VMS Access Control List Editor Manual

Order Number: AA–LA41A–TE

April 1988

This document describes the VMS Access Control List Editor.

Revision/Update Information:	This document supersedes the *VAX/VMS Access Control List Editor Reference Manual,* Version 4.4.
Software Version:	VMS Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital ™

ZK4531

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*	CANADA	INTERNATIONAL
Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario K2K 2A6 Attn: Direct Order Desk	Digital Equipment Corporation PSG Business Manager c/o Digital's local subsidiary or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.
In New Hampshire, Alaska, and Hawaii call 603-884-6660.
In Canada call 800-267-6215.
*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).
Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript™ printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

™ PostScript is a trademark of Adobe Systems, Inc.

Contents

PREFACE vii

NEW AND CHANGED FEATURES ix

ACL Editor Description ACL–1

1 AN OVERVIEW OF ACLS ACL–1

2 INVOKING THE ACL EDITOR ACL–2

3 KEYPAD EDITING ACL–3
3.1 ACL Editing Commands ACL–3
3.2 VT200-Only ACL Editing Commands ACL–10
3.2.1 Using the PASTE Buffer • ACL–11
3.3 Control Key ACL Editing Commands ACL–11
3.4 Terminating the ACL Editing Session ACL–12

4 RECOVERING AN ACL EDITING SESSION ACL–12

5 ACCESS CONTROL LISTS ACL–13
5.1 Identifier ACEs ACL–13
5.1.1 Specifying Identifiers in Identifier ACEs • ACL–13
5.1.2 Specifying Options in Identifier ACEs • ACL–14
5.1.3 Specifying Access in Identifier ACEs • ACL–15
5.1.4 Sample Identifier ACEs • ACL–16
5.2 Default Protection ACE ACL–17
5.3 Security Alarm ACE ACL–18

6 CUSTOMIZING THE ACL EDITOR ACL–19
6.1 Modifying Variables in the ACL Section File ACL–19
6.2 Using the ACL Editor CALL_USER Routine ACL–21

ACL Editor Usage Summary ACL–23

ACL Editor Qualifiers ACL–24

/JOURNAL ACL–25
/MODE ACL–26
/OBJECT ACL–27
/RECOVER ACL–28

INDEX

FIGURES

ACL–1 VT200 Keypad ACL–4

Preface

Intended Audience

This manual is intended for all system users.

Document Structure

This document consists of the following three sections:

- Description—Provides an overview and detailed usage information for the Access Control List (ACL) Editor; describes the keypad commands used to create and modify ACLs.
- Usage Summary—Outlines the following ACL editor information:

  -Invoking the editor
  -Exiting from the editor
  -Directing output
  -Restrictions or privileges required
- Qualifiers—Describes ACL editor qualifiers, including format, parameters, and examples.

Associated Documents

To learn more about access control lists, refer to the *Guide to VMS System Security*.

If you are a system programmer and plan on modifying the VAXTPU ACL section file from which the ACL editor was built, you should be familiar with the *VAX Text Processing Utility Manual*. It describes the VAXTPU programming language and the concepts involved in modifying and processing a VAXTPU section file.

Conventions

The following conventions are observed in this manual:

Convention	Meaning
$\boxed{\text{RET}}$	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.
$ SHOW TIME 05-JUN-1988 11:55:22	In examples, system output (what the system displays) is shown in black. User input (what you enter) is shown in red.
$ TYPE MYFILE.DAT . . .	In examples, a vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to a command is shown or that not all the data a user would enter is shown.
input-file, . . .	In examples, a horizontal ellipsis indicates that additional parameters, values, or other information can be entered, that preceding items can be repeated one or more times, or that optional arguments in a statement have been omitted.
[logical-name]	Brackets indicate that the enclosed item is optional. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

New and Changed Features

The following technical features have been added to the *VMS Access Control List Editor Manual* for VMS Version 5.0:

- Support for access control lists (ACLs) on batch and device (printer, server, and terminal) queues. Specify the keyword QUEUE to the /OBJECT qualifier to edit ACLs on queues.
- Addition of a new CALL_USER routine function code ACLEDIT$C_MESSAGE. ACLEDIT$C_MESSAGE assumes the input string is a VMS error code and returns in the ACL editor message window the message text associated with the error code.

ACL Editor Description

An access control list (ACL) is a collection of entries that grant or deny access for specific users or groups of users of a system object. By carefully defining the individual access control list entries (ACEs) that make up an ACL, you can control user access to a particular object more closely than through the use of the default UIC-based protection scheme.

The Access Control List (ACL) editor is a screen-oriented editor used to create and maintain ACLs. You can use ACLs to define types of access for users of a system object, such as a file.

The description section of the ACL editor is divided into the following six parts:

- Section 1 provides an overview of access control lists (ACLs).
- Section 2 describes how to invoke the ACL editor.
- Section 3 displays the keypad commands available for use during an ACL editing session.
- Section 4 explains how to recover from an editing session that was interrupted abnormally.
- Section 5 describes the different types of ACEs available in an access control list.
- Section 6 describes how to customize the ACL editor.

1 An Overview of ACLs

An access control list consists of access control list entries (ACEs) that grant or deny access to a particular system object. You can place ACLs on the following types of objects:

- Devices
- Files (including directory files)
- Group global sections
- System global sections
- Logical name tables
- Queues

Typically, ACLs are used when you want to provide access to a system object for some, but not all users. When the VMS operating system receives a request for access to an object having an ACL, it searches each access control list entry in the ACL, stopping at the first match. If another match occurs in the ACL, it has no effect. Therefore, ACEs granting or denying access to a system object for specific users should appear in the ACL before ACEs identifying broader classes of users.

For example, if you want to grant user SMITH read access to a system object and deny all other interactive users all types of access to the object, place the ACE for user SMITH before the ACE identifying all interactive users on the system.

The use of ACLs is optional. Using ACLs permits more detail in the defining of user access and can enhance the security of system objects. However, creating and maintaining ACLs requires both user and processor time.

Each ACL consists of one or more ACEs. There is no limit to the number of ACEs that an ACL can contain, or to the number of characters in an ACE. However, long ACLs increase the amount of time necessary to gain access to an object.

The following types of ACEs are available in an ACL:

- Identifier
- Default protection
- Security alarm

Refer to Section 5 for a description of these ACEs.

2 Invoking the ACL Editor

Use the ACL editor to define an ACL for a system object or to edit an existing ACL. Invoke the ACL editor using the DCL command EDIT/ACL followed by the name of the object whose ACL you want to create or modify. For example, the following command invokes the ACL editor to create an ACL for the file INVENTORY.DAT:

```
$ EDIT/ACL INVENTORY.DAT
```

You can also invoke the ACL editor using the DCL command SET ACL/EDIT. In addition to invoking the ACL editor, the SET ACL command can be used to manipulate entire ACLs or individual ACEs without invoking the ACL editor.

If the object whose ACL you want to create or modify is not a file, you must specify the type of object with the /OBJECT=type qualifier. For example, the following command invokes the ACL editor to create an ACL for the disk DOCD$:

```
$ EDIT/ACL/OBJECT=DEVICE DOCD$
```

You can invoke the ACL editor to modify an existing ACL or to create a new ACL on the system object. If a system object has an ACL, the ACL will appear on the screen when the ACL editor is invoked.

The ACL editor can be invoked from within a program written in any VAX language that generates calls using the VAX Calling Standard. Refer to the *VMS Utility Routines Manual* for more information on using the callable interface to the ACL editor.

3 Keypad Editing

The ACL editor provides several features that simplify its use, including the following:

- Automatic text insertion (prompt mode)
- Dynamic ACE syntax checking
- Keypad editing that includes online help

By default, the ACL editor prompts for each ACE and provides values in the various fields within an ACE whenever possible. The /MODE qualifier controls the choice of mode. To disable prompting, specify /MODE=NOPROMPT with the SET ACL or EDIT/ACL command.

The FIELD, ITEM, and ENTER commands on the keypad enable you to take full advantage of prompt mode in the ACL editor.

- FIELD—Completes the current ACE field and moves the cursor to the next ACE field or subfield, inserting text as needed. If the ACL editor is not in prompt mode, the ACL editor advances to the next field in the current existing ACE.
- ITEM—Selects the next item for the current ACE field. If the ACL editor is not in prompt mode, this key is ignored.
- ENTER—Indicates that the current ACE is complete. The ACL editor terminates the insertion and verifies that the syntax of the ACE is complete. You can press ENTER while the cursor is located at any position within the ACE. (Performs the same function as the RETURN key.)

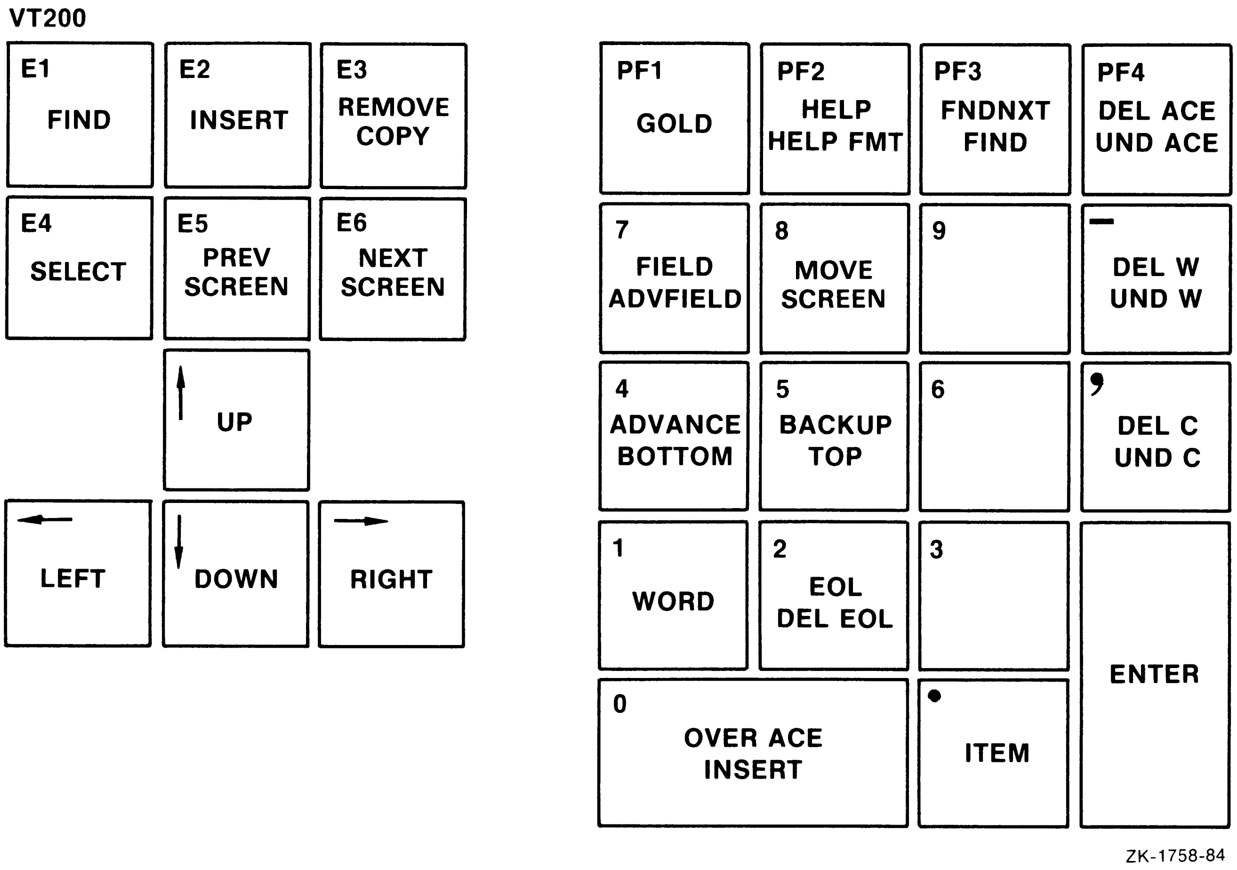
Access the online help facility of the ACL editor by using the HELP and HELP FMT commands on the keypad. The HELP command provides help on the editing keypad; the HELP FMT command provides help on ACE.

The following section contains a diagram of the default keypad functions for VT100 and VT200 series terminals. Also included in the section is a functional description of each of the ACL editing commands. You can supplement or change these key definitions by modifying and recompiling the ACL editor section file SYS$LIBRARY:ACLEDIT.TPU. Refer to Section 6 for more information if you intend to modify the default ACL section file.

3.1 ACL Editing Commands

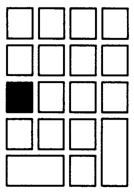
Figure ACL–1 depicts the default ACL editor keypad functions for VT200 series terminals. The numeric keypad on VT100 series terminals is identical to that of the VT200 terminal shown in Figure ACL-1; VT100 terminals, however, do not have the supplemental editing keypad (keys E1 through E6).

Figure ACL–1 VT200 Keypad



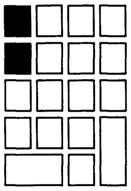
On the VT100 and VT200 series terminals, you can use the following keypad commands during an ACL editing session:

ADVANCE



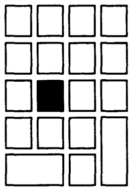
Sets the current direction forward for the FIND, FNDNXT, MOVE SCREEN, OVER ACE, and WORD keys. ADVANCE means that movement is toward the end of the ACL.

ADV FIELD



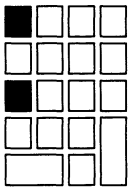
Completes the current ACE field and moves the cursor to the next ACE field.

BACKUP



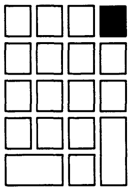
Reverses the current direction for the FIND, FNDNXT, MOVE SCREEN, OVER ACE, and WORD keys. BACKUP means that movement is toward the beginning of the ACL.

BOTTOM



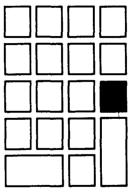
Sets the cursor position after the last line of the last ACE. Any entries you add are placed at the end of the ACL.

DEL ACE



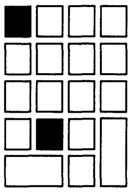
Deletes the entire ACE in which the cursor is positioned and stores it in the delete-ACE buffer.

DEL C



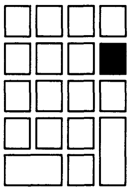
Deletes the character on which the cursor is positioned and stores it in the delete-character buffer.

DEL EOL



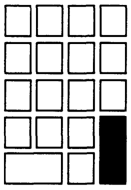
Deletes text from the current cursor position to the end of the line and stores it in the delete-line buffer.

DEL W



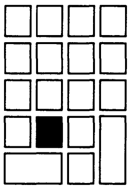
Deletes the text from the current cursor position to the beginning of the next word and stores it in the delete-word buffer.

ENTER



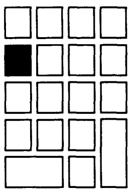
Indicates that the current ACE is complete. The ACL editor terminates the insertion and verifies the syntax of the ACE. You can press ENTER while the cursor is located at any position within the ACE. (Pressing RETURN produces the same results.)

EOL



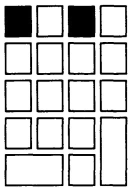
Moves the cursor to the end of the current line.

FIELD



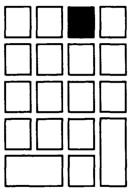
Completes the current ACE field and moves the cursor to the next ACE field or subfield.

FIND



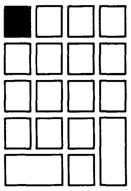
Searches for an occurrence of a string. Press the FIND key and then type the string. Press the ENTER key to search for the string in the current direction, or the ADVANCE or BACKUP key to change the search direction.

FNDNXT



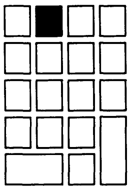
Searches in the current direction for the next occurrence of the string previously entered with the FIND key.

GOLD



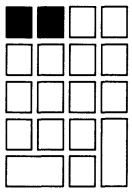
When pressed before another keypad key, specifies the second key's alternate function (the bottom function on the keypad diagram).

HELP



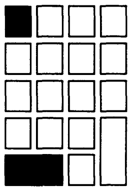
Use the HELP key to get information about using the editing keypad.

HELP FMT



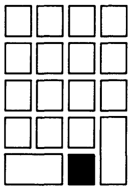
Press the GOLD key followed by the HELP key to get information about ACE formats.

INSERT



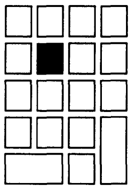
Moves all text from the current line down one line, leaving a blank line where an ACE is to be inserted.

ITEM



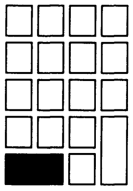
Selects the next item for the current ACE field.

MOVE SCREEN



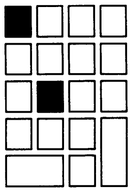
Moves the cursor one screen in the current direction (see ADVANCE or BACKUP). A screen is defined as two-thirds the number of lines in the display.

OVER ACE



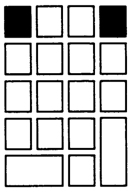
Depending on the current direction, moves the cursor to the beginning of the next ACE or the beginning of the previous ACE.

TOP



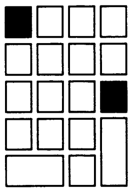
Moves the cursor position to the first character of the first ACE in the access control list.

UND ACE



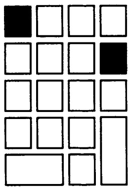
Inserts the contents of the delete-ACE buffer in front of the ACE in which the cursor is currently positioned.

UND C



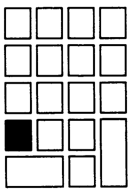
Inserts the contents of the delete-character buffer directly in front of the cursor.

UND W



Inserts the contents of the delete-word buffer directly in front of the cursor.

WORD



Moves the cursor one word forward or backward, depending on the current direction.

The following keys and key sequences can be used to move the cursor or to shift the display window:

- ↓—Moves the cursor to the character directly in line below it. If the ACE in which the cursor is positioned is new, the ACL editor processes the ACE before moving the cursor. If the entry is incomplete or improperly formatted, an error occurs and the cursor does not move.
- ←—Moves the cursor one character to the left. If the cursor is at the left margin, moves it to the rightmost character in the line above.
- →—Moves the cursor one character to the right. If the cursor is at the right margin, moves it to the leftmost character in the line below.
- ↑—Moves the cursor to the character directly in line above it. If the ACE in which the cursor is positioned is new, the ACL editor processes the ACE before moving the cursor. If the entry is incomplete or improperly formatted, an error occurs and the cursor does not move.
- GOLD ←—Shifts the text in the display window 8 characters to the left.
- GOLD →—Shifts the text in the display window 8 characters to the right.

You can use the following keyboard keys to supplement the keypad keys. Keys in parentheses indicate the equivalent key for a VT200 series terminal.

- BACKSPACE (F12)—Moves the cursor to the beginning of the current line.
- DELETE (⌫)—Deletes the character to the left of the cursor.
- LINE FEED (F13)—Deletes the text from the cursor back to the beginning of the word. If the cursor is positioned at the first character of the word, deletes to the beginning of the previous word.
- TAB (TAB)—Moves the text located to the right of the cursor to the next tab stop.

3.2 VT200-Only ACL Editing Commands

On a VT200 series terminal, you can also use the supplemental editing keypad commands during an ACL editing session:

- FIND—Elicits the **Search for:** prompt as the first step in the FIND operation. Type the search string after the prompt; then press either the DO or the ENTER key to process the search. Performs the same function as the FIND keypad command.
- INSERT HERE—Indicates where an ACE is to be inserted, or, if support for the PASTE buffer is enabled, indicates the line where the selected text in the PASTE buffer is to be inserted. By default, support for the PASTE buffer is disabled; Section 3.2.1 explains how to enable the PASTE buffer.
- REMOVE—Removes the selected text to the PASTE buffer. Each time REMOVE is used, the previous contents of the PASTE buffer are deleted. By default, support for the PASTE buffer is disabled; Section 3.2.1 explains how to enable the PASTE buffer.

- COPY (GOLD REMOVE)—Copies the selected text to the PASTE buffer. Each time COPY is used, the previous contents of the PASTE buffer are deleted. By default, support for the PASTE buffer is disabled; Section 3.2.1 explains how to enable the PASTE buffer.
- SELECT—Marks the beginning of a range of text to be removed or copied to the PASTE buffer. Press SELECT; move the cursor to include the desired amount of text to be removed or copied; press either REMOVE or COPY (GOLD REMOVE) to complete the operation. By default, support for the PASTE buffer is disabled; Section 3.2.1 explains how to enable the PASTE buffer.
- PREV SCREEN—Moves the cursor to the previous screen. By default, a screen is defined as two-thirds the number of lines in the display.
- NEXT SCREEN—Moves the cursor one screen forward. By default, a screen is defined as two-thirds the number of lines in the display.

3.2.1 Using the PASTE Buffer

For VT200 series terminals, the ACL editor provides a PASTE buffer that allows you to move sections of text from one part of the ACL to another. The commands used to move sections of the ACL in and out of the PASTE buffer are INSERT HERE, REMOVE, COPY, and SELECT.

By default, support for the PASTE buffer is disabled. To enable the PASTE buffer, do either of the following:

- To enable the PASTE buffer for the current editing session only, press CTRL/D. At the **TPU command:** prompt, type the following:

  ```
  TPU command: ACLEDIT$X_PASTE_BUFFER:=1
  ```

- To permanently enable the PASTE buffer, you must change the value of the variable ACLEDIT$X_PASTE_BUFFER in the ACL editor section file from 0 to 1 and recompile the ACL section file. Section 6 describes how to modify and recompile this file.

If you enable the PASTE buffer, you should also switch the value of the variable ACLEDIT$X_CHECK_MODIFY from 1 (check whether the ACE can be modified) to 0 (do not check whether the ACE can be modified). The two features (PASTE buffer support and the check for a modifiable ACE) are not usually compatible. To change the value of ACLEDIT$X_CHECK_MODIFY, use the same method you used previously to enable the PASTE buffer: edit and recompile the ACL section file to make the change permanent, or press CTRL/D to make the change for the current editing session only.

3.3 Control Key ACL Editing Commands

The following control keys also perform editing functions:

- CTRL/A—Determines whether characters are entered in insert mode or overstrike mode. Insert mode (the default) inserts a character to the left of the current character. Overstrike mode replaces the current character.
- CTRL/D—Allows you to execute one TPU command.
- CTRL/H—Moves the cursor to the beginning of the line. (Performs the same function as the BACKSPACE key.)

- CTRL/J—Deletes the text from the cursor back to the beginning of the word. (Performs the same function as the LINE FEED key.)
- CTRL/R—Refreshes the screen display. Clears and redraws the screen, deleting any extraneous characters or messages that might have appeared on the screen but are not part of the ACL you are editing. (Performs the same function as the CTRL/W key.)
- GOLD CTRL/R—Returns the ACL to its original state before the ACL editor was invoked. (Performs the same function as GOLD CTRL/W.)
- CTRL/U—Deletes the text from the cursor to the beginning of the line.
- CTRL/W—Refreshes the screen display. Clears and redraws the screen, deleting any extraneous characters or messages that might have appeared on the screen but are not part of the ACL you are editing. (Performs the same function as the CTRL/R key.)
- GOLD CTRL/W—Returns the ACL to its original state before the ACL editor was invoked. (Performs the same function as GOLD CTRL/R.)
- CTRL/Z—Ends the editing session and updates the ACL. (Unless otherwise specified, any recover and journal files are deleted.)
- GOLD CTRL/Z—Ends (quits) the editing session without saving any of the changes made to the object's ACL. (Unless otherwise specified, any recover and journal files are deleted.)

3.4 Terminating the ACL Editing Session

During an editing session, you can return to the original ACL (before any changes were made) and start over by pressing GOLD CTRL/R.

Press CTRL/Z to exit from a completed editing session. Your changes are not made until you exit. Because it is necessary to obtain exclusive access to a file to update an ACL, other users of a shared file may get a file-locked (or access conflict for directory files) error message when the ACL is updated at the end of the editing session.

If you do not want to incorporate the edits into the ACL, quit the file by pressing GOLD CTRL/Z. All edits made during the session are ignored.

4 Recovering an ACL Editing Session

By default, if an editing session ends abnormally, a journal file is created. A journal file contains the data from the editing session that was interrupted. The journal file name defaults to the file name of the edited file with a JOU extension. You can use the /JOURNAL qualifier to specify an alternate journal file name. You can also use the /NOJOURNAL qualifier to override the behavior and prevent the creation of a journal file.

To recover an interrupted editing session, use the /RECOVER qualifier. If the journal file name is different from the default, specify it with /RECOVER.

For more information on any of these EDIT/ACL qualifiers, see the Qualifier section of this manual.

5 Access Control Lists

This section describes the different types of access control list entries (ACEs) that make up an access control list (ACL). The type of access protection needed determines the type of ACE used in a given situation. Following are the three types of ACEs:

- Identifier—Controls the type of access allowed to a particular user or group of users.
- Default protection—Defines the default protection placed on all files and subdirectories in a directory. Applicable only to directory files.
- Security alarm—Provides a security alarm that indicates when an object is accessed in a specified way.

The exact format of an ACE depends on its type, but all ACEs are enclosed in parentheses. Following is the standard ACE format:

```
(type[,options][,access_to_grant])
```

5.1 Identifier ACEs

An identifier ACE controls the types of access allowed to specific users based on user identification. Following is the format for an identifier ACE:

```
(IDENTIFIER=identifier[,options][,access])
```

5.1.1 Specifying Identifiers in Identifier ACEs

The first field in the identifier ACE is the keyword IDENTIFIER followed by one or more identifiers. An identifier can be one of the following:

- User identification code (UIC)
- General, established by the system manager in the system rights database
- System-defined

A UIC can be in either numeric or named UIC format, as described in the *VMS DCL Concepts Manual.*

A general identifier, defined in the system rights database, is an alphanumeric string of 1 to 31 characters that must contain at least one alphabetic character. It can include the characters A through Z, dollar signs ($), underscores (_), and the numbers 0 through 9.

The system manager creates and assigns general identifiers and UICs to system users using the Authorize Utility (AUTHORIZE).

System-defined identifiers are automatically defined by the system when the system manager creates a rights database. The following identifiers are system-defined identifiers:

BATCH	All attempts at access made by batch jobs
NETWORK	All attempts at access made over the DECnet-VAX network
INTERACTIVE	All attempts at access made by interactive processes

LOCAL	All attempts at access made by users logged in at local terminals
DIALUP	All attempts at access made by users logged in at dial-up terminals
REMOTE	All attempts at access made by users logged in via a network

Generally, use only one of the six system-defined identifiers at a time. You can use them with other identifiers (UICs and general identifiers). When you specify multiple identifiers, connect them with plus signs (+).

The system takes the access action included in the ACE only for the user who matches all the identifiers. For example, if you wanted to grant read access to user [301,25] running a batch job, you would specify the identifier ACE as the following:

```
(IDENTIFIER=[301,25]+BATCH,ACCESS=READ)
```

Although it is unusual for a number of users to share the same UIC, it is likely that a number of users will share the same general identifier. Users with the same general identifier do not need to be in the same UIC-based group. Furthermore, a single user can be associated with a number of different general identifiers as defined in the rights database. The creator of an ACL has considerable flexibility in selecting sets of users and defining access capabilities for them.

For example, the user identified by the UIC [301,25] is a member of the UIC-based group 301. That user may be the only member of group 301 who is also associated with the general identifier PERSONNEL. An ACE defining a particular type of access for the users associated with the general identifier PERSONNEL grants that type of access to that user, but not to the other members of group 301.

5.1.2 Specifying Options in Identifier ACEs

The options field in an identifier ACE controls whether an ACE is propagated, can be displayed, or can be deleted. This field in an identifier ACE begins with the keyword OPTIONS and takes one or more of the following keywords:

DEFAULT	Indicates that an ACE is to be included in the ACL of any files created within a directory. When the ACE is propagated, the DEFAULT indicator is removed from the ACL of the created file. This option is valid only for directory files. A default ACE does not grant or deny access; it just affects the ACL of new files.
HIDDEN	Indicates that this ACE should only be changed by the application that added it. The ACL editor does not permit modification or deletion. Thus, the ACL editor displays the ACE only to show its relative position within the ACL, not to facilitate editing of the ACE. The DCL DIRECTORY and SHOW ACL commands do not display hidden ACEs.
PROTECTED	Indicates that an ACE will be preserved even when an attempt is made to delete the entire ACL. A protected ACE must be deleted specifically with the ACL editor or by specifying the ACE on the command line of the DCL command SET ACL.

NOPROPAGATE	Indicates that, when copying an ACL from one version of a file to a later version of the same file, the ACE is not copied to the newer version.
NONE	Indicates that no options apply to an ACE. Although you can enter OPTIONS=NONE when you create the ACE, OPTIONS=NONE is not displayed when the ACE is displayed.

Connect multiple options with plus signs (+). If you specify any other options with the NONE option, the other options take precedence.

Identifier ACE for a Directory

The OPTIONS=DEFAULT option of an identifier ACE allows users to define one or more default ACEs for inclusion in the ACLs for files created in a particular directory. A default ACE is supplied for all new files created in that directory; any existing files are not supplied with the default ACE. Thus, if you wanted all files in the directory [MALCOLM] to have an ACE that permitted read and write access to users with the PERSONNEL identifier, you could include the following ACE in the ACL for the file MALCOLM.DIR:

```
(IDENTIFIER=PERSONNEL,OPTIONS=DEFAULT,ACCESS=READ+WRITE)
```

As a result of this ACE, any file created in the [MALCOLM] directory has the following ACE:

```
(IDENTIFIER=PERSONNEL,ACCESS=READ+WRITE)
```

Notice that the DEFAULT option does not appear in the file's ACE. However, any subdirectory created in the MALCOLM directory has the DEFAULT option as part of its ACE so that the default ACE can be propagated throughout the entire directory tree.

5.1.3 Specifying Access in Identifier ACEs

The third field in an identifier ACE specifies what type of access you are allowing the users identified in the first field of the ACE. This field begins with the keyword ACCESS followed by a string of access actions connected by plus signs. The following types of access are allowed in an identifier ACE:

READ	Accessor can read a file, read from a disk, or allocate a device.
WRITE	Accessor can read or write a file.
EXECUTE	Accessor can execute an image file or look up entries in a directory by explicitly specifying file names.
DELETE	Accessor can delete a file.
CONTROL	Accessor has all the privileges of the object's owner.
NONE	Accessor has no access to the object.

5.1.4 Sample Identifier ACEs

The most common type of ACL is one that defines the access to a file for a group of users. In the following ACL example, access to a file is based on the identity of a user. PERSONNEL, SECURITY, and SECRETARIES are general identifiers assigned to appropriate sets of users by the system manager using AUTHORIZE. NETWORK is a system-defined identifier, while [20,*] and [SALES,JONES] are examples of UIC identifiers.

```
(IDENTIFIER=SECURITY,OPTIONS=PROTECTED,ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL)
(IDENTIFIER=PERSONNEL,ACCESS=READ+WRITE+EXECUTE+DELETE)
(IDENTIFIER=SECRETARIES,ACCESS=READ+WRITE)
(IDENTIFIER=[20,*],ACCESS=READ)
(IDENTIFIER=NETWORK,ACCESS=NONE)
(IDENTIFIER=[SALES,JONES],ACCESS=NONE)
```

In the preceding example, the ACE providing the greatest amount of file access is listed at the top of the ACL. Any users holding both the SECURITY and PERSONNEL identifiers obtain maximum access rights through the first match, which is the SECURITY identifier. In this example, the user with UIC [SALES,JONES] is prohibited from any access to the file, unless that user also happens to have one of the general identifiers (which is an oversight on the part of the creator of the ACL). If the ACL creator wants to be absolutely certain that the user with UIC [SALES,JONES] could not possibly gain access to the file, the ACE at the bottom of the ACL should be moved to the top.

The order of the ACEs in the example permits a number of users to gain types of file access over the DECnet-VAX network. The users with the identifiers of SECURITY, PERSONNEL, SECRETARIES, and UIC [20,*] can all gain some access over the network, although only those with the identifier SECURITY can gain full access. The fifth ACE prevents all other users from network access. While this might be the intent of the ACL creator, it would be an unfortunate oversight if it were not. Remember that the system searches the ACL sequentially and grants the user only the access specified in the first matching ACE. All subsequent ACEs are ignored.

The first ACE is the only ACE containing an option field (the PROTECTED option). Using this option prevents the first ACE from being deleted unless you have explicitly deleted the ACE with the ACL editor, or you have specified the ACE with the SET ACL/EDIT/DELETE command.

Identifier ACEs for Other Objects

Create identifier ACEs for other system objects, such as devices, as you create ACEs for files or directories. For example, suppose your company has a special letter-quality printer (TTA8) that is used only for printing checks. As a result, the check forms are always loaded in the printer. This device is never to be used for logins, and no queues are directed to it. Only one user, MGREY, is allowed read and write access to it. The system manager can establish this restriction by setting the protection on the printer with the following command:

```
$ SET PROTECTION=(S,O,G,W)/DEVICE TTA8:
```

The following identifier ACE, applied to the object TTA8, restricts access to the device:

```
(IDENTIFIER=MGREY,ACCESS=READ+WRITE)
```

5.2 Default Protection ACE

The default protection ACE is used to ensure that one type of UIC-based protection is propagated throughout a directory tree. This type of ACE allows you to specify protection for one directory structure that is different from the default protection applied to other directories. Default protection ACEs can be applied only to directory files.

Following is the format for a default protection ACE:

```
(DEFAULT_PROTECTION[,options],protection_mask)
```

This type of ACE is specified by the keyword DEFAULT_PROTECTION. The second field (the options field) in a default protection ACE controls whether an ACE is propagated, can be displayed, or can be deleted. This field in a default protection ACE begins with the keyword OPTIONS and takes one or more of the following keywords:

HIDDEN	Indicates that this ACE can only be changed by the application that added it. The ACL editor does not permit modification or deletion. Thus, the ACL editor displays the ACE only to show its relative position within the ACL, not to facilitate editing of the ACE. The DCL DIRECTORY and SHOW ACL commands will not display hidden ACEs.
PROTECTED	Indicates that an ACE is preserved even when an attempt is made to delete the entire ACL. A protected ACE must be specifically deleted with the ACL editor or by specifying the ACE on the command line of the DCL command SET ACL.
NOPROPAGATE	Indicates that, when copying an ACL from one version of a file to a later version of the same file, the ACE is not propagated.
NONE	Indicates that no options apply to an ACE. Although you might enter OPTIONS=NONE when you create the ACE, OPTIONS=NONE is not displayed when the ACE is displayed.

Connect multiple options with plus signs (+). If you specify any other options with the NONE option, the other options will take precedence.

The protection mask is specified the same as for UIC-based protection, with the user categories—SYSTEM, WORLD, GROUP, and OWNER—and the access categories—READ, WRITE, EXECUTE, and DELETE. See the discussion of UIC-based protection in the *VMS DCL Dictionary* for more information.

The following sample ACE, included in an ACL for the directory MALCOLM, sets up default protection so that any files created in the directory allow system and owner groups read, write, execute, and delete access. Group and world groups are denied access.

```
(DEFAULT_PROTECTION,S:RWED,O:RWED)
```

When you add or change the default protection for a directory, there is no effect on the files already created in the directory. All new files will receive the default protection.

If you want to have the default protection ACE PROTECTED, which saves its ACE if an attempt is made to delete the entire ACL, create the following ACL:

```
(DEFAULT_PROTECTION,OPTIONS=PROTECTED,S:RWEDC,O:RWEDC,G,W)
```

5.3 Security Alarm ACE

The security alarm ACE allows you to specify that an alarm message be sent to the security operator's terminal if a certain type of access takes place. (The DCL command SET AUDIT enables the security operator's terminal to receive security alarms.)

The security alarm ACE specifies the type of access that you want to protect. When the specified access is violated, an alarm message is sent to security operators.

Although you can create alarm ACEs in an ACL that cause the system to observe the event and take the required action, you should also coordinate protection with your system's security manager (the person who possesses the SECURITY privilege). The security manager is responsible for enabling the alarm feature. Since this feature uses system resources, the security manager might be reluctant to leave it enabled at all times.

Following is the format of a security alarm ACE:

```
(ALARM_JOURNAL=SECURITY[,options][,access])
```

This type of ACE is specified by the keywords ALARM_JOURNAL=SECURITY. The second field in a security alarm ACE begins with the keyword OPTIONS, which takes one or more of the following keywords:

DEFAULT	This option is valid only for directory files. Indicates that an ACE is to be included in the ACL of any files created within a directory. When the ACE is propagated, the DEFAULT indicator is removed from the ACL of the created file.
HIDDEN	Indicates that this ACE can only be changed by the application that added it. The ACL editor does not permit modification or deletion. Thus, the ACL editor displays the ACE only to show its relative position within the ACL, not to facilitate editing of the ACE. The DCL DIRECTORY and SHOW ACL commands will not display hidden ACEs.
PROTECTED	Indicates that this ACE is preserved even when an attempt is made to delete the entire ACL. A protected ACE must be explicitly deleted with the ACL editor or by specifying the ACE on the command line of the DCL command SET ACL.
NOPROPAGATE	Indicates that, when copying an ACL from one version of a file to a later version of the same file, the ACE is not propagated.
NONE	Indicates that no options apply to this ACE. Although you enter OPTIONS=NONE when you create the ACE, OPTIONS=NONE is not displayed when the ACE is displayed.

Connect multiple options with plus signs (+). If you specify any other options when specifying NONE, the other options take precedence.

The third field in an alarm ACE controls the type of access that causes the alarm to be sent.

Specify any of the following access actions with the ACCESS keyword:

READ	Generates an alarm if an accessor attempts to read the object.
WRITE	Generates an alarm if an accessor attempts to read or write the object.
EXECUTE	Generates an alarm if an accessor attempts to execute the object.
DELETE	Generates an alarm if an accessor attempts to delete the object.
CONTROL	Generates an alarm if an accessor attempts to perform control operations on the object, such as changing the protection on the object.
SUCCESS	Generates an alarm for each successful attempt by an accessor to access the object with the access specified from the preceding set.
FAILURE	Generates an alarm for each unsuccessful attempt by an accessor to access the object with the access specified from the preceding set.

Note: For an alarm to have any effect, you must include SUCCESS or FAILURE or both.

6 Customizing the ACL Editor

Because the current version of the ACL editor was written using the VAX Text Processing Utility (VAXTPU), you can modify the ACL editor to meet your own needs. You can modify and recompile the ACL section file SYS$LIBRARY:ACLEDIT.TPU (the source file used to create the compiled ACL section file SYS$LIBRARY:ACLEDT$SECTION.TPU$SECTION) or create your own ACL section file.

See the *VAX Text Processing Utility Manual* for more information on writing and processing section files.

6.1 Modifying Variables in the ACL Section File

Following is a list of the variables and their defaults available through the ACL section file:

Variable	Meaning
ACLEDIT$X_CHECK_DUPLICATES	Controls whether or not a check for duplicate ACEs is made. This variable can take the following values: 0 No duplicate ACE check is made. 1 A duplicate ACE check is made. If the ACE to be entered matches an existing ACE, an error message is returned. This is the default.

Variable	Meaning
ACLEDIT$X_CHECK_MODIFY	Allows or disallows modification of ACEs. This variable can take the following values: 0 The ACE can be modified. 1 The ACE cannot be modified. If an attempt is made to modify the ACE, it is replaced with the original ACE. This is the default.
ACLEDIT$X_DIRECTORY_FILE	Indicates whether or not the object is a directory file. This variable can take the following values: 0 The object is not a directory file. 1 The object is a directory file.
ACLEDIT$X_PASTE_BUFFER	Controls whether or not PASTE buffer support is enabled for VT200 series terminals. This variable can take the following values: 0 PASTE buffer support is disabled. This is the default. 1 PASTE buffer support is enabled.
ACLEDIT$X_PROMPT	Controls whether or not automatic text insertion (prompt mode) is enabled. This variable can take the following values: 0 Prompt mode is disabled. 1 Prompt mode is enabled. This is the default.
ACLEDIT$X_USE_DEFAULT_OPT	Controls whether or not the DEFAULT option can be used with nondirectory ACEs. This variable can take the following values: 0 The DEFAULT option can only be used with ACEs of directory (.DIR) files. This is the default. 1 The DEFAULT option is available for use with ACEs of all object types.
ACLEDIT$C_WINDOW_SHIFT	Specifies the number of columns to shift the edit window in the desired direction, GOLD ← for a left shift and GOLD → for a right shift. The default is 8 columns.

If you modify any of the preceding variables listed or change any other part of the ACL section file, you must recompile the section file with the following command:

```
$ EDIT/TPU/NOSECTION/COMMAND=SYS$LIBRARY:ACLEDIT
```

Use the previous command if you make changes directly to the source code file (SYS$LIBRARY:ACLEDIT) that creates the compiled ACL section file SYS$LIBRARY:ACLEDT$SECTION. If you are adding a private command file to the existing ACL section file, recompile the section file using the following command:

```
$ EDIT/TPU/SECTION=SYS$LIBRARY:ACLEDT$SECTION/COMMAND=your_section.TPU
```

The resulting compiled VAXTPU ACL section file is placed in your current directory. To use the new section file, you must do one of the following:

- Move the resulting compiled section file, ACLEDT$SECTION.TPU$SECTION, to the SYS$LIBRARY directory. This changes the default ACL editor section file for all users.
- Keep the compiled section file in your directory and define the logical name ACLEDT$SECTION in your LOGIN.COM file to point to the file using the following command:

```
$ DEFINE ACLEDT$SECTION yourdisk:[yourdir]ACLEDT$SECTION
```

Note that the default file type for the section file before compiling (the source file) is TPU and the default file type for the compiled section file is TPU$SECTION. See the *VAX Text Processing Utility Manual* for more information on writing and processing a VAXTPU section file.

6.2 Using the ACL Editor CALL_USER Routine

The ACL editor CALL_USER routine is part of the shareable image SYS$LIBRARY:ACLEDTSHR.EXE. You can incorporate the ACL editor CALL_USER routine with its existing function codes into your own ACL section file, or you can write your own CALL_USER routine that recognizes a different set of function codes.

The ACL editor CALL_USER routine recognizes only those functions used by the ACL editor VAXTPU section file. All other function codes are passed to a user-supplied CALL_USER routine; if the high-order word of the CALL_USER function code (bits <16:31>) contains the ACL editor facility code, it is handled by the ACL editor CALL_USER routine. Otherwise, an attempt is made to locate a user-supplied CALL_USER routine. Refer to the description of the CALL_USER routine in the *VAX Text Processing Utility Manual* for more information on creating your own CALL_USER routine.

Following is a description of the CALL_USER routine function codes supported by the ACL editor:

Function Code	Mnemonic	Description
18153473	ACLEDIT$C_PARSE_ACE	Parses the input string (ACE) and returns the parsed (binary) ACE if no errors are found. Otherwise, the returned string contains a zero as the first two characters, and the unparsed portion of the input ACE as the remainder of the string.
18153474	ACLEDIT$C_CHECK_MODIFY	Returns the string "READ_WRITE" if the ACE can be modified by the user. Otherwise, returns the string, "READ_ONLY."
18153475	ACLEDIT$C_PROMPT_MODE	Returns the string "PROMPT_MODE" if the prompt mode option was specified. Otherwise, returns the string "NOPROMPT_MODE."

Function Code	Mnemonic	Description
18153476	ACLEDIT$C_CHECK_ACE	Parses the input string (ACE) and returns the parsed (binary) ACE if no errors are found. Otherwise, the ACE text is highlighted in reverse video and a VAXTPU variable of the form ACLEDIT$X_RANGE_x is created to identify the ACE in error. (The "x" is a sequential number starting with 1.)
18153477	ACLEDIT$C_CHECK_DIR	Returns the string "DIRECTORY_FILE" if the object being edited is a directory file. Otherwise, returns the string "NODIRECTORY_FILE."
18153478	ACLEDIT$C_SET_CANDIDATE	Parses the input string (ACE) and returns the string "PARSE_OK" if no error was encountered. Otherwise, returns the string "PARSE_ERROR." If the parse was successful, a check is made for duplicate ACEs using the CALL_USER function ACLEDIT$C_CHECK_DUP.
18153479	ACLEDIT$C_CHECK_DUP	Parses the input string (ACE) and returns the string "PARSE_ERROR" if an error was encountered. Otherwise, the parsed (binary) ACE is compared with the candidate ACE set by the CALL_USER function ACLEDIT$C_SET_CANDIDATE. Returns the string "DUPLICATE_ACE" if the ACE is a duplicate or "UNIQUE_ACE" if it is not a duplicate.
18153482	ACLEDIT$C_MESSAGE	Assumes the input string is a VMS error code and returns in the ACL editor message window the message text associated with the error code.

ACL Editor Usage Summary

The VMS Access Control List (ACL) Editor creates or modifies an access control list for a specified object.

FORMAT

EDIT/ACL *object-spec*

COMMAND PARAMETER

object-spec

Specifies the object whose access control list is to be created or edited using the ACL editor. If an access control list does not exist, it is created. The object specified can be a file, directory, device, global section, logical name table, or batch or print queue.

If the object is a file, the ACL editor does not provide a default file type. If you omit the file type, it is presumed to be null. The specified file must be a disk file on a Files-11 On-Disk Structure Level 2 formatted volume. If the object is a directory, specify a file specification with the file type of DIR. If the object type is anything other than a file or a directory, you must specify /OBJECT=type as the qualifer.

No wildcard characters are allowed in the object specification.

usage summary

Invoke the ACL editor with the DCL command EDIT/ACL. The /ACL qualifier is required. Exit from the editing session by pressing CTRL/Z. If the session ends abnormally and journaling has been in effect, you can recover the ACL with the /RECOVER qualifier. Quit the editing session without saving the edits by pressing GOLD CTRL/Z.

By default, the editing session occurs at the SYS$OUTPUT device. As the result of a successful editing session, an access control list is created for an object.

You can invoke the ACL editor to create or modify the ACL only for an object that you own, have control access to, or can gain access to by a privilege such as BYPASS, GRPPRV, READALL, or SYSPRV.

You can create an access control list only for a file or directory that currently exists and is on a Files-11 On-Disk Structure Level 2 disk.

ACL EDITOR QUALIFIERS

The EDIT/ACL command qualifiers provide you with control over journaling and recovering the editing session.

/JOURNAL

Controls whether a journal file is created for the editing session.

FORMAT

/JOURNAL *[=file-spec]*
/NOJOURNAL

DESCRIPTION

By default, the ACL editor keeps a journal file containing a copy of modifications made during an editing session. If you specify /NOJOURNAL, no journal file is generated. If you omit the qualifier or specify /JOURNAL, a journal file is created. If you omit the journal file specification, by default the journal file is given the name of the input file specification with a JOU extension.

No wildcard characters are allowed in the journal file specification.

If your editing session ends abnormally, you can invoke the ACL editor again and recover the changes made during the aborted session by specifying the /RECOVER qualifier and the name of the journal file if it differs from the default name.

EXAMPLE

```
$ EDIT/ACL/JOURNAL=COMMONACL.SAV MECH1117.DAT
```

With this command, you create a journal file named COMMONACL.SAV that contains a copy of the ACL and the editing commands used when creating the ACL for the file MECH1117.DAT. The file COMMONACL.SAV can then be used for recovery of an interrupted editing session for MECH1117.DAT's ACL. A journal file is created by default, but the qualifier is specified to include the name COMMONACL.SAV, a more descriptive name than the default journal file, MECH1117.JOU.

/MODE

Specifies the use of prompting during the editing session.

FORMAT

/MODE=*option*

DESCRIPTION

By default, the ACL editor selects prompt mode. Use the /MODE qualifier to specify one of the following prompt options:

PROMPT	Specifies that, where possible, the selected field within the ACE is initially filled with the first of a list of items that might apply to the field.
NOPROMPT	Specifies that no prompting is used by the ACL editor.

EXAMPLE

```
$ EDIT/ACL/MODE=NOPROMPT WEATHERTBL.DAT
```

With this command, you initiate an ACL editing session to create an ACL for the file WEATHERTBL.DAT. The /MODE=NOPROMPT qualifier specifies that no assistance is required in entering the ACL entries.

/OBJECT

Specifies the type of object whose ACL is being edited. The /OBJECT qualifier is required unless the object whose ACL is being edited is a file or a directory file.

FORMAT

/OBJECT=*type*

DESCRIPTION

The /OBJECT qualifier is required unless the object whose ACL is being edited is a file. The following keywords may be specified with /OBJECT:

FILE	Specifies that the object type is a file or a directory file.
DEVICE	Specifies that the object type is a device.
SYSTEM_GLOBAL_SECTION	Specifies that the object type is a system global section.
GROUP_GLOBAL_SECTION	Specifies that the object type is a group global section.
QUEUE	Specifies that the object type is a batch or device (printer, server, or terminal) queue.
LOGICAL_NAME_TABLE	Specifies that the object type is a logical name table.

EXAMPLES

1
```
$ EDIT/ACL/OBJECT=DEVICE WORK1
```

The /OBJECT qualifier is used in this command because it is a device ACL being edited.

2
```
$ EDIT/ACL/OBJECT=QUEUE FAST_BATCH
```

The command in this example creates an ACL for the FAST_BATCH queue. Note that if you create an ACL for a generic queue, you should create identical ACLs for all execution queues to which jobs can be directed.

/RECOVER

Determines whether or not the ACL editor restores the object's ACL from a journal file at the start of an editing session.

FORMAT

/RECOVER *[=file-spec]*
/NORECOVER

DESCRIPTION

The /RECOVER qualifier specifies that the ACL editor should restore the ACL from the journal file specified by input-file-spec.JOU. This operation restores the ACL to the state it was in when the last ACL editing session ended abnormally.

If the journal file has a file name other than input-file-spec.JOU, specify it with the /RECOVER qualifier.

EXAMPLE

```
$ EDIT/ACL/JOURNAL=SAVEACL MYFILE.DAT
    .
    .
    .
User creates ACL until system crashes
    .
    .
    .
$ EDIT/ACL/JOURNAL=SAVEACL/RECOVER=SAVEACL MYFILE.DAT
    .
    .
    .
ACL is restored and user proceeds with editing until done
    .
    .
    .
   ^Z
$
```

Initiate the ACL editing session using the /JOURNAL qualifier, which specifies that the journal file SAVEACL will be saved if the session ends abnormally. The session proceeds until aborted by a system crash. To recover from the aborted editing session and continue with additional edits, enter the second EDIT/ACL command. This session begins by restoring the session with the journal file SAVEACL. When you exit the editing session by pressing CTRL/Z, the journal file SAVEACL.JOU is deleted.

Index

A

Access control list
  See ACL
Access control list entry
  See ACE
ACE (access control list entry)
  default protection • ACL–17
  format • ACL–13
  identifier • ACL–13
  security alarm • ACL–18
ACL (access control list) editor • ACL–23
  customizing • ACL–19
  ACL qualifiers • ACL–24 to ACL–28
  exiting • ACL–12
  invoking • ACL–2
  keypad editing • ACL–3
  quitting • ACL–12
  recovering • ACL–12
ACL section file • ACL–19

E

EDIT/ACL command • ACL–23
Editing session
  keypad editing • ACL–3

H

Help facility • ACL–3

I

Identifier ACE • ACL–13
  example • ACL–15, ACL–16
  specifying • ACL–13
  specifying access • ACL–15
  specifying options • ACL–14

J

/JOURNAL qualifier • ACL–25

K

Keypad editing • ACL–3
  control key editing commands • ACL–11
  editing commands • ACL–3
  VT200-specific editing commands • ACL–10

M

/MODE qualifier • ACL–26

O

/OBJECT qualifier • ACL–27

P

PASTE buffer • ACL–11

R

/RECOVER qualifier • ACL–28

S

Security alarm ACE • ACL–18
  specifying access • ACL–19
  specifying options • ACL–18

V

VT200-specific editing commands • ACL–10

Reader's Comments

VMS Access Control List
Editor Manual
AA–LA41A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less ______________________________

____

____

What I like best about this manual is ______________________________

____

____

What I like least about this manual is ______________________________

____

____

I found the following errors in this manual:

Page	Description

Additional comments or suggestions to improve this manual:

____

____

____

____

I am using **Version** ______ of the software this manual describes.

Name/Title ______________________________ Dept. ______________

Company ______________________________ Date ______________

Mailing Address ______________________________

______________________________ Phone ______________

— — Do Not Tear - Fold Here and Tape —

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

— — Do Not Tear - Fold Here —

Cut Along Dotted Line

Reader's Comments

VMS Access Control List
Editor Manual
AA–LA41A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less ______________________________

____

____

What I like best about this manual is ______________________________

____

____

What I like least about this manual is ______________________________

____

____

I found the following errors in this manual:

Page	Description
______	______________________________
______	______________________________
______	______________________________
______	______________________________
______	______________________________

Additional comments or suggestions to improve this manual:

____

____

____

____

I am using **Version** ______ of the software this manual describes.

Name/Title ______________________________ Dept. ______________

Company ______________________________ Date ______________

Mailing Address ______________________________

______________________________ Phone ______________

--- Do Not Tear - Fold Here and Tape ---

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

--- Do Not Tear - Fold Here ---

Cut Along Dotted Line