing a number that associates terminal user interactions with one of several performance groups. 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 **OS/VS2 System Programming Library: Initialization and Tuning Guide.**

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

Sample LOGON Pre-prompt Exit Routine

A sample LOGON pre-prompt exit is shown in Figure 9. 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.

```
         CSECT   IKJEFLD
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 9. 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 successful completion or attention interrupt

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

8 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:
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 and Syntax Checking

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 **OS/VS2 TSO Command Language Reference,** GC28-0646. For more information on how to specify the EDIT macro instruction during system generation, see **OS/VS2 System Programming Library: 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, if desired. Each syntax checker and its associated data set type must be defined during system generation by using the EDIT macro instruction. (See OS/VS2 System Programming Library: 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 10.

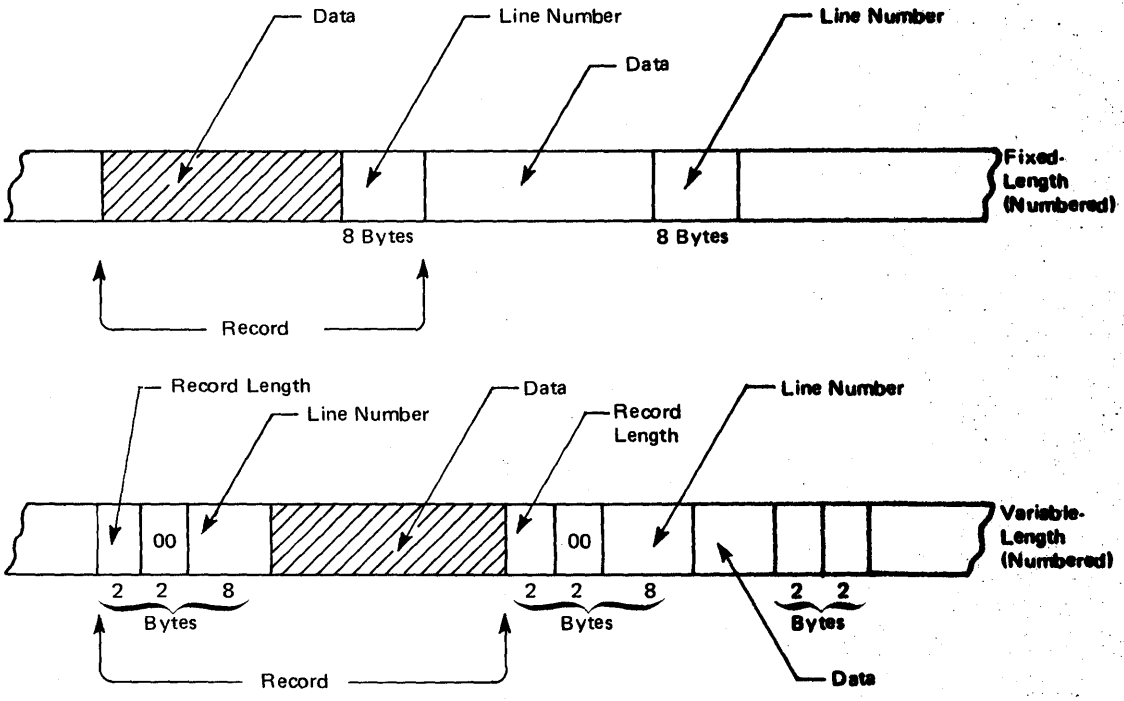


Figure 10. Format of Records Passed to Syntax Checkers

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 11. An installation-written syntax checker can also use this interface, which consists of the syntax checker parameter list.

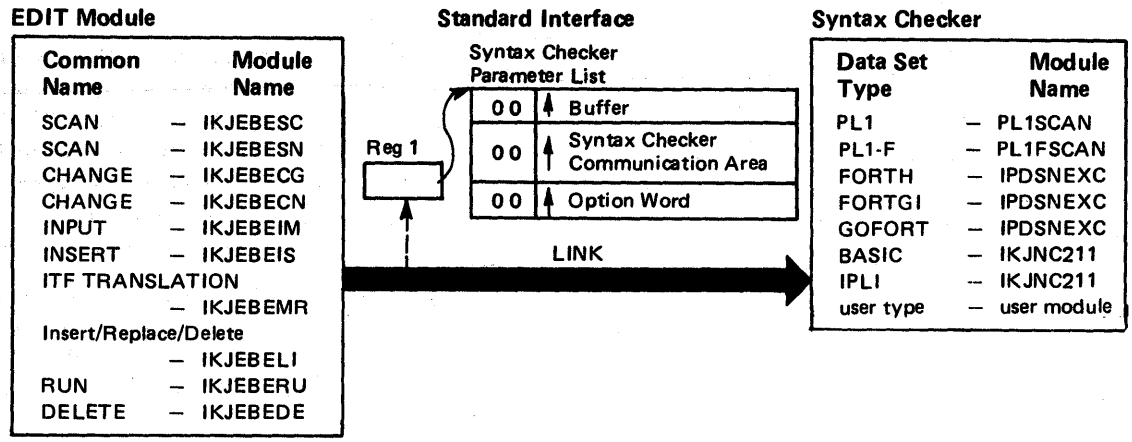


Figure 11. Interface Between EDIT Program and Syntax Checkers

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 12. 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 13. 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 14. 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 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 **OS/VS2 TSO Guide to Writing a Terminal Monitor Program or Command Processor,** GC28-0648, 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 linkage-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 the ATTACH macro support in the system includes recognition of the RSAPF keyword and 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 either the keyword RSAPF is not recognized or 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 OS/VS utility programs IEHDASDR, IEBCOPY, IEHMOVE, IEHATLAS, IEHINITT, and IEHPROGM. TSO users can execute these utilities 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. 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. The 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.

You can communicate service requests to the control program using a set of macro instructions and commands provided by IBM. Whereas most of the macro instructions and commands have no restrictions on use by applications programmers and console operators, some of them should be restricted in use to systems programmers and installation-approved personnel.

Note: For more information on restricted macro instructions, refer to **OS/VS2 SPL: Supervisor,**.

This section describes the TSO commands that should be installation controlled. The ACCOUNT and OPERATOR commands should be totally restricted in use. These commands are completely described in this book.

Coding the Commands

The table appearing near the beginning of each command or subcommand indicates how each is to be coded. The table does not attempt to explain the meanings of the parameters; the parameters are explained in the text following the table. See the Figure 15 for a sample subcommand.

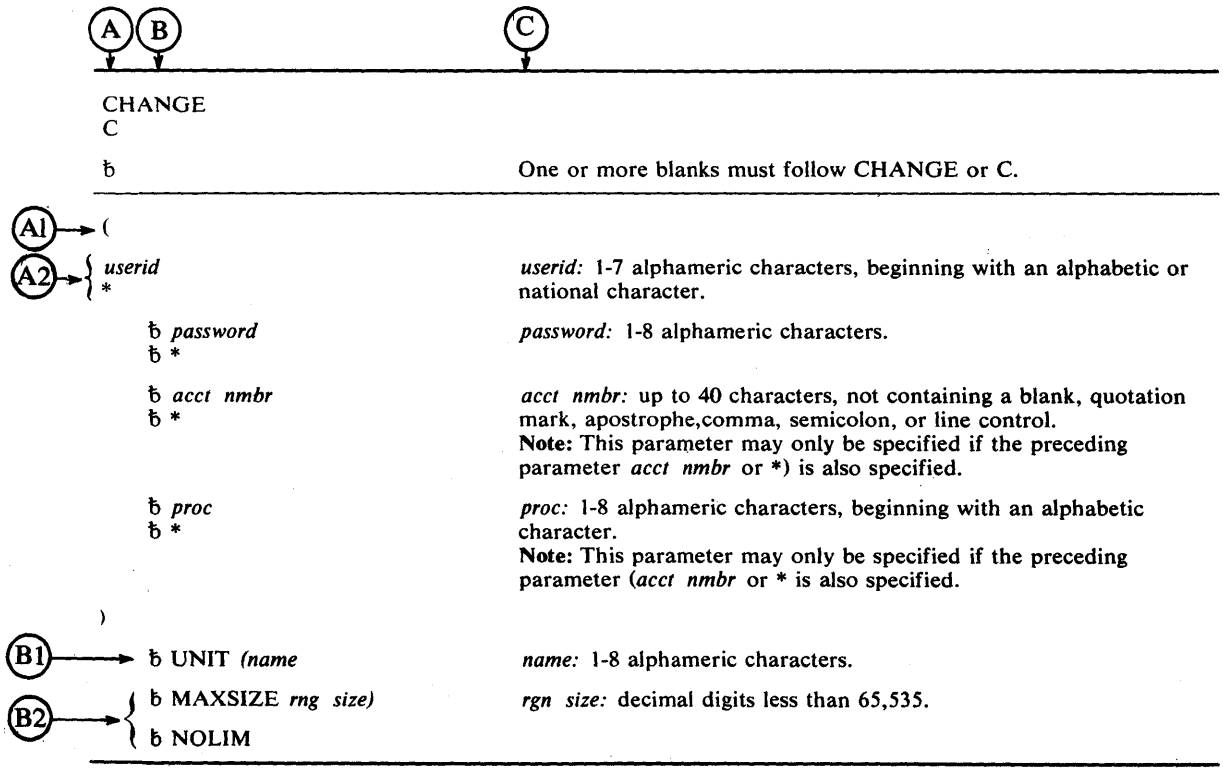


Figure 15. Sample Subcommand

- The first column, (A), contains those parameters that are required for that TSO command or subcommand. If a single line appears in that column, (A1), the parameter on that line is required and must be coded. If two or more lines appear together, (A2), the parameter appearing on one and only one of the lines must be coded.

- The second column, (B), contains those parameters that are optional for that TSO command or subcommand. If a single line appears in that column, (B1), the parameter on that line is optional. If two or more lines appear together, (B2), although the entire parameter is optional, if you elect to make an entry, one and only one of the lines may be coded.
- The third column, (C), provides additional information for coding the TSO command or subcommand. When substitution of a variable is required, the following classifications should be understood:

symbol: any symbol valid in the assembler language. That is, an alphabetic character followed by 0-7 alphameric characters, with no special characters and no blanks.

decimal digit: any decimal digit up to the value indicated in the parameter description. If both symbol and decimal digit are indicated, an absolute expression is also allowed.

default: a value that is used in default of a specified value, and that the system assumes if the parameter is not coded.

* (asterisk):represents a null field when creating a new data set.

Use the parameters to specify the services and options to be performed, and write them according to the following general rules:

- If the selected parameter is written in all capital letters (for example, ACCT or OPER), code the parameter exactly as shown.
- If the selected parameter is written in italics (for example, *value* or *pdsname*), substitute the indicated value, or pdsname.
- Read the table from top to bottom, and code the parameters in the order shown. Code commas and parentheses exactly as shown.
- If a parameter is selected, read column 3 before proceeding to the next parameter. Column 3 will often contain notes pertaining to restrictions on coding the parameter.

Continuation Lines

You may continue a command or subcommand on one or more lines by following this rule:

Continuation of a command may be done by using a hyphen (minus sign) or a plus sign. Placing either one of these special characters as the last character on the line will allow you to continue to the next line. One caution should be observed; the plus sign will cause the deletion of any leading delimiters (blanks, commas, tabs, and comments) on the continued line.

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. (You can use this command only if your installation has given you the authority to do so.) 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. The Broadcast Data Set can contain notices and mail for all USERIDs which are defined to it.

The ACCOUNT command is written as follows:

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; add new data to existing entries.
CHANGE	Change data in specified fields of UADS entries.
DELETE	Delete entries or parts of entries from the UADS.
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 BRODCAST data set and synchronize it with the UADS.

In MVS, TSO users can be authorized (by means of the MOUNT parameter of the ACCOUNT subcommands ADD and CHANGE to allow volumes to be mounted. 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). There is no message 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 has responded to the request. Users authorized to allow volume mounting should be aware of this situation. It is advisable to send the operator a message prior to issuing the command requesting a particular volume.

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 User Attribute Data Set (UADS) 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.

The ADD subcommand of ACCOUNT is written as follows:

ADD A	
ƀ	One or more blanks must follow ADD or A.
(	
userid *	*userid:* 1-7 alphameric characters, beginning with an alphabetic or national character.
ƀ *password* ƀ *	*password:* 1-8 alphameric characters.
ƀ *acct nmbr* ƀ *	*acct nmbr:* Up to 40 characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character. **Note:** This parameter may only be specified if the preceding parameter (*password* or *) is also specified.
ƀ *proc* ƀ *	*proc:* 1-8 alphameric characters, beginning with an alphabetic character. **Note:** This parameter may only be specified if the preceding parameter (*acct nmbr* or *) is also specified.
)	
ƀ DATA *(password* ƀ *acct nmbr* ƀ *proc)* ƀ DATA *(acct nmbr* ƀ *proc)* ƀ DATA *(proc)*	*password:* List of passwords (1-8 alphameric characters), separated by commas or blanks. *acct nmbr:* List of account numbers (up to 40 characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character), separated by commas or blanks. *proc:* List of procedure names (1-8 alphameric characters, beginning with an alphabetic character), separated by commas or blanks.
ƀ SIZE *(rgn size)*	*rgn size:* decimal digits less than 65,535.
ƀ UNIT *(name)*	*name:* 1-8 alphameric characters.
ƀ MAXSIZE *(rgn size)* ƀ NOLIM	*rgn size:* decimal digits less than 65,535. **Default:** NOLIM
ƀ ACCT ƀ NOACCT	**Default:** NOACCT
ƀ OPER ƀ NOOPER	**Default:** NOOPER
ƀ JCL ƀ NOJCL	**Default:** NOJCL
ƀ USERDATA *(data)*	*data:* 4 hexadecimal digits.
ƀ PERFORM *(perf groups)* ƀ NOPERFORM	*perf groups:* list of decimal digits 1-255, separated by commas or blanks. **Default:** NOPERFORM
ƀ MOUNT ƀ NOMOUNT	**Default:** NOMOUNT
ƀ DEST *(userid)* ƀ NODEST	*userid:* 1-7 alphameric characters, beginning with an alphabetic or national character. **Default:** NODEST

The parameters are explained below:

The first positional parameter specifies the point within the UADS structure where data is to be added. If all four items are specified a new USERID will be created.

If less than four items are specified, information will be added to an existing USERID. The sum of the items in the positional list and the DATA list must equal four.

An asterisk (*) indicates a null field when you are creating a new entry.

(

specifies the beginning of a series of positional parameters.

userid
*

specifies a user identification that identifies the UADS entry and the entry in the SYS1.BRODCAST data set for the user *(userid)* or specifies that information for all user identifications in the UADS is to be added (*). The entry that this field identifies may be an existing entry to which new data is to be added or a new entry that is to be added to the UADS.

b *password*
b *

specifies a word that a user must enter before he can use the system *(password)* or specifies all passwords for the indicated user identification (*). The password helps indicate the structure in the UADS to which data is being added or, when you are adding an entire new entry, the password is part of the data being added.

b acct nmbr
b *

specifies an account number used for administrative purposes *(acct nmbr)* or specifies all account numbers for the indicated password for the indicated user identification(*). The account number helps indicate the structure in the UADS to which data is being added or, when you are adding an entire new entry, the account number is part of the data being added.

b *proc*
b *

specifies the name of a procedure that is invoked when the user enters the LOGON command *(proc)* or specifies all procedure names for the indicated account number for the indicated password for the indicated user identification (*). You should not specify this field for the first positional parameter unless you are adding an entire new entry to the UADS.

)

specifies the end of a series of positional parameters.

b DATA *(password* b *acct nmbr* b *proc)*
b DATA *(acct nmbr* b *proc)*
b DATA *(proc)*

specifies the data that is to be added to the existing entry:

password specifies a password or a list of passwords to be added to the existing entry at the location indicated by the positional parameter. When you specify a list of passwords, the list must be enclosed within a spearate set of parentheses embedded within the set of parentheses required for the DATA keyword.

acct nmbr specifies an account number of a list of account numbers to be added to the existing entry. When you specify a list of account numbers, the list must be enclosed within a separate set of parentheses embedded within the set of parentheses required for the DATA keyword. No more than 255 identical account numbers may exist under one password in a user entry.

proc specifies a procedure name or a list of procedure names to be added to the existing entry. When you specify a list of procedure names, in addition to one or more other fields, the list must be enclosed within separate set of parentheses embedded within the set of parentheses required for the DATA keyword. You should specify the region size requirements and device units for each procedure by using the SIZE and UNIT keyword. No more than 255 identical procedure names may exist under one account number in a user entry.

- SIZE *(rgn size)*

  specifies the minimum region size, in 1024-byte units, that the user will have assigned to him if he does not specify a size himself. You can specify a SIZE atrribute for each unique combination of password, account number, and procedure name in the entry. If you specify SIZE(0), the default value is the minimum size available.

- UNIT *(name)*

  specifies the name of a group of devices that the procedure will use. (Data sets allocated via the catalog are an exception. See the ALLOCATE command in the **OS/VS2 TSO Command Language Reference**, GC28-0646). You can specify a UNIT attribute for each unique combination of password, account number, and procedure in the entry.

- MAXSIZE *(rgn size)*
- NOLIM

  specifies the maximum region size, in 1024-byte units, that the user may request at LOGON (MAXSIZE) or that the user is not restricted to a maximum region size (NOLIM). If you specify MAXSIZE(0), the default of NOLIM is assumed. Use this parameter only when you add a complete entry to the UADS.

- ACCT
- NOACCT

  specifies that the user can (ACCT) or cannot (NOACCT) use the ACCOUNT command, thereby controlling access to the time sharing system. Use this parameter only when you add a complete entry to the UADS.

- OPER
- NOOPER

  specifies that the user can (OPER) or cannot (NOOPER) use the OPERATOR command. Use this parameter only when you add a complete entry to the UADS.

- JCL
- NOJCL

  specifies that the user can (JCL) or cannot (NOJCL) use the SUBMIT, STATUS, CANCEL, and OUTPUT commands. Use this parameter only when you add a complete entry to the UADS.

- USERDATA *(data)*

  specifies what the installation-defined data is to be added to in the user entry in the UADS. The two-byte field in the UADS is the EBCDIC representation of the four hexadecimal characters specified as *data*. The meaning of the field is defined by the user. Use this parameter only when you add a complete entry to the UADS.

- PERFORM *(perf groups)*
- NOPERFORM

  specifies that the TSO user is (PERFORM) or is not (NOPERFORM) authorized to request performance groups at logon. A default performance group will result from the TSO user's logon procedure. Use this parameter only when you add a complete entry to the UADS.

- MOUNT
- NOMOUNT

  specifies that dynamic allocation requests for this TSO userid are (MOUNT) or are not (NOMOUNT) authorized to cause volume mounting as necessary. Use this parameter only when you add a complete entry to the UADS.

- DEST *(userid)*
- NODEST

  specifies whether there is (DEST) or is not (NODEST) a default destination to which SYSOUT data sets dynamically allocated by this user will be routed. This default can be overridden by ALLOCATE, FREE, and other commands. Use this parameter only when you add a complete entry to the UADS.

Example 1

Operation: Add a new entry to the UADS.

```
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, 288104 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(mylog) size (73)
```

Example 5

Operation: Add a new account number and new 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.

The CHANGE subcommand of ACCOUNT is written as follows:

CHANGE C	
ƀ	One or more blanks must follow CHANGE or C.
(	
userid *	*userid:* 1-7 alphameric characters, beginning with an alphabetic or national character.
ƀ *password* ƀ *	*password:* 1-8 alphameric characters.
ƀ *acct nmbr* ƀ *	*acct nmbr:* up to 40 characters, not containing a blank, quotation mark, apostrophe, comma, semicolon, or line control. **Note:** This parameter may only be specified if the preceding parameter (*password* or *) is also specified.
ƀ *proc* ƀ *	*proc:* 1-8 alphameric characters, beginning with an alphabetic character. **Note:** This parameter may only be specified if the preceding parameter (*acct nmbr* or *) is also specified.
)	
ƀ DATA *(userid)* ƀ DATA *(password)* ƀ DATA *(acct nmbr)* ƀ DATA *(proc)*	*userid:* 1-7 alphameric characters, beginning with an alphabetic or national character. *password:* 1-8 alphameric characters. *acct nmbr:* up to 40 characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character. *proc:* 1-8 alphameric characters, beginning with an alphabetic character.
ƀ SIZE *(rgn size)*	*rgn size:* decimal digits less than 65,535
ƀ UNIT *(name)*	*name:* 1-8 alphameric characters.
ƀ MAXSIZE *(rgn size)* ƀ NOLIM	*rgn size:* decimal digits less than 65,535.
ƀ ACCT ƀ NOACCT	
ƀ OPER ƀ NOOPER	
ƀ JCL ƀ NOJCL	
ƀ USERDATA *(data)*	*data:* 4 hexadecimal digits.
ƀ PERFORM *(perf groups)* ƀ NOPERFORM	*perf groups:* lists of decimal digits 1-255, separated by commas or blanks.
ƀ MOUNT ƀ NOMOUNT	
ƀ DEST *(userid)* ƀ NODEST	*userid:* 1-7 alphameric characters, beginning with an alphabetic or national character.

The parameters are explained below:

(
specifies the beginning of a series of positional parameters.

userid
*
specifies a user identification that identifies the UADS entry to be changed *(userid)* or specifies that all user identifications in the UADS are to be changed (*).

- *password*
- *
specifies a word that a user must enter before he can use the system *(password)* or specifies all passwords for the indicated user identification (*). The password helps locate the data begin changed and, when you are changing a password, identifies the password being changed.

- *acct nmbr*
- *
specifies an account number used for administrative purposes *(acct nmbr)* or specifies all account numbers for the indicated password for the indicated user identification (*). The account number helps locate the data being changed and, when you are changing an account number, identifies the account number being changed.

- *proc*
- *
specifies the name of a procedure that is invoked when the user enters the LOGON command *(proc)* or specifies all procedure names for the indicated account number for the indicated password for the indicated user identification(*). The procedure name, when specified, is the data being changed.

)
specifies the end of a series of positional parameters.

- DATA *(userid)*
- DATA *(password)*
- DATA *(acct nmbr)*
- DATA *(proc)*

specifies the data that is to be changed from the existing entry:

userid specifies a user identification to replace the existing user identity.

password specifies a password to replace the existing password.

acct nmbr specifies an account number to replace the existing procedure name.

proc specifies a procedure name to replace the existing procedure name.

- SIZE *(rgn size)*

specifies the minimum region size, in 1024-byte units that the user will have assigned to him if he does not specify a size himself. You can specify a SIZE attribute for each unique combination of password, account number, and procedure name in the entry. If you specify SIZE)E(0), the default value is the minimum size available.

- UNIT *(name)*

specifies the name of a group of devices that the procedure will use. (Data sets allocated via the catalog are an exception. See the ALLOCATE command in the **OS/VS2 TSO Command Language Reference**, GC28-0646.)

- MAXSIZE *(rgn size)*
- NOLIM

  specifies the maximum region size, in 1024-byte units, that the user may request at LOGON (MAXSIZE) or that the user is not restricted to a maximum region size (NOLIM). If you specify MAXSIZE(0), the default of NOLIM is assumed.

- ACCT
- NOACCT

  specifies that the user can (ACCT) or cannot (NOACCT) use the ACCOUNT command, thereby controlling access to the UADS.

- OPER
- NOOPER

  specifies that the user can (OPER) or cannot (NOOPER) use the OPERATOR command.

- JCL
- NOJCL

  specifies that the user can (JCL) or cannot (NOJCL) use the SUBMIT, STATUS, CANCEL, and OUTPUT commands.

- USERDATA *(data)*

  specifies what the installation-defined data is to be changed to in the user entry in the UADS. The two-byte field in the UADS is the EBCDIC representation of the four hexadecimal characters specified as *data*. The meaning of the field is defined by the user.

- PERFORM *(perf groups)*
- NOPERFORM

  specifies that the TSO user is (PERFORM) or is not (NOPERFORM) authorized to request performance groups at logon. A default performance group will result from the TSO user's logon procedure.

- MOUNT
- NOMOUNT

  specifies that dynamic allocation requests for this TSO userid are (MOUNT) or are not (NOMOUNT) authorized to cause volume mounting as necessary.

- DEST *(userid)*
- NODEST

  specifies whether there is (DEST) or is not (NODEST) a default destination to which SYSOUT data sets dynamically allocated by this user will be routed. This default can be overridden by ALLOCATE, FREE, and other commands. This parameter is meaningful only when a TSO user is being created or modified.

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 command.

```
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 consisits of a user identification, a procedure name, and attributes (no password or account number).

```
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 User Attribute Data Set (UADS). Each terminal user has an entry in the UADS. Each entry contains several items of data. The data that you want to delete may be a part of an existing entry.

The DELETE subcommand of ACCOUNT is written as follows:

DELETE D	
ƀ	One or more blanks must follow DELETE or D.
(	
userid *	*userid:* 1-7 alphameric characters, beginning with an alphabetic or national character.
ƀ *password* ƀ *	*password:* 1-8 alphameric characters.
ƀ *acct nmbr* ƀ *	*acct nmbr:* up to 40 characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character. **Note:** This parameter may only be specified if the preceding parameter (*password* or *) is also specified.
)	
ƀ DATA *(password)* ƀ DATA *(acct nmbr)* ƀ DATA *(proc)*	*password:* list of passwords (1-8 alphameric characters), separated by blanks or commas. *acct nmbr:* list of account numbers (up to 40 characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character), separated by blanks or commas. *proc:* list of procedure names (1-8 alphameric characters, beginning with an alphabetic character), separated by commas or blanks.

The parameters are explained below:

(

specifies the beginning of a series of positional parameters.

userid
*

specifies a user identification that identifies the UADS entry to be deleted *(userid)* or specifies that all user identifications in the UADS are to be deleted(*).

ƀ *password*
ƀ *

specifies a word that a user must enter before he can use the system *(password)* or specifies all passwords for the indicated user identification(*). The password helps indicate the particular existing structure from which data is being deleted or, when you are deleting a password, the password is the data being deleted.

ƀ *acct nmbr*
ƀ *

specifies an account number used for administrative purposes *(acct nmbr)* or specifies all account numbers for the indicated password for the indicated user identification (*). The account number helps indicate the structure from which data is being deleted or, when you are deleting an account number, the password is the data being deleted.

)

specifies the end of a series of positional parameters.

- DATA *(password)*
- DATA *(acct nmbr)*
- DATA *(proc)*

specifies the data that is to be deleted from an existing entry:

password specifies a password or list of password to be deleted from the existing entry at the location indicated by the first positional parameter.

acct nmbr specifies an account number or list of account numbers to be deleted from the existing entry.

proc specifies a procedure name or list of procedure names to be deleted from the existing entry.

Deleting an Entire Entry: To delete an entire entry from the UADS, you only need to know the user identification for the entry. You must specify the user identification as the first and only field of the first positional parameter.

Deleting Data from an Existing Entry: To use the DELETE subcommand to delete data from an existing entry, you must identify:

a. The location within the entry.

b. The data that you want to delete.

Example 1

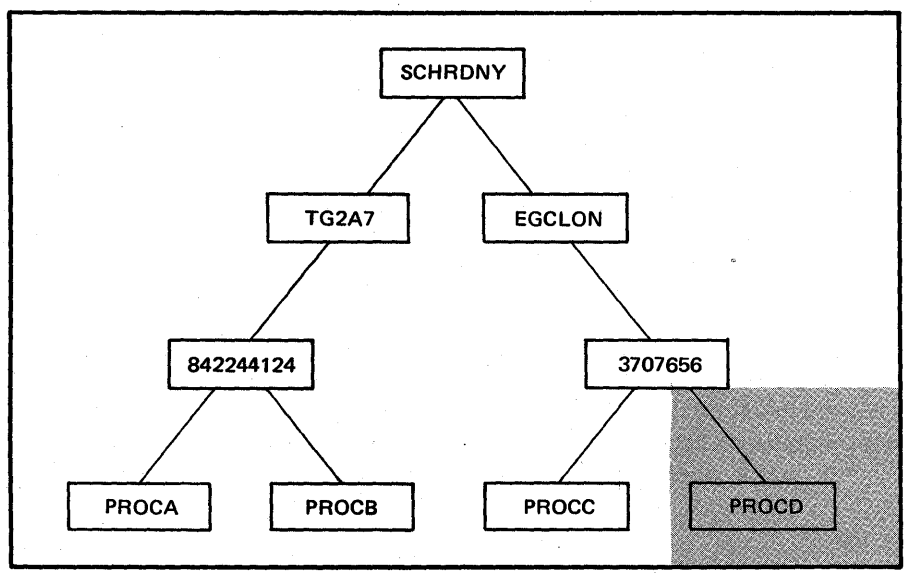
Operation: Delete an entire entry from the UADS.

```
delete (early08)
```

Example 2

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

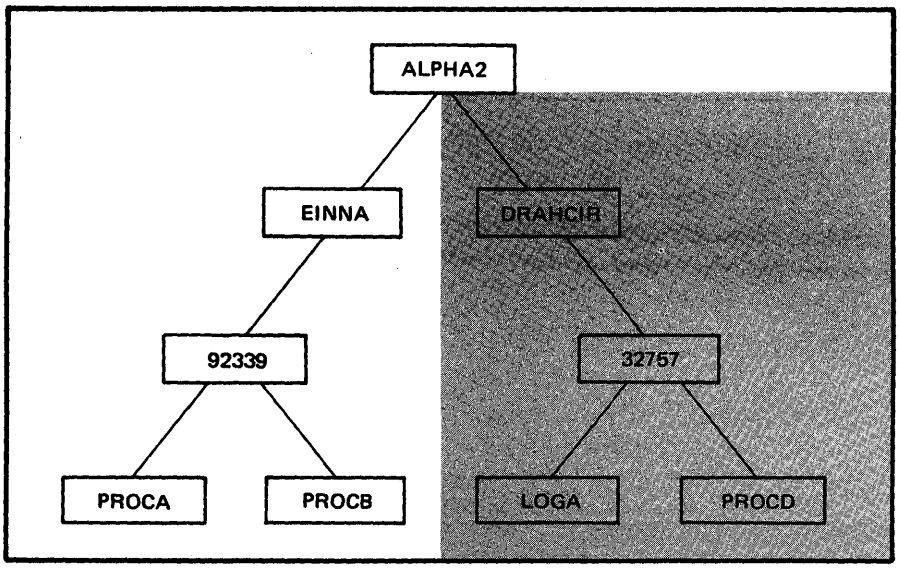
```
delete (schrdny egclon 3707656) data(procd)
```



Example 3

Operation: Delete an account number from an entry in the UADS having the following index 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 END subcommand of ACCOUNT is written as follows:

```
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 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 *OS/VS2 TSO Command Language Reference,* GC28-0646. The HELP subcommand of ACCOUNT is written as follows:

HELP
H

ъ *subcmd*	*subcmd:* symbol (name representing subcommand). **Default:** list of ACCOUNT subcommands.
ъ ALL ъ FUNCTION ъ SYNTAX ъ OPERANDS ъ OPERANDS *(parms)*	*parms:* list of symbols (names representing parameters), separated by commas or blanks. **Default:** ALL **Note:** This parameter may only be specified if the preceding parameter is also specified.

The parameters are explained below:

ъ *subcmd*
specifies the subcommand that you want to have clarified.

ъ ALL
ъ FUNCTION
ъ SYNTAX
ъ OPERANDS
ъ OPERANDS *(parms)*
specifies the explanatory information requested:

ALL specifies that you want a description of the function, the syntax, and the parameters of the subcommand that you specified.

FUNCTION specifies that you want a description of the referenced subcommand's function.

SYNTAX specifies that you want a definition of the proper syntax for the reference subcommand.

OPERANDS specifies that you want an explanation of the parameters applicable to the referenced subcommand.

parms specifies the particular keywords that you want to have explained. If you do not specify any keywords, all of the applicable keywords willbe included.

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 parameters for the CHANGE subcommand displayed at your terminal.

```
h change operands
```

LIST Subcommand of ACCOUNT

Use the LIST subcommand to display entries in the User Attribute Data Set (UADS) or to display fields of data from within particular entries.

The LIST subcommand of ACCOUNT is written as follows:

LIST L	
ƀ	One or more blanks must follow LIST or L.
(	
userid *	*userid:* 1-7 alphameric characters, beginning with an alphabetic or national character.
ƀ *password* ƀ *	*password:* 1-8 alphameric characters.
ƀ *acct nmbr* ƀ *	*acct nmbr:* up to 40 characters, not containing a blank, tab, quotation mark, apostrophe, comma, semicolon, or line control character. **Note:** This parameter may only be specified if the preceding parameter (*password* or *) is also specified.
ƀ *proc* ƀ *	*proc:* 1-8 alphameric characters, beginning with an alphabetic character. **Note:** This parameter may only be specified if the preceding parameter (*acct nmbr* or *) is also specified.
)	

The parameters are explained below:

(

specifies the beginning of a series of positional parameters.

userid
*

specifies a user identification that identifies the UADS entry to be listed *(userid)* or specifies that all user identifications in the UADS (*) are to be listed.

ƀ *password*
ƀ *

specifies a word that a user must enter before he can use the system *(password)* or specifies all passwords for the indicated user identification (*). The password helps indicate the structure to be displayed.

ƀ *acct nmbr*
ƀ *

specifies an account number used for administrative purposes *(acct nmbr)* or specifies all account numbers for the indicated password for the indicated user identification (*). The account number helps indicate the structure to be displayed.

ƀ *proc*
ƀ *

specifies the name of a procedure that is invoked when the user enters the LOGON command *(proc)* or specifies all procedure names for the indicated account number for the indicated password for the indicated user identification (*). The procedure name helps indicate the particular structure to be displayed.

)
specifies the end of a series of positional parameters.

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 User Attribute Data Set (UADS) displayed at your terminal.

The LISTIDS subcommand of ACCOUNT is written as follows:

LISTIDS
LISTI

Example 1

Operation: List all user identifications in the UADS.

```
listids
```

SYNC Subcommand of ACCOUNT

Use the SYNC subcommand to build a new Broadcast data set and synchronize it with the UADS by entering the userids from the UADS into the Broadcast data set. This causes the loss of all messages (MAIL) from the Broadcast data set. The SYNC subcommand is useful when the two data sets have gotten out of synchronization either because of I/O errors in SYS1.BRODCAST or because the installation has mounted and cataloged a new UADS.

The SYNC subcommand of ACCOUNT is written as follows:

SYNC

OPERATOR Command

Use the OPERATOR command (along with its subcommands) to regulate and maintain TSO from a terminal.

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.

This command may be used only by personnel who have been given the authority to do so by the installation management. The authority to use OPERATOR is normally given to personnel responsible for system operation, and is recorded in the User Attribute Data Set.

The OPERATOR command is written as follows:

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 the number of TSO users, the number of batch jobs, the TSO job messages that are awaiting a reply from the system operator, and the time of day and the date.
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.
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 terminal user. When you use the CANCEL command to terminate a terminal session, accounting information will be presented to the user.

The CANCEL subcommand of OPERATOR is written as follows:

CANCEL C	
b	One or more blanks must follow CANCEL or C.
U=*userid*	*userid:* 1-7 alphameric characters, beginning with an alphabetic or national character.
,DUMP	

The parameters are explained below:

U = *userid*
specifies the user identification for a user whose terminal session is to be terminated.

,DUMP
specifies that an abnormal-end-of-job storage dump is to be taken. The dumps will be printed on the system output device.

Example 1

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

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

DISPLAY Subcommand of OPERATOR

Use the DISPLAY subcommand to obtain a listing of:

- The number of terminal users for each time sharing region.
- The number of batch jobs executing.
- The messages from time sharing jobs that are awaiting replies from an operator.
- The time of day and the date.

The DISPLAY subcommand of OPERATOR is written as follows:

DISPLAY D	
ъ	One or more blanks must follow DISPLAY or D.
T TS R J JOBS	
,L ,LIST	**Note:** This parameter may not be used with T above.

The parameters are explained below:

T
TS
R
J
JOBS
specifies the information to be displayed at the terminal:

T specifies that you want the time of day and the date.

TS specifies that you want the number of TSO users currently logged on the system.

R specifies that you want a listing of the ids of messages that are awaiting a response from an operator.

J and JOBS specify that you want the number of batch jobs executing.

,L
,LIST
specifies that you want the following additional information displayed:

For TS, a list of TSO userids currently logged on the system.

For R, a list of the beginnings of the messages corresponding to the ids.

For J and JOBS, a list of the jobnames and V=R region boundaries.

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
```

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 END subcommand of OPERATOR is written as follows:

```
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 **OS/VS2 TSO Command Language Reference**, GC28-0646, if you desire more detailed information.

The HELP subcommand of OPERATOR is written as follows:

```
HELP
H
```

ƀ *subcmd*	*subcmd:* symbol (name representing subcommand). **Default:** list of OPERATOR subcommands.
ƀ ALL ƀ FUNCTION ƀ SYNTAX ƀ OPERANDS ƀ OPERANDS *(parms)*	*parms:* list of symbols (name representing parameters), separated by commas or blanks. **Default:** ALL **Note:** This parameter may only be specified if the preceding parameter is also specified.

The parameters are explained below:

ƀ *subcmd*
specifies the subcommand that you want to have clarified.

ƀ ALL
ƀ FUNCTION
ƀ SYNTAX
ƀ OPERANDS
ƀ OPERANDS *(parms)*
specifies the explanatory information requested:

ALL specifies that you want a description of the function, syntax, and the parameters of the subcommand that you specified.

FUNCTION specifies that you want a description of the referenced subcommand's function.

SYNTAX specifies that you want a definition of the proper syntax for the referenced subcommand.

OPERANDS specifies that you want an explanation of the parameters applicable to the referenced subcommand.

parms specifies the particular keywords that you want to have explained. If you do not specify any keywords, all of the applicable keywords will be included.

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 MONITOR subcommand of OPERATOR is written as follows:

MONITOR MN	
b	One or more blanks must follow MONITOR or MN.
SESS STATUS JOBNAMES	
,T	**Note:** This parameter may only be specified with SESS or JOBNAMES above.

The parameters are explained below:

SESS
STATUS
JOBNAMES
specifies the terminal and background job activities to be monitored:

SESS indicates that you are to be notified whenever any terminal session is initiated or terminated. The user's identification will be displayed at your terminal. If the session terminates abnormally, the user identification will appear in the diagnostic message; the message "user LOGGED OFF" will not appear if the session was canceled.

If you specify the T parameter, the system displays the time of day in addition to the users identification. The format of the time output is shown under the T parameter description.

STATUS specifies that you want the data set names and volume serial numbers of data sets with dispositions of KEEP, CATLG, or UNCATLG to be displayed whenever the data sets are freed.

JOBNAMES specifies that you want the name of each job to be displayed both when the job starts and when it terminates, and that you want unit record allocation to be displayed when the job step starts. If a job terminates abnormally, the jobname will appear in the diagnostic message; the message 'jobname ENDED' will not appear.

If you specify the T parameter with the JOBNAMES parameter, the system displays the time of the day in addition to the jobnames. The format of the output is shown under the T parameter description.

,T
specifies that you want the time of day to be displayed in the following format.
hh.mm.ss
The variables in this format are:
hh - Hours (00-23)
mm - Minutes (00-59)
ss - Seconds (00-59)
Whenever one TSO user specifies this parameter, all subsequent users of the MONITOR command will also receive the time at their terminals.

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, save a message in the SYS1.BRODCAST data set, list, delete, or send a specified message from notices section of the SYS1.BRODCAST data set, and list all messages in the notices section of SYS1.BRODCAST data set. 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 by a console operator.

The SEND subcommand of OPERATOR is written as follows:

SEND SE	
b	One or more blanks must follow SEND or SE.
'msg' *msg nmbr* LIST	*msg:* up to 115 characters. *msg nmbr:* decimal digits. **Note:** If LIST is specified, no other parameters may be specified.
,ALL ,USER=*(user ids)* ,BRDCST ,OPERATOR=*rte code* ,CN=*console id* ,LIST ,DELETE	*user ids:* 1-7 alphameric characters, beginning with an alphameric or national character. IDs are separated by commas. *rte code:* decimal digits 1-15. *console id:* decimal digits 0-64. **Default:** ALL **Note:** LIST and DELETE may not be specified with *'msg'* above.
NOW LOGON SAVE	**Default:** NOW **Note:** This parameter may only be specified with USER above.

The parameters are explained below:

'msg'
msg nmbr
LIST

specifies the message you want to send *('msg')* or the number of a notice in the SYS1.BRODCAST data set *(msg nmbr)*. LIST specifies that a listing of all the SEND notices retained in the system is to be produced at your terminal; each message will be preceded by a system-assigned number.

If *'msg'* is specified, you must enclose the text of the message within apostrophes (as indicated). The message must be contained on one line (you cannot continue a message on a second line). If you want a quotation mark printed in the message, you must enter two quotation marks.

,ALL
,USER=*(user ids)*
,BRDCST
,OPERATOR=*rte code*
,CN=*console id*
,LIST
,DELETE

specifies additional information about the message to be sent:

ALL specifies that all terminal users are to receive the message. Terminal users who are currently using the system will receive the message immediately.

USER specifies the user identification of one or more terminal users who are to receive the message. The USERID must already exist in the Broadcast Data Set. This is done automatically via the ACCOUNT command when updating the UADS.

BRDCST specifies that the message is to be queued to all active consoles.

OPERATOR specifies that the message is to be queued to the console associated with the routing code assigned.

CN specifies that the message is to be queued to the operator console indicated by the console identification. (The console configuration message issued at IPL can be used for reference to determine the appropriate ids.) Console id 0 causes the message to be sent to the master console, and is substituted if the specified console id is a 2-digit value outside the valid range.

LIST is defined above.

DELETE specifies that you want a message to be deleted.

NOW
LOGON
SAVE

specifies how the message is to be handled:

NOW specifies that the message is to be sent immediately. If the recipient is not logged on, you will be notified and the message will be deleted.

LOGON specifies that the message is to be sent immediately if the recipients are logged on and receiving messages. Otherwise:

- If you specify a user identification, the message is retained in the "mail" section of the SYS1.BRODCAST data set and deleted by the system after it is sent to the intended user.
- If you specify "ALL", the message will be stored in the "notices" section of the SYS1.BRODCAST data set and retained there until the operator deletes it.

SAVE specifies that the message text is to be entered in the appropriate section of the SYS1.BRODCAST data set without being sent to any user. If you specify a user identification, the message is stored in the mail section of SYS1.BRODCAST data set and is deleted by the system after it is sent to the intended user. If you specify ALL, the message will be stored in the notices section of SYS1.BRODCAST data set and retained there until the operator deletes it.

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
```

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 STOPMN subcommand of OPERATOR is written as follows:

STOPMN PM	
b	One or more blanks must follow STOPMN or PM.
JOBNAMES SESS STATUS	

The parameters are explained below:

JOBNAMES
SESS
STATUS
specifies the monitoring operations to be terminated:

JOBNAMES specifies that the operations provided by the JOBNAMES parameter of the MONITOR subcommand are to be stopped. The system will stop displaying the names as they start and end.

SESS specifies that the operations provided by the SESS parameter of the MONITOR subcommand are to be stopped. The system will stop notifying the terminal whenever a terminal session is initiated or terminated.

STATUS specifies that the operations provided by the STATUS parameter of the MONITOR subcommand are to be stopped. The system will stop displaying the names and volume serial numbers of data sets with dispositions of KEEP, CATLG, or UNCATLG at job step end and job end.

Example 1

Operation: Stop the display of the names of jobs as they begin execution and terminate.

```
stopmn jobnames
```

Example 2

Operation: Stop the display of starting and stopping of terminal session.

```
pm  sess
```

Index

OS/VS2 System Programming Library:
TSO
GC28-0629-1

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

What is your occupation? _____

Number of latest Technical Newsletter (if any) concerning this publication: _____

Please indicate your address in the space below if you wish a reply.

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.)

Cut or Fold Along Line

Cut or Fold Along Line

Your comments, please . . .

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

First Class
Permit 81
Poughkeepsie
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department D58, Building 706-2
PO Box 390
Poughkeepsie, New York 12602

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

OS/VS2 System Programming Library: TSO (S370-39) Printed in U.S.A. GC28-0629-1

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)