CA VM:Director™ for z/VM

Reference Guide

Release 3.1, Second Edition

7/21/2014

CA Technologies Product References

This document references the following CA Technologies products:

- CA Mainframe VM Product Manager
- CA VM:Account™
- CA VM:Archiver™ for z/VM
- CA VM:Backup™ for z/VM
- CA VM:Director™ for z/VM
- CA VM:Schedule™
- CA VM:Tape™ for z/VM

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 3: Configuration File Reference 351

Chapter 1: Introduction

CA VM:Director is an efficient and comprehensive directory and disk space management system for z/VM. It significantly reduces the time required to manage the resources defined in the z/VM directory and prevents costly errors in DASD mapping. With CA VM:Director, you can delegate aspects of directory management to your users without jeopardizing system control.

CA VM:Director manages the z/VM directory by distributing the burden of user ID and DASD management from a single system administrator to several directory managers. They also allow users to make some directory changes for their own virtual machines.

This section contains the following topics:

About This Document

This document is an alphabetic reference of all CA VM:Director configuration file records, commands, user exits, and utilities.

Note: For more information about using CA VM:Director, see the *Administration Guide* or *Directory Manager Guide*. These books describe how to perform common tasks using the CA VM:Director screens and line–mode commands.

This document assumes you are familiar with the following:

- z/VM, CP, CMS and the Shared File System (SFS) standards, conventions, and operation
- The daily requirements of your VM environment
- CA VM:Director functions
- Directory management and VM security concepts

Format Conventions for Code Syntax

This section describes the format and conventions used to document commands, utilities, and user exits. Each convention provides examples, describing how to use commands, how to use options, or how the system responds to user entries.

Note: The examples and instructions throughout this document use VMANAGER as the user ID for system administrator and maintenance functions. Also, this document uses VMDIRECT as the user ID for the service virtual machine (svm). These user IDs are the default values. If you use non-default user IDs for the system administrator or the svm, replace the default values throughout this document with the values that you use.

Command Abbreviations

When a command contains uppercase and lowercase letters, then the uppercase letters denote the shortest acceptable abbreviation that you can use to type the command. However, when a code item appears entirely in uppercase letters, you cannot abbreviate the item.

You can type the code item in uppercase letters, lowercase letters, or any combination.

Example:

```
CMDNAme
```

In this example, you can enter CMDNA, CMDNAM, or CMDNAME in any combination of uppercase and lowercase letters.

Continuation

The code syntax or code fragment definitions can continue from one line to the next line. The following examples describe code continuation:

Example 1:

```
A | B C | D
```

This code is equivalent to the following code:

```
A
| B C
| D
```

Example 2:

```
{choice1 | choice2 | choice99}
```

This code is equivalent to the following code:

```
 { choice1
 | choice2
 | choice99 }
```

Default Values

An underlined code item denotes the default value. The system uses the default value unless you override it. You can override the default value by coding an option from the available list.

Example:

[<u>parm1</u> | *parm2* | *parm3*]

In this example, the code item *parm1* is the default value, and this is used by the system when you do not specify any of the options. However, you can code *parm1*, *parm2*, or *parm3.*

Keywords and Constants

A keyword name or constant always appears in uppercase letters. Code the keyword name or constant exactly as shown in the following example:

Example:

```
STOP {tracenumber | * | [USER] userid}
```

This example displays the USER keyword.

Optional Choices

Defines optional code items—denoted by square brackets around a code item.

Example:

```
CMDName [parm1]
```

In this example, you can choose *parm1* or no parameter at all. However, when two or more items are enclosed in square brackets and separated by vertical bar characters, all of them are optional.

Multiple Optional Choices

When two or more items are enclosed in square brackets and separated by vertical bar characters, all of them are optional.

Example:

In this example, you can choose *parm1, parm2, parm3,* or nothing at all.

```
[parm1 | parm2 | parm3]
```

Positional Parameters

Commands with positional parameters are identified by nested square brackets. Each positional parameter requires the specification of all previous positional parameters. The following example describes the positional parameter:

Example:

```
CMDName [posparm1 [posparm2 [posparm3]]]
```

In this example, *posparm3* can be specified only when *posparm1* and *posparm2* are also specified.

Repeatable Choices

A list of code items enclosed in square brackets and followed by an ellipsis means that you can select more than one item or, in some cases, repeat a single item.

Example 1:

```
[value1 | value2 | value3]...
```

In this example, you can choose a single value, more than one value, or none of the values.

Repetition

An ellipsis following a code item means that the code item can be repeated.

Example 2:

```
Repeat…
```

Example 3:

```
{Operand1 | Operand2}...
```

In this example, you must choose at least one operand and you can choose both operands. If one of the operands is a variable, you may also be able to repeat that operand with different values.

Required Choices

You must select one item from a list of items when they are enclosed in curly braces. The items are separated by a vertical bar character.

Example:

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

In this example, your choice results in CMDNAME A, CMDNAME B, or CMDNAME C.

Special Symbols

The following list describes the meaning of the special symbols used in codes:

- {} (encloses a list of operands, one of which is required).
- [] (enclose an optional operand or operands).
- " " (enclose the name of a syntax fragment)
- underline (identifies a default value)
- | (separates alternative operands)
- . . . (Indicates that the preceding item or group can be repeated).

Symbols

The following list displays symbols. These symbols should be coded exactly as they appear in the code syntax.

- * (Asterisk)
- : (Colon)
- , (Comma)
- = (Equal Sign)
- — (Hyphen)
- ( )(Parentheses)
- . (Period)

Syntax Fragments

Some codes use fragments, when the code syntax is too lengthy. The fragment name appears between double quotes in the code syntax.

The expanded fragment appears in the syntax after all other parameters or at the bottom of the code syntax. A heading with the fragment name identifies the expanded fragment.

Example1:

```
CMDName "Parms"

Parms:

[ParmA] [ParmB] [ParmC]
```

In this example, the fragment is named "Parms", and the expanded fragment appears at the bottom of the code syntax.

Example 2:

```
CMDName [( "Options"]


Options:

  OptionA
| OptionB
| OptionC
| OptionD [OptionD1]
```

In this example, the fragment is named Options. The list of options is too long to fit on one line, so they are listed vertically. Note that the outermost brackets and the "(" symbol are in the main line, not in the vertical list. Also note that one option has its own optional parameter.

System Response

Uppercase characters represent system responses or prompts.

Example:

```
ENTER YOUR LOGON PASSWORD:
```

This example displays a system response.

User–Entered Commands or Records

User-entered commands are shown in bolded lowercase letters even though you can enter commands in either upper or lower case.

Example:

```
vmsecure addentry writers tcom (noformat nowait
```

This example shows what a user-entered command looks like.

In this example "writers tcom" is the file name and file type of the directory entry you are adding.

However, if the entry is a record, it appears in uppercase letters.

Example:

```
ACCESS DRCT 1B0 U
```

This example shows a configuration record that is entered by a user.

Variables

Lowercase items in italics denote variables.

Example:

CMDNAme *varname*

In this example, *varname* represents a variable that you must specify when you code the command CMDNAME.

Moving Through Screens and Menus

The following commands bring up either screens in which you fill in fields or menus from which you select other options:

- CONFIG
- MANAGE
- ULIST
- USER

All four commands require that you enter your logon password before you can use them. CA VM:Director prompts you for your password after you enter the command and before CA VM:Director displays its menus or entry screens. The logon password is required as another level of access control; it ensures that unauthorized people cannot use administration commands if they happen to gain access to your terminal.

To use these commands without having to enter your logon password, you must have NOPASS authorization in the AUTHORIZ CONFIG file.

Note:

- For more information about the NOPASS authorization, see Waiving Password Requirements for Commands in the *Administration Guide*.
- For more information about these commands and how they work, see Command Reference (see page 31).

Help With Screens and Menus

You can get information about any currently displayed menu or screen by pressing PF1 (Help). This program function key displays a help screen that describes what you can do from the menu or screen.

When you are finished reading the help screen, press PF3 (End) to return to the original screen.

CA VM:Director Commands

You can enter CA VM:Director commands from two places:

- CMS, either directly or through CP SMSG
- CA VM:Director server console

The format in which you must enter a CA VM:Director command depends on where you enter it.

  - From CMS directly, you must precede CA VM:Director commands with the name of the service virtual machine:

    `VMDIRECT` *`command`* `[`*`parameters`*`]`

  All CA VM:Director commands can be entered from CMS.

  - Through CP SMSG, you must precede CA VM:Director commands with #CP SMSG and the name of the CA VM:Director service virtual machine:

    `#cp smsg vmdirect` *`command`* `[`*`parameters`*`]`

- Only some commands can be entered through CP SMSG, and you must have NOPASS authorization in the AUTHORIZ CONFIG file to issue these commands through CP SMSG. These commands are:
  - ABEND
  - CMS

- END
- EXPIRE
- JOURNAL
- MAP
- NOLOG
- QPCB
- QUERY
- RESET
- TRACE

■ From the CA VM:Director console, do not precede CA VM:Director commands with the name of the CA VM:Director service virtual machine:

  command [*parameters*]

  Only some CA VM:Director commands can be entered at the CA VM:Director console, and only when the CA VM:Director console is running connected. These commands are:

  - ABEND
  - CMS
  - END
  - JOURNAL
  - MAP
  - QLOCK
  - QPCB
  - RESET

Important! Do not enter other commands from the CA VM:Director service virtual machine console. Further, never issue commands to other products from the CA VM:Director console. CA VM:Director operation will cease while that command executes, and this could cause your entire system to hang.

Help with CA VM:Director Commands

You can get help for selected CA VM:Director commands through the CMS HELP facility. To see an overview of CA VM:Director help and a list of commands for which help is available, enter the following from CMS:

```
help vmdirect
```

To see the help screen for one of the commands listed in the overview, move the cursor to that command and press PF1. When you are finished reading the help screen, press PF3 to return to the help overview screen and PF3 again to exit the CMS HELP facility.

You can bypass the help overview screen and get help for a command directly by entering **help vmdirect** and the name of the command for which you need help. For example, to see the help screen on the USER command, enter the following from CMS:

```
help vmdirect user
```

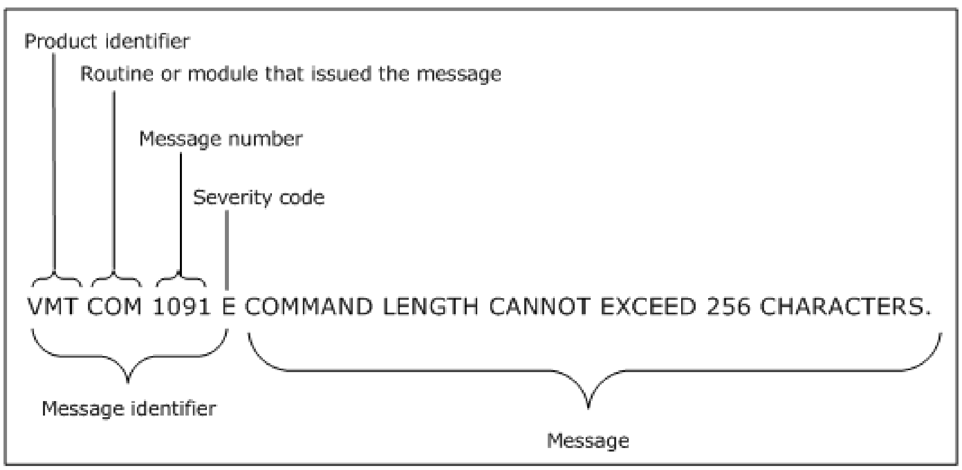
Note: For more information about using the CMS HELP facility, see the IBM *CMS User's Guide* appropriate for your system.

CA VM:Director Messages

While using CA VM:Director, you receive messages that do one of the following:

- Inform you of the status and progress of the function or task
- Prompt you for information about your system
- Indicate if an error occurred
- Provide instructions for resolving errors

All messages are in this format:



You can tell the type of message by the last letter of the identifier, which is its *severity code*. The severity code can be one of the following:

A

Action message.

You need to perform an action, for example, move the cursor or enter information.

E

Error message.

The function or task can continue normally if you correct the error, but it may end.

I

Informational message.

CA VM:Director has completed initialization or execution of a task. When an informational message appears immediately after a severe error message (severity code S), it tells you how to correct the problem the severe error message identified.

R

Response message.

Requests information. You need to respond to a CA VM:Director message.

S

Severe error message.

An error preventing the function or task from continuing has occurred. You are returned to CMS and must restart the procedure after correcting the error.

T

Terminal error message.

A serious internal error has occurred, preventing the CA VM:Director service virtual machine from continuing execution.

W

Warning message.

An abnormal condition has occurred. The function or task continues although the results may be affected by the abnormal condition.

Help with CA VM:Director Messages

Note: For more information or help for a CA VM:Director message you receive, see the *Message Reference Guide* or use the CMS HELP facility for that message.

Message Reference Guide lists all messages CA VM:Director can generate, explains the possible causes of each message, and tells you how to respond to that message.

To get help for a CA VM:Director message, from CMS, enter **help**, followed by the three–character product identifier, the message number, and its severity code.

For example, to get help for the message VMXCOM1091E*,* enter the following from CMS:

`help vmx1091e`

When you are finished reading the message help, press PF3 to exit the CMS HELP facility.

Note: For more information about using the CMS help facility, see the IBM *CMS User's Guide* appropriate for your system.

Changes in CA VM:Director

For a complete listing of all changes in CA VM:Director, see the accompanying *Release Notes.*

Chapter 2: Command Reference

This chapter presents overview information for CA VM:Director commands, then provides the syntax and explains each command in detail. The commands are organized alphabetically.

Command Introduction

The table below lists and briefly describes the CA VM:Director commands. The authorizations chapter in the *Administration Guide* lists the authorizations users must have in the AUTHORIZ CONFIG file to use these commands for the purposes listed in that table. (Commands followed by an asterisk are part of the Application Programming Interface.)

Command	Function
ABEND	Terminates CA VM:Director operation abnormally
ADDENTRY*	Creates a user ID or directory profile
ADDMDISK*	Adds a minidisk for a user ID
ADMIN	Creates or opens for edit a directory pool definition, a directory profile, or a skeleton file, or opens the VMDIRECT GLOBALS file, the VMDIRECT MANAGERS file, the VMDIRECT POSIX file, or the SFS Managers Configuration Menu
ASSIGN	Assigns a user ID to a different manager
AUDITEXT	Extracts current audit information
CHANGE	Renames a user ID
CHGMDISK*	Moves or changes a minidisk
CHGVOLNM	Changes all references to the volser of any DASD volume controlled by CA VM:Director
CLASS	Assigns a CP privilege class
CMD	Routes another command to an Agent product server in a Single System Image environment
CMS	Executes a CMS or CP command on the CA VM:Director service virtual machine
COMPRESS	Defragments disk storage

Command	Function
CONFIG	Opens the configuration files for editing or updates the SFS configuration
CPFMTXA	Allows authorized users to run the CP DASD FORMAT/ALLOCATE utilities CPFMTXA/ICKDSF
DELENTRY*	Deletes an existing user ID or directory profile
DELETE*	Deletes file spaces for an active user ID
DELMDISK*	Deletes a user ID's minidisk
DISPLINK	Displays links to a user ID's minidisks
DUPENTRY	Creates a directory entry for a new user ID, based on the directory entry for an existing user ID
DUPMDISK*	Creates an exact duplicate of an existing minidisk
EDIT	Opens a user directory entry for editing
EDX	Opens a user directory entry for editing, expanding any INCLUDE statement in that directory entry
END	Terminates CA VM:Director operation normally
ENROLL*	Enrolls a user ID in an SFS file pool
ENTRY	Updates or queries an attribute or statement in a directory entry. Subcommands specify the type of attribute or statement that is affected.
EXPIRE	Expires a user ID's logon password
EXTRACT	Extracts directory information
GENINCL	Adds an INCLUDE statement to a user ID's directory entry
GETENTRY*	Retrieves a current copy of a user ID's directory entry or a directory profile
GETPWEXP	Displays user ID password expiration information
GRANT AUTHORITY	Authorizes users to gain access to SFS directories or the files in them
JOURNAL	Displays password violations or resets a password violation count to zero
LISTAUTH	Lists all authorizations in the AUTHORIZ CONFIG file that pertain to an individual user ID or authority
LOCAL (prefix)	Directs that the CA VM:Director product server on the local node execute the command
LOCK*	Prevents updates to a CMS file, user ID, or directory profile

Command	Function
MACLOAD	Copies a macro from your A-disk to the CA VM:Director service virtual machine's A-disk
MAINT	Performs line-mode USER command and MANAGE command functions
MANAGE	Displays screens that let you define new user IDs and modify existing ones
MAP	Maps a volume
MAY	Queries an authorization in the AUTHORIZ CONFIG file
MDSKSCAN	Scans a user ID's minidisks
MODIFY*	Modifies the SFS allocation limits for a user ID
MOVE2SFS	Copies data from minidisks to SFS
MULTIPLE	Performs user ID management functions on many user IDs at the same time
NEWIPL	Changes an IPL system name or device in all directory entries to a new IPL system name or device
NOLOG	Changes a user ID's password to NOLOG
OVERRIDE	Alters privilege classes without shutting down CA VM:Director
PAINT	Changes a CA VM:Director screen
PASSWORD	Sets a password for a user ID
QLOCK	Displays all CA VM:Director locks
QPCB	Lists active CA VM:Director processes
QSTART	Displays the time CA VM:Director was most recently started
QUERY	Provides information about a number of CA VM:Director functions
REBUILD	Condenses and defragments the CP object directory. Use this command only under the direction of Technical Support
RECLAIM	Reclaims DASD space from MOVERO minidisks
REPENTRY*	Replaces a directory entry or a directory profile
RESET	Resets password violation counts for user IDs
REVOKE AUTHORITY	Revokes access to SFS directories and the files in them from users

Command	Function
SCAN*	Searches and returns information about records in the VM source directory.
TAG*	Adds or manipulates user or installation specific named directory comment records.
TAKEOVER	Recovers from an outage of the member node running the Master product server virtual machine in a Single System Image environment.
TRACE	Traces execution of a CA VM:Director macro
TRANSFER	Transfers a minidisk from one user ID to another
ULIST	Displays information about user IDs
UNLOCK*	Removes a lock from a CMS file, profile, or user ID
USER	Lets users modify values in their own directory entry

Logon Password Prompts

All CA VM:Director commands, except for the following, prompt for a logon password before you can use them:

- DISPLINK
- EXTRACT
- PAINT
- QLOCK
- QPCB
- QSTART
- QUERY

This is a security feature to protect unauthorized use of powerful commands through your user ID. However, if your terminal is secure and there is no reason to restrict your user ID's activity, you can give yourself NOPASS authorization in the AUTHORIZ CONFIG file to all the other commands. You can give authorization to bypass this password requirement when appropriate—for example, when users need to run EXECs that include these commands.

Positional Parameters

Many parameters for CA VM:Director commands are positional. For commands where this is so, you must specify all parameters that precede a positional parameter if you want to specify that positional parameter.

Many positional parameters have default values, which are indicated by asterisks in command syntax boxes. You can enter an asterisk for these positional parameters to use the parameter's default value.

For commands that support pattern matching by use of an asterisk, this manual does not display the asterisk in the command syntax box, but explains in the description of the relevant parameter or option that it supports pattern matching through use of the asterisk.

Application Programming Interface Commands

Some of the CA VM:Director commands are also application programming interface (API) commands. API commands simplify application and EXEC creation that use CA VM:Director features. Instead of prompting for responses, these commands let you provide all the information about the command itself.

The following commands can be used as API commands in application programs or EXECs:

- ADDENTRY
- ADDMDISK
- CHGMDISK
- DELENTRY
- DELETE
- DELMDISK
- DUPMDISK
- ENROLL
- GETENTRY
- LOCK
- MODIFY
- REPENTRY
- SCAN
- TAG
- UNLOCK

Because you can use these commands directly (interactively) or add them to executable functions, their results, as described in this chapter, vary predictably. This is true particularly of results based on default or assumed values, because the default or assumed values are always those of the user ID that submits the command for processing. For example, if you direct the output of a command to the virtual reader when you enter a command directly at your terminal, the output is sent to your own virtual reader; however, if you direct the output of a command to the virtual reader when you put this command in a program, the output is sent to the virtual reader of the user ID that submitted the program for execution.

Common API Command Options

Many of the CA VM:Director API commands support similar processing options, which are described below. If one of these options is used differently for a particular command, then its difference is explained in the detailed description of that command.

Routing Messages

CA VM:Director API commands return messages that can be trapped. The default destination for messages is the terminal of the user ID that issued the API command (if the command is in a program, this is the user ID that ran the program), but you can use the following options to redirect messages to multiple destinations. The calling application can parse messages in a file or program stack and use their information for any purpose.

FILE [*filename*]

Places all informational and error messages generated by the API command in a CMS file on the issuing user's A-disk.

By default, the name of this CMS file is the name of the command that generated the messages (for example, ADDENTRY). The filetype is always RESULTS.

To store the messages in a different file, specify that filename. If you do not specify a filename, the FILE option must be the last one on the command line.

If a file with the name *filename* RESULTS A does not exist on the issuing user ID A-disk, CA VM:Director creates it. If the file does exist, CA VM:Director appends new messages to its end.

You cannot use the FILE option with the NOWAIT option.

[<u>MSG</u> | NOMSG]

Displays (MSG) or does not display (NOMSG) informational and error messages generated by the API command at the issuing user's terminal.

[STACK] FIFO | LIFO]

Places all informational and error messages in the issuing user's program stack: in first-in, first-out (FIFO) order or last-in, first-out (LIFO) order.

The STACK option cannot be used with the NOWAIT option.

[STEM *stemname*]

The named REXX stem variable does one of the following:

- Supplies input parameters to the command
- Returns command responses or query results from command execution

Commands with a STEM option must be issued from a REXX program.

Running Commands in Disconnected Mode

Some API commands format information or write information to minidisks. These processes can take some time to complete. You can write your program or EXEC to either wait for the API command to finish before other work can be done from the issuing terminal, or to complete the work in a servant, allowing the issuing user ID to continue with other work.

Because a process running on a servant has no communication path to the issuing user ID's virtual machine, CA VM:Director sends all messages generated by the process to the servant. When an API command in the servant completes, CA VM:Director sends the console file of the servant's activity to the issuing user's reader queue.

One option is available on most API commands that can run the process on a servant:

[Wait | NOWait]

Waits (WAIT) for the API command to finish processing before allowing the issuing user ID to continue or does not wait (NOWAIT) but runs the process on a servant so the issuing user ID can continue.

The Servant Facility must be active for the NOWAIT option to work.

The WAIT | NOWAIT option on an API command is ignored if you specify the NOFORMAT option on that API command, since NOFORMAT implies that there is no time-consuming command processing. The option is also ignored if you create a directory profile or a minidisk on T-DISK or V-DISK, since no time-consuming activity involving a minidisk actually occurs.

The NOWAIT option cannot be used with the FILE or STACK option.

Directory Entry Types

Many commands operate on the directory entry files on the Directory Database minidisk. These files contain collections of directory statements that begin with a statement which defines the type of entry in the file. There are four types of directory entries:

- USER
- PROFILE
- IDENTITY
- SUBCONFIG

Each entry type is allowed to contain statements appropriate to that type of entry. That fact imposes a requirement that not all commands operate with all entry types. Each command which operates on a directory entry will specify the type of entries it supports.

CMS Block Size of Formatted Minidisks

The following table lists the values allowed for the CMS block size and the capacity of each cylinder in blocks and K-bytes for various types of DASD. You can refer to this table whenever you are allocating minidisks on a CKD DASD.

Device Type	**Block Size**							
	512		**1024**		**2048**		**4096**	
	Blocks	**Kbytes**	**Blocks**	**Kbytes**	**Blocks**	**Kbytes**	**Blocks**	**Kbytes**
3380	690	345K	465	465K	270	540K	150	600K
3390	540	270K	495	495K	315	630K	180	720K

Subpool Categories

Subpool categories are predefined variables that represent certain groups of subpools from which to allocate space. To use subpool categories, you must be a directory manager and have a subpool list defined in the MANAGERS file. You can specify subpool categories with the following commands:

- ADDENTRY (on an MDOPT record in the skeleton file or directory entry provided, or with the SUBPOOL option)
- ADDMDISK
- COMPRESS
- MAINT MANAGE
- MANAGE
- MULTIPLE (the second subpool parameter only)
- REPENTRY (on an MDOPT record in the entry that you provide)

Variable Subpool Categories

The subpool categories are listed in the following table, Variable Subpool Categories.

Variable Subpool Category	**Subpools Specified** (Subpools specified by a category include all the subpools listed in the DASD CONFIG file that fall in that category except those defined with the EXPLICIT attribute on a SUBPOOL record.)
*MINE	Your list of subpools
*MNGR	Your owning manager's list of subpools
*BOTH	Your owning manager's list of subpools and your list of subpools, in that order
*ANY	Your owning manager's list of subpools and your list of subpools in that order, then the remaining subpools in the reverse order of their appearance in the DASD CONFIG file **Note:** You must have ANYPOOL authorization to use the *ANY subpool category. (For more information, see the section Authorizations for Command Processing in the chapter "Authorizations" of the *Administration Guide*.)

ABEND Command

Use the ABEND command to force CA VM:Director to terminate operation abnormally.

```
ABEND
```

Description

The ABEND command forces CA VM:Director to terminate operation abnormally and causes an ABN001 abend.

Control returns to CMS with a return code of 200. CA VM:Director issues the VMDUMP command and sends the dump to the user ID specified on a DUMP record in the PRODUCT CONFIG file.

Use the ABEND command to terminate CA VM:Director only when an unusual situation arises and you need to terminate the CA VM:Director system with a dump. If you want to generate a dump of the CA VM:Director system without terminating, use the CA VM:Director CMS command to issue the CP VMDUMP command on the CA VM:Director service virtual machine's console or from an authorized user ID.

Note: For more information about terminating CA VM:Director operation under normal conditions, see the section END Command (see page 150).

ADDENTRY Command

Use the ADDENTRY command to create a new USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

To create a directory entry for a new directory profile:

```
ADDEntry name [{filetype | *} [filemode | *]] ( PROFILE "Options"
```

```
Options:
```

```
[ACCount number]
[DIST code]
[FILE [filename]]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

To use a skeleton file to create a directory entry for a new user ID:

```
ADDEntry name [skelname | *] [("Options"]
```

```
Options:
```

```
[ACCount number]
[DIST code]
[ENROLL]
[BLOcks number]
[STORGrp number]
[FILEPool poolname]
[IN identity]
[ON member]

[FILE [filename]][Format | NOFormat]
[MANager mgrid]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[LIFO]
[FIFO]
[Wait | NOWait]
```

Note: Options can be entered in any order.

To create a directory entry for a new user ID, without using a skeleton file:

```
ADDEntry name [{filetype | *} [{* | filemode}]] ( NOSKel "Options"

Options:

[ACCount number]
[DIST code]
[ENROLL]
[BLOcks number]
[STORGrp number]
[FILEPool poolname]
[IN identity]
[ON member]
[IPLstmt]
[FILE [filename]]
[Format | NOFormat]
[MANager mgrid]
[MSG | NOMSG]
[STACK [FIFO |LIFO]]
[FIFO]
[LIFO]
[SUBPOOL poolname]
[Wait | NOWait]
[STEM stemname[.]]
```

Note: Options can be entered in any order.

Authorizations

You must have ADDENTRY authorization through a GRANT ADDENTRY record in the AUTHORIZ CONFIG file.

If the skeleton file or input file used to create the directory entry specifies the creation of minidisks, you must have ADDMDISK authorization through a GRANT ADDMDISK record in the AUTHORIZ CONFIG file.

If the skeleton file or input file used to create a user ID enrolls that user ID in SFS, you must be an SFS manager authorized to allocate space in the target file pool and user storage group.

Definitions

***name* [*skelname* | *]**

Creates a directory entry, of the specified name, based on the specified skeleton file.

The entry must not already exist. The default skeleton file is the new entry's directory manager's default skeleton file, as defined in the VMDIRECT MANAGERS file. If you do not specify a *skelname*, then CA VM:Director uses a default skeleton file for the manager who is issuing the ADDENTRY command. Specifying an '*' is the same as specifying the default skeleton file name.

***name* [{*filetype*|*} [{*filemode*|*}]] (NOSKel**

Creates a directory entry, of the specified name, based on the specified input file.

The entry must not already exist. The input file must have the same name as the entry; its default filetype is DIRECT, and its filemode must be one of your currently accessed minidisks; the default is A. An asterisk for *filetype* is the same as specifying DIRECT. An asterisk for *filemode* uses the normal CMS search order to find the file.

***name* [{ *filetype*|*} [{*filemode*|*}]] (PROFILE**

Specifies the name of the directory profile to create and the filetype and filemode of the input file to use. The directory profile must not already exist.

CA VM:Director uses *name* as the filename of the input file. The default for *filetype* is DIRECT. The filemode must be one of the issuing user ID's currently accessed minidisks; the default for *filemode* is A. An asterisk for *filetype* is the same as specifying DIRECT. An asterisk for *filemode* uses the normal CMS search order to find the file.

[ACCount *number*]

Specifies the primary account number for the new user ID or directory profile, overriding the primary account number specified by the skeleton file or input file.

[DIST *code*]

Specifies the distribution code for the new user ID or directory profile, overriding the distribution code specified by the skeleton file or input file.

[ENROLL]

[BLOcks *number*]

[STORGrp *number*]

[FILEPool *poolname*]

[IPLstmt]

Enrolls the user ID in SFS, allocating the specified number of 4K blocks to this user ID in the specified user storage group in the specified file pool. Specification of any of these options enrolls the user ID in SFS. The IPL statement can also be modified to identify the file pool to use at logon.

By default, CA VM:Director uses the default number of blocks, user storage group, and file pool defined for the new user's SFS manager. The ADDENTRY command does not modify the IPL statement unless the IPLstmt option is specified on the command line or on the *ENROLL special comment in the skeleton or input file.

If you specify the IPLstmt option, you can identify a sample PROFILE EXEC file to copy to the minidisk or top-level directory by placing the *SP= special comment in the skeleton or input file. If a sample profile is available or specified on a *SP= special comment, that profile is copied to the top-level directory for the user as PROFILE EXEC. If there is already a PROFILE EXEC file in the user's top-level directory, a sample profile specified on a *SP= special comment is not copied.

[Format | NOFormat]

Formats (FORMAT) or does not format (NOFORMAT) the user's new minidisks after allocating them. This option applies to all minidisks created by this command, unless overridden by an MDOPT statement in the directory entry.

You must have NOFORMAT ADD authorization to specify NOFORMAT. CA VM:Director ignores the FORMAT or NOFORMAT option when creating minidisks on T-DISK or V-DISK. The FORMAT or NOFORMAT option on an MDOPT statement can override these options for specific minidisks.

The NOFORMAT option is not valid with the WAIT | NOWAIT option.

[SUBPOOL *poolname*]

Specifies the default SUBPOOL name used to contain minidisks allocated during ADDENTRY. This option applies to all minidisks created by this command, unless overridden by an MDOPT statement in the directory entry.

You must have ANYPOOL authorization to specify SUBPOOL *ANY. CA VM:Director ignores the SUBPOOL option when creating minidisks on T-DISK or V-DISK. The SUBPOOL option on an MDOPT statement can override these options for specific minidisks.

[IN *identity*]

Specifies that a SUBCONFIG entry will be created using either an input file or a skeleton. A BUILD statement for the new SUBCONFIG entry will be added to the named *identity* directory entry. If IN is specified, then ON must also be specified.

[ON *member*]

Specifies that a SUBCONFIG entry will be created using either an input file or a skeleton. A BUILD statement for the new SUBCONFIG will specify the named *member* of an SSI complex. The *member* name must be '*' if used in a non-SSI environment. If ON is specified, then IN must also be specified.

[MANager *mgrid*]

Specifies the directory manager for the new user ID; if the command also enrolls the user ID in SFS, this directory manager is also the user ID's SFS manager. The manager must be defined in the VMDIRECT MANAGERS file. If you do not specify this option, you are the directory manager of the new user ID.

[STEM stemname[.]]

Specifies the name of REXX stem variable array containing the statements which define the directory entry contents. Any filetype or filemode parameters are ignored. The array contains the directory entry definition, so no disk file is referenced. The number of statements in the directory entry is defined in element *stemname*.0, *stemname.*1 contains the first statement in the directory entry, and so on. The stem must be defined in the same REXX program which calls the ADDENTRY command, before the command is called.

[FILE [*filename*]]

[<u>MSG</u> | NOMSG]

[STACK [<u>FIFO</u> | LIFO]

[FIFO]

[LIFO]

[<u>Wait</u> | NOWait]

These options are explained in Common API Command Options (see page 36).

Description

Issue the ADDENTRY command to create any type of directory entry. Create a USER, IDENTITY, or SUBCONFIG entry by specifying either a skeleton file or an input file to build the directory entry. You create a directory profile by using an input file to build the directory entry.

When creating a directory entry for a USER, IDENTITY, or SUBCONFIG, CA VM:Director first creates the entry and calls any applicable user exits. If CA VM:Director detects errors, it does not create the entry, does not create any minidisks, and does not enroll the user ID in any file pools. If CA VM:Director does not detect errors, it creates minidisks and enrolls the user in a file pool if specified.

If CA VM:Director manages your SFS, then all file pools that it manages must be active and available so that it can check them all for file space that may be owned by the same user ID name as the one you are adding. If any file pool managed by CA VM:Director is not available for checking, or if CA VM:Director finds any file space in an available file pool owned by the same user ID name, the command will fail.

If CA VM:Director creates a new 191 minidisk for a new user ID or enrolls the user ID in a file pool with the IPLstmt option, you can identify a sample PROFILE EXEC to copy to the minidisk or top-level directory by placing the special comment record, *SP=, in the skeleton or input file.

When creating a directory entry for a directory profile, CA VM:Director does not call any user exits.

Note: When adding a USER or IDENTITY entry, the ADDENTRY command ignores any password specified on the USER statement and sets the initial logon password to be the same as the user ID.

New Minidisks

CA VM:Director creates minidisks for a new entry based on MDISK and MDOPT statements in the skeleton or input file from which it builds the user ID’s directory entry.

MDISK statements define minidisks. MDOPT statements are special CA VM:Director statements used to define options for minidisks. MDOPT statements precede MDISK statements; CA VM:Director applies the options specified on an MDOPT statement to all MDISK statements that follow that statement until CA VM:Director encounters an MDOPT statement with different options. In this way, the issuing user ID can specify unique options for each minidisk. If CA VM:Director does not find an MDOPT statement in the skeleton or input file, it uses options specified on the ADDENTRY command and default values described in MDOPT Statements.

When ADDENTRY executes, CA VM:Director passes the values specified on the command line, the MDISK statements, and the MDOPT statements to the ADDMDISK command, which creates the minidisks. Before it places the new directory entry in the source directory, CA VM:Director replaces the MDISK statements in the entry with the actual MDISK statements that define the new minidisks and removes the MDOPT statements.

Only the ADDENTRY and REPENTRY commands recognize MDISK and MDOPT statements in skeleton or input files. Other CA VM:Director commands that create user IDs from skeleton or input files ignore them.

MDISK Statements

CA VM:Director supports a special format for MDISK statements when used with the ADDENTRY command. This special format lets CA VM:Director automatically determine where to place each new minidisk. The parameters appear in the same order as in a regular MDISK statement, and their values are very similar to those on the ADDMDISK command.

```
Mdisk vaddr [{devtype | *} [{start | *} [{size | *} [{volid | *} ["Parms"]]]]]
Mdisk vaddr {devtype | *} TDSK [size | *]
Mdisk vaddr FB-512 VDSK [{size | *} ["Parms"]]
Mdisk vaddr {devtype | *} DEVNO [{rdev | *} ["Parms"]]
```

```
Parms:
```

```
{mode | *} [rpw [wpw [mpw]]]
```

Parameter definitions are explained in ADDMDISK Command.

MDOPT statements

```
MDOPT “Mdoptparms”
```

```
Mdoptparms:
```

```
[Format | NOFormat]
[SUBpool {poolname | *}]
[BLKsize {blksize | *}]
[Label {label | *}]
[AUnit {unitsize | *}]
```

CA VM:Director ignores the FORMAT, NOFORMAT, SUBPOOL, and LABEL options on MDOPT statements when creating minidisks on T-DISK or V-DISK. CA VM:Director ignores all options on MDOPT statements when creating minidisks with the DEVNO parameter.

Note: MDOPT statements must precede MDISK statements. CA VM:Director applies the options specified on a MDOPT statement to all MDISK statements that follow that statement until CA VM:Director encounters an MDOPT statement with different options.

[Format | NOFormat]

Specifies whether CA VM:Director is to format the new minidisks defined in MDISK statements that follow. For these minidisks, this parameter overrides the FORMAT option on the ADDENTRY command.

[SUBpool *poolname* | *}]

Specifies the subpool from which CA VM:Director is to allocate the new minidisks. The default for *poolname* is *MNGR, which searches the list of subpools in which the directory manager for this user ID can allocate space for the first subpool with enough space.

If you specify the subpool by name, that subpool must be defined in the DASD CONFIG file and included in your or the owning manager’s list of subpools in the VMDIRECT MANAGERS file. ANYPOOL authorization overrides this restriction. To specify a subpool list, use one of the variable subpool categories listed in the table, Variable Subpool Categories.

[BLKsize [*blksize* | *}]

Specifies the CMS block size CA VM:Director uses when formatting new minidisks. Possible values for blksize are 512, 1024, 2048, 4096, 1K, 2K, and 4K. The default for blksize is the value specified for AUNIT; if no value is specified for AUNIT, the default for blksize is the default block size defined for the device type.

If NOFORMAT is specified on the ADDENTRY command or the MDOPT statement for a minidisk, CA VM:Director ignores this parameter.

[Label [*label* | *}]

Specifies the minidisk label CA VM:Director is to give new minidisks when it formats them. The default is DSKvaddr. CA VM:Director replaces vaddr with the virtual device address.

NOFORMAT is specified on the ADDENTRY command or the MDOPT statement for a minidisk, CA VM:Director ignores this parameter.

[AUnit [*unitsize* | *}]

Specifies an alternate allocation unit size for CA VM:Director to use when it calculates the size of the new minidisks. When this parameter is used, CA VM:Director interprets the value for *size* on MDISK statements as the number of blocks of the size *unitsize*. It then uses this number to calculate the actual allocation size for the type of device used.

Possible values for *unitsize* are 512, 1024, 2048, 4096, 1K, 2K, and 4K. The default for *unitsize* is the specified or default value for BLKSIZE.

Example:

The following directory entry BOBBY DIRECT exists on your read/write A-disk:

```
USER BOBBY BOBBY 5M 12M G 64
INCLUDE PROJ1
MDOPT FORMAT SUBPOOL MAIN LABEL BOB191 BLKSIZE 1024
MDISK 191 3380 * 1M * MR
*ENROLL 100 2 FPMKTG
```

The following PROF1 directory profile also exists:

```
PROFILE PROF1
MACH ESA
IPL CMS PARM AUTOCR
* CONSOLE 009 3215
SPOOL 00C 2540 READER A
SPOOL 00D 2540 PUNCH A
SPOOL 00E 1403 A
*
* Get CMS disks
*
LINK MAINT 190 190 RR
LINK MAINT 19E 19E RR
LINK MAINT 19D 19D RR
```

To create user ID BOBBY and a 191 minidisk labeled BOB191 for BOBBY from subpool MAIN, and enroll him in filepool FPMKTG, enter:

```
vmdirect addentry bobby (noskel
```

Return Codes and Error Messages

The following table lists messages for errors that do not cause immediate termination but do prevent CA VM:Director from creating a directory entry. CA VM:Director validates as much of the directory entry as possible. The command exits with return code 54 and message 0633E.

ADDENTRY Command: Error Messages (No Immediate Termination)

Message Number	Text
0383I	FROM EXIT: (*status message from userexits*)
0450E	ACCOUNT NUMBER *number* REJECTED BY VM:ACCOUNT
0451E	ACCOUNT NUMBER *number* IS INVALID FOR USER *userid*
0452E	VM:ACCOUNT NOT AVAILABLE TO VERIFY NEW ACCOUNT NUMBER
0453E	UNEXPECTED RETURN CODE *code* FROM VM:ACCOUNT

The following table lists return codes and messages for errors that cause immediate termination.

Return Code	Message Number	Text
2	0018E	INVALID OPTION *option*
4	0644E	NOWAIT OPTION SPECIFIED BUT THE SERVANT FACILITY IS NOT ACTIVE
6	0648E	THE NOWAIT OPTION MAY NOT BE SPECIFIED WITH THE STACK OR FILE OPTIONS
8	0707E	SPECIAL *authorization* AUTHORIZATION IS REQUIRED TO USE THE *option* OPTION
9	0428E	SKELETON *skeleton* IS INVALID
12	0038E	MISSING PARAMETER
14	0531E	PARAMETER *parameter* IS TOO LONG
15	0328E	INVALID USERID *userid*
16	0265E	NOT AUTHORIZED FOR: *authorization words*
18	0127E	USER *userid1* EXCEEDED USER *userid2 type* PASSWORD LIMIT [FOR *vaddr*]
20	0349E	PASSWORD IS INCORRECT

Return Code	Message Number	Text
21	0364E	FILE *filename filetype filemode* IS BEING UPDATED.
22	0358E	PROFILE | USER *filename* ALREADY EXISTS
23	0041E	USER *userid* IS NOT A DIRECTORY MANAGER
25	0690E	A USERID VALUE IS REQUIRED FOLLOWING THE MANAGER OPTION
26	0358E	USER *filename* ALREADY EXISTS ON HOLD
28	0039E	INVALID PARAMETER *parameter*
30	0643E	EXTRA PARAMETERS *parameters* WERE ENTERED AND ARE NOT ALLOWED
31	0463E	MANAGER *mgrid's* USERID LIMIT OF *nnnnnn* HAS BEEN REACHED
32	0681E	FILE *filename filetype filemode* DOES NOT EXIST
37	7016E	INVALID NUMBER OF BLOCKS: *blocks*
40	0076S	USER ERROR *error* COPYING FILE *filename filetype filemode*
42	0621E	UNEXPECTED RETURN CODE *code* FROM OPEN
44	0277E	FIRST RECORD *word1 word2* CONFLICTS WITH FILEID *filename filetype* (for profiles)
46	0277E	FIRST RECORD *word1 word2* CONFLICTS WITH FILEID *filename filetype* (for users)
48	0631E	THE NAME *name* ON THE PROFILE | USER RECORD MUST MATCH THE FILENAME OF THE DIRECTORY ENTRY
52	0018E	INVALID OPTION *option*
54	0633E	ONE OR MORE ERRORS WERE DETECTED WHILE PROCESSING THE UPDATE OF USERID *userid*
56	0429E	NEWUSER EXIT INDICATES *mgrid* SHOULD BE DIRECTORY MANAGER FOR USER *userid* WITH ACCOUNT *accountnumber*
58	0363E	USER *userid* REJECTED BY INSTALLATION EXIT ROUTINE

Return Code	Message Number	Text
60	0341E	NEWUSER EXIT REJECTED USER *userid* AND ACCOUNT *account*

Note:

- For explanations of the message routing and disconnect options, see Common API Command Options (see page 36).
- For MDISK statement parameter definitions, see ADDMDISK Command (see page 53). (They are equivalent to the ADDMDISK command parameters.)
- For an explanation of the NOFORMAT authorization, see the "Authorizations for Command Processing" table in the *Administration Guide.*
- For an explanation of SKELETON files, see the section "Skeleton Files" in the *Administration Guide*.
- For an explanation of the Sample Profile comment (*SP=), see the section "Special Comment Reference" in the *Administration Guide*.
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide*.

ADDMDISK Command

Use the ADDMDISK command to add a minidisk definition within a USER, IDENTITY, or SUBCONFIG directory entry.

To create a permanent minidisk:

```
ADDMdisk entry ["Parms"] [("Options"]
```

```
Parms:
```

```
 {vaddr | *}
[{subpool | *}
[{size | *}
[{devtype | *}
[{volser | *}
[{loc | *}
[{mode | *}
[rpw[wpw [mpw ]]]]]]]]
```

```
Options:
```

```
[AUnit unitsize]
[FILE [filename]]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[BLKsize blksize]
[Format  | NOFormat]
[Label [label]]
[Wait | NOWait]
[Cache | NOCache]
[MDC | NOMDC | RECORDMDC]
[SYSCtl | DEVCtl | NOCtl]
[PAValias {vdev | vdev1-vdev2 | vdev.numdevs} ... ]
[WRKALleg | NOWRKALleg]
```

Note: Options can be entered in any order.

To create a permanent fullpack minidisk defined by a real device address:

```
ADDMdisk entry vaddr DEVNO ["Parms"] [("Options"]
```

```
Parms:

{rdev | *}
[{devtype | *}
[*
[*
[{mode | *}
[rpw [wpw [mpw ]]]]]]

Options:

[FILE [filename]]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO ]
[Cache | NOCAche]
[MDC | NOMDC | RECORDMDC]
[SYSCtl | DEVCtl | NOCtl]
[PAValias { vdev | vdev1-vdev2 | vdev.numdevs} ... ]
[WRKALleg | NOWRKALleg]
```

Note: Options can be entered in any order.

To create a temporary minidisk:

```
ADDMdisk entry {vaddr | *} TDSK [“Parms”] [("Options"]


Parms:

{size | *} [devtype |*]

Options:

[AUnit unitsize]
[FILE [filename]]
[MSG | NOMSG]
[STACK [ FIFO | LIFO]]
[FIFO ]
[LIFO ]
```

Note: Options can be entered in any order.

To create a virtual minidisk in storage:

```
ADDMdisk entry {vaddr|*} VDSK ["Parms"] [("Options"]
```

```
Parms:

{size | *}
[FB-512
[*
[*
[{mode | *}
[rpw [ wpw [ mpw ]]]]]]]
```

```
Options:

[AUnit unitsize]
[FILE [filename]]
[MSG | NOMSG]
[STACK [FIFO|LIFO]]
[FIFO ]
[LIFO ]
```

Note: Options can be entered in any order.

Authorizations

You must have ADDMDISK authorization through a GRANT ADDMDISK record in the AUTHORIZ CONFIG file.

Either you or the directory manager that will own the directory entry must be authorized to allocate from the subpool in which you specify the minidisk to be created. It is not required that both you and the owning manager be authorized. Alternately, either one of you can have ANYPOOL authorization, which allows you to allocate space from any subpool.

You must have NOFORMAT ADD authorization through a GRANT NOFORMAT ADD record in the AUTHORIZ CONFIG file to create a minidisk without formatting it.

Definitions

entry

Specifies the directory entry which will contain the minidisk. The directory entry must not be in HOLD status.

vaddr

Specifies the virtual address of the new minidisk. The default is 191.

DEVNO

Creates a permanent fullpack minidisk defined by a real device address. CA VM:Director does not control this type of minidisk and does not map the internal minidisk structure. You must specify * for both the *volser* and *loc* parameters to use the subsequent parameters.

rdev

Specifies the read device address of the volume on which to define the minidisk. No validation of this address is done until the user logs on and CP verifies the presence of the minidisk. The *rdev* parameter is used in the position of the *size* parameter in other versions of the parameter list. If you do not specify an *rdev* value or if you specify an *, CA VM:Director uses the virtual device address specified for *vaddr*.

subpool

Specifies the subpool from which to allocate space for the new minidisk or the subpool list to search for space to allocate for the new minidisk. The default is *MNGR, which searches the list of subpools in which the directory manager for this user ID can allocate space for the first subpool with enough space.

If you specify the subpool by name, that subpool must be defined in the DASD CONFIG file and included in your or the owning manager's list of subpools in the VMDIRECT MANAGERS file. ANYPOOL authorization overrides this restriction. The following predefined lists can be searched for an available subpool. The first subpool in the list with enough space to accommodate the new minidisk is the one that is used:

Subpool list	Subpools represented
*MINE	All subpools in which you, or the person submitting this ADDMDISK command, can allocate space
*MNGR	All subpools in which the directory manager for this user ID can allocate space
*BOTH	Those in the *MNGR and *MINE lists, in that order
*ANY	Any subpool defined to CA VM:Director. **Note:** Your user ID must have ANYPOOL authorization.

CA VM:Director searches subpools in a list according to the order in which the MANAGER record in the VMDIRECT MANAGERS file specifies them. It does not include in the search any subpool that is defined with the EXPLICIT attribute.

TDSK

Creates the minidisk in the system temporary disk (T-DISK) space. You can specify the device type of the T-DISK in the devtype parameter. Otherwise, CA VM:Director uses the default value that the DEVTYPE record specifies in the VMDIRECT MANAGERS file for the owning manager.

VDSK

Creates the minidisk in the system's virtual disk (V-DISK) space. You must specify FB-512 or * for *devtype* and * for both *volser* and *loc* to use the subsequent parameters.

size

Specifies the size of the minidisk. CA VM:Director interprets *size* differently, depending on the *devtype* and AUNIT command specifications. It evaluates the following possibilities in the order presented:

Size	***devtype*** **parameter**	**AUNIT option**	**Meaning of *size***
Number followed by K or M	N/A	N/A	Size in kilobytes (K) or megabytes (M), rounded up to the next allocatable unit
* or no value	N/A	N/A	Default size that the DEFSIZE record defines for the owning manager in the VMDIRECT MANAGERS file; if no DEFSIZE record exists, the default is 1M
Number	Not specified	Specified	Number of blocks of the size specified on AUNIT
Number	Specified	Not specified	Number of 512-byte blocks for FBA devices or number of cylinders for CKD devices
Number	Not specified	Not specified	Number of 512-byte blocks

devtype

Specifies the new minidisk's device type, for example, 3380 or 3390. All device types CA VM:Director supports are valid, including those defined in the DASD CONFIG file by means of the DEVTYPE record. By default, CA VM:Director determines *devtype* based on the location of the allocation.

The *devtype* parameter is a determinant in calculating the size of the new minidisk. For more information, see the description of the *size* parameter, above.

volser

Specifies the volume serial number (volser) of the DASD volume on which to allocate the new minidisk. The volume must be defined in the DASD CONFIG file and the subpool (specified on the *subpool* parameter) must be defined on one of the volume's extents. By default, CA VM:Director determines *volser* based on the location of the allocation.

loc

Specifies the location (starting cylinder or block) at which to allocate the new minidisk on a given volume. This location can be any integer up to ten digits for number of cylinders or blocks.

By default, CA VM:Director allocates the minidisk according to the algorithms defined for the applicable subpool.

mode

Specifies the read/write mode for the new minidisk when the user logs on. CP defines the valid modes, which are R, RR, W, WR, M, MR, and MW, each with an optional suffix of V, S, E, VS, VE, D, VD, SD, ED, VSD, and VED. The default mode is MR.

rpw wpw mpw

Specifies the read, write, and multiple link passwords, respectively, for the new minidisk.

Each password can be one to eight characters in length. These parameters do not have default values, and you cannot specify an asterisk for them. If one of the passwords is not specified, CA VM:Director places no value on the generated MDISK statement for that password and does not allow anyone to link to that minidisk with the respective link mode.

[AUnit *unitsize*]

Specifies an alternate allocation unit size to use to calculate the size of the new minidisk. When this option is used, CA VM:Director interprets the value for the *size* parameter as the number of blocks of the size *unitsize*. It then uses this number to calculate the actual allocation size for the type of device used.

Possible values for *unitsize* are 512, 1024, 2048, 4096, 1K, 2K, and 4K. The default for *unitsize* is the specified or default value for BLKSIZE.

[BLKsize *blksize*]

Specifies the CMS block size to use when formatting the new minidisk. Possible values for *blksize* are 512, 1024, 2048, 4096, 1K, 2K, and 4K. By default, CA VM:Director uses the value specified for AUNIT; if no value is specified for AUNIT, it uses the default block size defined for the device type.

You cannot specify BLKSIZE for minidisks allocated to T-DISK or V-DISK. If the NOFORMAT option is specified, CA VM:Director ignores the BLKSIZE option.

[CAche | NOCAche]

Enables (CACHE) or disables (NOCACHE) minidisk caching at the control unit. CA VM:Director puts the specified option on a MINIOPT record for the minidisk.

You can combine these options with the MDC and NOMDC options to get the desired level of caching. These options cannot be used with the DASDOPT record options SYSCTL, DEVCTL, and NOCTL.

[Format | NOFormat]

Formats (FORMAT) or does not format (NOFORMAT) the new minidisk after allocating it. If you format a minidisk at virtual address 191, CA VM:Director copies a sample PROFILE EXEC, if available, to the minidisk upon completion.

You must have NOFORMAT ADD authorization to specify NOFORMAT. CA VM:Director ignores the FORMAT and NOFORMAT options for minidisks defined on T-DISK or V-DISK.

[Label [*label*]]

Specifies the minidisk label to give the new minidisk during formatting. The default is DSK*nnnn*. CA VM:Director replaces *nnnn* with the device number.

If you specify the NOFORMAT option, CA VM:Director ignores the LABEL option.

[MDC | NOMDC | RECORDMDC]

MDC enables full track minidisk cache for the minidisk. NOMDC disables full track minidisk cache. RECORDMDC specifies that the minidisk will use record level minidisk caching.

CA VM:Director puts the specified option on a MINIOPT record for the minidisk.

These options cannot be used with the DASDOPT record options SYSCTL, DEVCTL, and NOCTL.

Note: For more information about MINIOPT and DASDOPT record options, see the IBM *CP Planning and Administration* guide for the z/VM release you are running.

[NOWRKALleg | WRKALleg]

WRKALLEG causes the working allegiance to be simulated on the minidisk. NOWRKALLEG causes no simulated working allegiance for the minidisk.

CA VM:Director puts the specified option on a MINIOPT record if specified for a permanent minidisk or on a DASDOPT record if the minidisk starts at cylinder 0 or is defined with DEVNO.

[PAValias {*vdev* | *vdev.numdevs* | *vdev1-vdev2*} …]

Defines a single virtual device (vdev) or multiple virtual devices which represent alias Parallel Access Volumes for a non-fullpack minidisk base Parallel Access Volume.

CA VM:Director puts the specified option on a MINIOPT record if specified for a permanent minidisk or on a DASDOPT record if the minidisk starts at cylinder 0 or is defined with DEVNO.

[SYSCtl | DEVCtl | NOCtl]

Specifies the kinds of I/O commands that can be issued against this minidisk:

Option	I/O commands that can be issued against this minidisk
SYSCTL	I/O commands that have a global effect on subsystem resources and functions, in addition to commands related to this minidisk
DEVCTL	I/O commands that affect only the resources and functions related to this minidisk
NOCTL	No I/O commands that affect subsystem resources and functions, regardless of whether they directly relate to this minidisk

CA VM:Director puts the specified option on a DASDOPT record for the minidisk.

Use these options only for full-pack minidisks. They cannot be used with the MINIOPT record options CACHE, NOCACHE, MDC, NOMDC and RECORDMDC.

Note: For more information about MINIOPT and DASDOPT record options, see the IBM *CP Planning and Administration* guide for the VM release you are running.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[Wait | NOWait]

These options are explained in Common API Command Options (see page 36).

Using ADDMDISK with Macros

The ADDMDISK command stacks a record for use by the calling macro when it successfully allocates the new minidisk if a CA VM:Director macro calls the ADDMDISK command. The stacked record is in last-in/first-out (LIFO) order and contains all the information necessary to build an MDISK record. The stacked record has the following format:

```
vaddr devtype start size volser linkmode rpw wpw mpw
```

The *start* and *size* values are in blocks for FBA and cylinders for CKD devices.

If another CA VM:Director macro does not call ADDMDISK, CA VM:Director does not stack any record.

For security reasons, the ADDMDISK command will remove pre-existing LINK statements from user directory entries that link to the newly created minidisk.

Examples

- To allocate and format a 291 minidisk for user ID BOBBY from subpool MAIN with a size of 10 cylinders and read/write access, enter:

  vmdirect addmdisk bobby 291 main 10 3380 * * mr

- To allocate and format a 291 minidisk for user ID BOBBY from subpool MAIN with a size of 10 512-byte blocks and read/write access, enter:

  vmdirect addmdisk bobby 291 main 10 * * * mr

- To allocate but not format a 2-cylinder minidisk at virtual address 391 on a 3390 for user ID BOBBY from the default subpool of BOBBY's manager, enter:

  vmdirect addmdisk bobby 391 * 2 3390 * * mr (noformat

Return Codes and Error Messages

The following table lists return codes and error messages for the ADDMDISK command.

Return Code	Message Number	Text
2	0127E	USER *userid1* EXCEEDED USER *userid2* type PASSWORD LIMIT [FOR vaddr]
4	0038E	MISSING PARAMETER
6	0531E	PARAMETER *userid* IS TOO LONG
8	0265E	NOT AUTHORIZED FOR *authorization words*
10	0349E	PASSWORD IS INCORRECT
12	0080E	USER *userid* DOES NOT EXIST
14	0632E	MANAGER *mgrid* WAS NOT FOUND IN THE MANAGERS FILE
16	0040E	TOTAL ALLOCATION LIMIT OF *number* EXCEEDED
18	0632E	MANAGER *mgrid* WAS NOT FOUND IN THE MANAGERS FILE

Return Code	Message Number	Text
20	0382E	USER *userid* IS ON HOLD
22	0621E	UNEXPECTED RETURN CODE *code* FROM OPEN
23	0304E	ADDRESS *vaddr* IS AN INVALID DEVICE ADDRESS
25	0531E	PARAMETER *vaddr* IS TOO LONG
26	0313E	DEVICE *device* DOES NOT EXIST
28	0634E	DIRECTORY PROFILE LINKS ALREADY EXIST FOR DEVICE *device*
30	none	*non-zero return code from* DIRECT_PRFLINKS
32	0635E	YOU ARE NOT A MANAGER AND HAVE NO SUBPOOLS THAT YOU ARE AUTHORIZED TO USE
33	0636E	ANYPOOL AUTHORIZATION IS REQUIRED TO ALLOCATE FROM SUBPOOL *subpool*
34	0637E	THE MINIDISK SIZE *size* MUST BE AN INTEGER NO LONGER THAN 7 DIGITS
35	0043E	SUBPOOL *subpool* IS UNDEFINED
36	0638E	A MINIDISK SIZE OF 0 IS NOT ALLOWED
37	7016E	INVALID NUMBER OF BLOCKS number
38	0547E	INVALID DEVICE TYPE *devtype* (not found in table)
40	0547E	INVALID DEVICE TYPE *devtype* (not CKD or FBA type)
41	0639E	YOU ARE NOT AUTHORIZED TO ADD A CACHE'd DEVICE
42	0640E	THE *parm* PARAMETER MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK | V-DISK
44	0531E	PARAMETER *volser* IS TOO LONG
52	0637E	THE MINIDISK SIZE *size* MUST BE AN INTEGER NO LONGER THAN 7 DIGITS

Return Code	Message Number	Text
54	0574E	CANNOT SPECIFY MINIDISK LOCATION WITHOUT VOLSER
58	0640E	THE MODE PARAMETER MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK | V-DISK
60	0642E	THE SPECIFIED LINK MODE *mode* IS INVALID
61	0531E	PARAMETER *password* IS TOO LONG
62	0640E	THE PASSWORD PARAMETER MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK | V-DISK
65	0383I	FROM EXIT: (MDISK *userexit*)
66	none	*non-zero return code from* DIRECT_QALLOC
67 (CA VM:Secure only)	none	*non-zero return code from* PENCRYPT
68	0643E	EXTRA PARAMETERS *parameters* WERE ENTERED AND ARE NOT ALLOWED
70	0644E	NOWAIT OPTION SPECIFIED BUT THE SERVANT FACILITY IS NOT ACTIVE
71	0648E	THE NOWAIT OPTION MAY NOT BE SPECIFIED WITH THE STACK OR FILE OPTIONS
72	0553E	YOU ARE NOT AUTHORIZED TO SKIP FORMATTING
73	0645E	A VALUE IS REQUIRED FOLLOWING THE *option* OPTION
74	0646E	THE SPECIFIED BLOCKSIZE *size* IS INVALID
75	0531E	PARAMETER *label* IS TOO LONG
76	0018E	INVALID OPTION *option*
77	0698E	THE *option* OPTION MAY NOT BE SPECIFIED WITH THE *option* OPTION
78	0647E	THE WAIT AND FORMAT OPTIONS MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK

Return Code	Message Number	Text
79	0677E	*devtype* MUST BE SPECIFIED; NO DEFAULT DEVTYPE FOR MANAGER *userid*
80	0364E	FILE *filename filetype* IS BEING UPDATED
81	0221E	MISSING PAVALIAS OPTION OPERAND
88	none	*non-zero return code from* DIRECT_OBTAIN
90	0209E	BLOCKSIZE 800 IS NOT SUPPORTED ON FB-512 DEVICES
91	0209E	BLOCKSIZE 800 IS NOT SUPPORTED ON FB-512 DEVICES
95	none	*non-zero return code from* DIRECT UPDATE
96	none	*non-zero return code from* MAN000S0
98	none	*non-zero return code from* MAN000R0

Note:

- For explanations of the message routing and disconnect options, see Common API Command Options (see page 36).
- For NOFORMAT authorizations, see the Authorizations for Command Processing table in the chapter "Authorizations" of the *Administration Guide*.
- For more information about the EXPLICIT attribute, see SUBPOOL Record (see page 394).
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide*.

ADMIN Command

Use the ADMIN command to create or open for edit a directory pool definition, a directory profile, or a skeleton file, or open the VMDIRECT GLOBALS file, the VMDIRECT MANAGERS file, the VMDIRECT POSIX file, or the SFS Managers Configuration Menu.

```
ADMIN {"ParmA" [prototype [ERASE]] | "ParmB"}
```

```
ParmA:

  POOL poolid
| PROFILE filename
| SKELETON filename
```

```
ParmB:

  GLOBALS
| MANAGERS
| POSIX
| SFSMGRS
```

Definitions

POOL *poolid*

Opens the directory pool definition *poolid* or creates *poolid*. The *poolid* must be one to three characters. CA VM:Director uses *poolid* as the prefix of the generated name of each of the directory pool members.

PROFILE *filename*

Opens the directory profile *filename* or creates *filename*.

SKELETON *filename*

Opens the skeleton file *filename* or creates *filename*.

GLOBALS

Opens the VMDIRECT GLOBALS file in an XEDIT session for editing.

MANAGERS

Opens the VMDIRECT MANAGERS file in an XEDIT session for editing.

POSIX

Opens the VMDIRECT POSIX file in an XEDIT session for editing.

SFSMGRS

Displays the SFS Managers Configuration Menu.

prototype

Specifies the name of the object to use as a prototype for creating a directory pool definition, directory profile, or skeleton.

ERASE

Deletes the specified directory pool definition, directory profile, or skeleton file. You can delete a directory profile only if no directory entries for existing users have an INCLUDE statement that refer to the profile.

Description

Use the ADMIN command to edit system configuration files.

Editing Directory Pool Definitions

The ADMIN POOL command lets you create, update, and delete directory pool definitions on a VM system. For example, you can change the name, configuration, or number of directory pool members.

Note: For more information about the POOL statement, see the IBM *CP Planning and Administration* guide for your release of VM.

If you are creating a directory pool definition and you do not specify a *prototype*, CA VM:Director creates the item using default records.

Example:

When you enter the following command:

vmdirect admin pool vmu

The system generates the following lines in an XEDIT session:

```
00001 USER VMU AUTOONLY 3M 3M G 64
00002 POOL LOW 00000 HIGH 00002 PROFILE DEFAULT
```

Note: You must replace the word DEFAULT with an existing PROFILE that at least contains a CONSOLE statement.

Editing Directory Profiles

If you are creating a directory profile and you do not specify a *prototype*, CA VM:Director creates the item using default records.

Editing Skeleton Files

You must specify a *prototype* if you are creating a skeleton file.

Editing the VMSECURE GLOBALS File

The VMDIRECT GLOBALS file contains GLOBALOPTS directory statements that create global definitions used during directory processing. For example, you can change the default machine mode for any user directory entry that does not contain a MACHINE statement. For more information about the GLOBALOPTS statement or the MACHINE parameter, see the IBM *CP Planning and Administration* guide for your release of VM.

CA VM:Director uses the audit code 1050 to indicate the ADMIN GLOBALS command was executed. Use of the command is also indicated in the VMXSRB report.

Editing the VMSECURE MANAGERS File

When you edit the VMDIRECT MANAGERS file, you can add a new directory manager or change an existing directory manager's allocation limit, subpool specifications, skeleton files, and default device type without any restrictions. However, you *cannot* remove an existing directory manager's records if active or held user IDs are assigned to that directory manager.

If you use the NEWUSER user exit, you may need to update it when you add a new directory manager.

Editing the VMSECURE POSIX File

When you edit the VMDIRECT POSIX file, you can:

- Add or change a group name, if the new group name is not already being used
- Delete a group name, if it or the GID, if defined, are not in use by any user
- Change the GID for a group name, if the GID is not in use by any user or the GID is defined on another POSIXGROUP record

CA VM:Director uses the audit code 1040 to indicate the ADMIN POSIX command was executed. Use of the command is also indicated in the VMXSRB report.

Working with the Product Configuration for SFS

You can use selections on this menu to authorize a directory manager to be an SFS manager, modify the allocation defaults and limits for an SFS manager, and delete an SFS manager authorization.

Examples

- To create a directory pool definition whose pool members will begin with BSD, enter:

  `vmdirect admin pool bsd`

- To create a directory profile named SALESMGR PROFILE that is based on a copy of the SALES profile, enter:

  `vmdirect admin profile salesmgr sales`

- To delete a directory profile named MKTG PROFILE, enter:

  `vmdirect admin profile mktg (erase`

- To access the skeleton file LEGAL so you can change its account number, enter:

  `vmdirect admin skeleton legal`

- For more information, see the following sections of the *Administration Guide*:
  - The chapter "Directory Managers" for adding or changing directory managers
  - The chapter "Skeleton Files" for building and using skeleton files
  - The section VMDIRECT GLOBALS File in the chapter "Configuring CA VM:Director for SFS" for configuring CA VM:Director for SFS administration
  - The chapter "POSIX Support in CA VM:Director" for information about the VMDIRECT POSIX file.

ASSIGN Command

Use the ASSIGN command to assign a USER or IDENTITY entry to a new directory manager.

```
ASsign entry mgrid [newentry [account]]
```

Definitions

entry mgrid

Identifies the USER or IDENTITY directory entry and the new directory manager to be assigned.

[*newentry* [*account*]]

Specifies the new name assigned for the entry and the new one-to-eight-character primary account number for the entry.

Description

The ASSIGN command assigns a USER or IDENTITY entry to a different directory manager from its current directory manager. It can also change the entry name and its primary account number. If you assign a new manager for an IDENTITY directory entry, the same new manager is assigned for each SUBCONFIG directory entry associated with that IDENTITY entry.

Important! Do not use the ASSIGN command with servant user IDs. The directory manager for the CA VM:Director service virtual machine and the servant virtual machines must be the same.

If you change the directory entry name through the ASSIGN command, CA VM:Director makes the change with these checks:

- Checks for pre-existing directory links to the new user ID and, if it finds any, removes them
- Updates all directory links to the minidisks owned by the old user ID to reflect the new user ID
- Renames the file spaces owned by the old user ID to reflect the new user ID, if the underlying file pool server supports renaming file spaces
- (CA VM:Secure only) Updates all rules that refer to the original user ID to reflect the new user ID, if your site uses the Rules Facility

If CA VM:Director manages your SFS and you are changing the user ID name, then all file pools that CA VM:Director manages must be active and available so that it can check them all for file space that may be owned by the same user ID name as the new one to which you are changing this user ID name. If any file pool managed by CA VM:Director is not available for checking, or if CA VM:Director finds any file space in an available file pool owned by the new user ID name, the command will fail.

If the user ID is an SFS user ID, the directory manager to which you are assigning the user ID must be an SFS manager and must have authority over any file pools and user storage groups in which the user ID is enrolled.

If the command tries to rename file spaces but the underlying file pool server does not support this function, the change fails.

Update the following to reflect the new user ID:

- Archived files (use the CA VM:Archiver reassign function)
- Backed up data (use the CA VM:Backup restore authorization user exit)
- Tapes (use the VMTAPE ASSIGN command)
- APPCPASS, CONSOLE, LOGONBY, SPECIAL, and XAUTOLOG records in the directory

If you have local user ID and account number conventions that the NEWUSER user exit enforces, changes you make through the ASSIGN command must conform to these conventions.

You cannot assign the CA VM:Director user ID with the ASSIGN command.

The ASSIGN command uses the CMS FILEPOOL EXEC to rename an individual file space within a file pool. Ensure that CA VM:Director has access to the MAINT 193 minidisk on which the FILEPOOL EXEC resides.

Examples

- To assign user ID FINANCE to directory manager ACCTMGR, enter:

  ```
  vmdirect assign finance acctmgr
  ```

- To assign user ID BOBBY under the new user ID PETER to directory manager ACCTMGR, enter:

  ```
  vmdirect assign bobby acctmgr peter
  ```

Note: For more information about SFS managers, see the chapter "Configuring CA VM:Director for SFS" in the *Administration Guide*.

AUDITEXT Command

The AUDITEXT command moves the contents of the VMDIRECT AUDIT file to your 191 minidisk (A-disk) and reinitializes the CA VM:Director AUDT minidisk.

```
AUDITEXT ["Parms"]
```

```
Parms:
```

`{`*`filename`*` | `<u>`AUDIT`</u>`} [{`*`filetype`*` | `<u>`EXTRACT`</u>`} [`*`filemode`*` | `<u>`A0`</u>`]]`

Definitions

filename filetype filemode

Specifies the name of the file in which to write the records from the AUDIT file. The default filename is AUDIT EXTRACT A0.

Description

CA VM:Director writes audit records to the VMDIRECT AUDIT file each time a user issues an audited CA VM:Director command. For performance reasons, CA VM:Director does not close the AUDIT file after it writes each record. Instead, it closes the file every 30 records, when the AUDITEXT command is issued, and at shutdown.

Your user ID must have a read/write A-disk to use the AUDITEXT command, and you must have created an AUDT minidisk for the CA VM:Director user ID and defined it on an ACCESS record in the PRODUCT CONFIG file.

Note: For the special considerations imposed when you run a Single System Image environment, see Single System Image Considerations (see page 80).

VMSECURE AUDIT File

The VMDIRECT AUDIT file exists in one of two administrator selected formats. The traditional AUDIT file consists of fixed-format, 80-byte records that contain common and variable information. The enhanced AUDIT file uses variable length record format to contain up to 256 bytes of information per record. The table provided in the following section shows the field definitions for the AUDIT file records. In a traditional AUDIT file, only 80 columns are present, and any fields past column 80 are lost. To convert the AUDIT file to enhanced format, see the description of the AUDRECFM configuration file statement in the chapter "Configuration File Reference."

VMSECURE AUDIT File Fields

The following table describes the contents of each field in those records.

Columns	Field	Description
1 – 6	date	Date the command was issued; in *yymmdd* format.
7 – 12	time	Time the command was issued; in *hhmmss* format.
13 – 20	addr	Terminal device address associated with the processing virtual machine, or **DSC** if it is a disconnected virtual machine. For additional information about terminal addresses, see the section Terminal Addresses in this chapter. All terminal addresses are left-justified and padded with blanks: ■ Real devices are represented by four hexadecimal digits. (Example: 0024) ■ Logical device addresses begin with the letter L followed by the four-digit hexadecimal device address. (Example: L0123) ■ IPv4 addresses are represented by eight hexadecimal digits. (Example: C7001350) ■ VTAM net name addresses (netid) represented by any eight characters. (Example: WEST0016) ■ IPV6ADDR ■ Remote addresses (resid) begin with the letter R followed by the four-digit hexadecimal line address. (Example: R0166)
21 – 28	userid	User ID or logon ID of the processing virtual machine. The userid is left-justified and padded with blanks.
29 – 32	cde	Audit code that represents the function performed. The audit code is a four-digit decimal number padded to the left with leading zeros. For a description of the audit codes, see the table, AUDITEXT Command Audit Codes, in this chapter.
33 – 40	$token_1$	First token, if any, associated with the audit code. The token is left-justified and padded with blanks. Typically, for CA VM:Director commands, *token1* is the user ID whose directory entry is being manipulated.

Columns	Field	Description
41 – 48	$token_2$	Second token, if any, associated with the audit code. The token is left-justified and padded with blanks. Typically, for CA VM:Director commands, *token2* is the virtual device address of the device being manipulated.
49 – 56	$token_3$	Third token, if any, associated with the audit code. The token is left-justified and padded with blanks. For the contents of *token3*, see the table, AUDITEXT Command Audit Codes, in this chapter.
57 – 64	$token_4$	Fourth token, if any, associated with the audit code. The token is left-justified and padded with blanks. For the contents of *token4,* see the table, AUDITEXT Command Audit Codes, in this chapter.
65	Flag	Identifies the type of terminal address in columns 13-20. Values for the flag are: ■ 0—Real device address ■ D—Disconnected ■ I—IPv4 address ■ 6—IPv6 address ■ L—Logical device ■ N—Net name; SNA logical unit name ■ R—Resid
66 - 80		Reserved for future use.
		Columns below are only present when the enhanced format for the AUDIT file is used. To convert the AUDIT file to enhanced format, see the description of the AUDRECFM configuration file statement in chapter “Configuration File Reference.”
81 - 112	IPv6 terminal address	When column 65 is a ‘6’, then columns 13-20 in the record will be set to the constant value ‘IPV6ADDR’, and these columns will be set to the 32 hexadecimal digits of the IPv6 terminal address.
113 - 145	IPv6 terminal address	When column 66 is a ‘6’, then columns 33-40 in the record will be set to the constant value ‘IPV6ADDR’, and these columns will be set to the 32 hexadecimal digits of the IPv6 terminal address.

Audit Codes

This table shows the CA VM:Director audit codes. Information in parentheses indicates the contents of the first, second, third, and fourth tokens for that audit code.

AUDITEXT Command Audit Codes

Audit Code	Description
10	Changed a user ID's logon password (*userid*)
20	Changed a user ID's logon storage size (*userid*, *oldsize*, *newsize*)
30	Changed a user ID's minidisk link mode and passwords. The existing mode is used if no mode change occurs. Otherwise, the new mode is recorded (*userid*, *vaddr*, *mode*)
40	Changed a user ID's virtual machine options (*userid*)
50	Changed a user ID's logical line editing symbols (*userid*)
60	Changed a user ID's screen color and highlighting (*userid*)
70	Defined a user ID's link to another user ID's minidisk (*userid*, *target userid*, *target owner's vaddr, link as vaddr*)
80	Removed a user ID's link by another user ID (*target userid, target vaddr, userid*)
90	Changed a user ID's account number (*userid*, *oldacct*, *newacct*, { TEMP | NEXTLOG | IMMED })
100	Changed a userid's distribution code (*userid*, *olddist*, *newdist*)
110	Deleted a user ID's link to another user ID's minidisk (*userid*, *target userid*, *target vaddr*)
120	Defined a new user ID (*userid*, *skeleton name*, *account*)
125	Defined a new user ID with the EDIT command (*userid*, *account*)
130	Defined a new minidisk for a user ID (*userid*, *vaddr*)
140	Scratched a user ID's minidisk (*userid*, *vaddr*)
150	Removed a user ID (*userid*, *account*, *number of minidisks*)
160	Placed a user ID on hold (*userid*, *account*, *number of minidisks*)
170	Changed the size of a user ID's minidisk (*userid*, *vaddr*)
180	Changed the device type of a user ID's minidisk (*userid*, *vaddr*)
190	Changed the location of a user ID's minidisk (*userid*, *vaddr*)
200	Edited a user ID's directory comments (*userid*)

Audit Code	Description
210	Edited a user ID's directory entry file (*userid*)
230	Edited the VMDIRECT MANAGERS file
240	Edited a skeleton file (*skeleton name*, *filename*, { CREATE | REMOVE | UPDATE })
250	Extracted directory information (*filename*, *filetype*, *filemode*)
260	Backed up the directory using VMXBKP01, optionally to tape, (*filename*, *filetype*, *filemode*)
270	Backed up the directory minidisk (*vaddr*)
280	Backed up the directory files (*filemode*)
310	Changed a virtual machine's user ID (*userid*, *new userid*)
320	Expired a virtual machine's password (*userid*)
330	Updated a command macro (*macro filename*)
340	Changed a user ID's logon and link passwords (*userid*)
345	Changed a user ID's logon password (*userid*)
350	Transferred a minidisk's ownership (*userid*, *oldvaddr*, *newuser*, *newvaddr*)
360	Assigned a user ID to a different directory manager (*userid*, *newuser*, *newacct*)
370	Changed the IPL statement or its parameters (*userid*, *sysname*, *parm1*, *parm2*)
380	Reactivated a user ID on hold (*userid*)
390	Scanned minidisks owned by a directory manager's users (*userid*, *vaddr*)
400	Changed a user ID's logon password to NOLOG (*userid*)
410	Reset a user ID's password journaling (*journal data*)
420	Directory manager reviewed a user ID (*userid*)
430	Issued TRACE command ({ MACRO | STOP }, *macro name*)
440	Started system
450	Shut down system
460	Extracted the audit file (*filename*, *filetype*, *filemode*)
470	Invalid password count exceeded (*journal data*)

Audit Code	Description
660	System administrator compressed minidisk space (*targetid*, *vaddr*, *extent start address*, *volser*)
670	Changed name of DASD volume (*userid*, *oldvol*, *newvol*, *number of minidisks*)
680	Moved a minidisk (*targetid*, *vaddr*)
710	Obtained servant user ID (*servantid*)
720	Changed address of virtual device (*userid*, *oldvaddr*, *newvaddr*)
730	Changed target user ID of directory link (*userid*, *vaddr*, *oldtuserid*, *newtuserid*)
740	Changed target virtual address of directory link (*userid*, *vaddr*, *oldtvaddr*, *newtvaddr*)
750	Edited a directory profile (*profile name*, { CREATE | REMOVE | UPDATE })
760	Updated a user ID due to profile update (*userid*, *profile name*)
770	Added a profile to a user ID's directory entry (*profile name*, *userid*)
780	Deleted a profile from a user ID's directory entry (*profile name*, *userid*)
790	Changed a profile in a user ID's directory entry (*userid*, *old profile name*, *new profile name*)
800	Changed a user ID's machine architecture (*userid*, *oldarch*, *newarch*)
810	Copied a directory entry or profile to issuing user ID's A-disk or reader ({ USER | PROFILE }, *entry name*, { A-DISK | RDR }, { LOCK | NOLOCK })
820	Replaced a directory entry or profile ({ USER | PROFILE }, *entry name*)
830	Created a permanent lock (*filename*, *filetype*, *filemode*)
840	Released a permanent lock (*filename*, *filetype*, *filemode*)
850	Moved a minidisk with R/O links (*userid*, *vaddr*, *systemid*, *newvaddr*)
860	Reclaimed a minidisk previously moved while linked R/O (*systemid*, *vaddr*)
910	Updated a configuration file ({ PRODUCT | SECURITY | AUTHORIZ | DASD })
920	Enrolled a user ID in a file pool (*userid*, *filepoolid*, *numberblocks*)

Audit Code	Description
930	Deleted a user ID from a file pool (*userid*, *filepoolid*)
940	Modified a user ID's allocation in a file pool (*userid*, *filepoolid*, *numberblocks*)
950	Added a file pool to the SFS configuration (*filepoolid*)
960	Removed a file pool from the SFS configuration (*filepoolid*)
970	Added a storage group to a file pool (*grpnumber*, *filepoolid*)
980	Removed a storage group from a file pool (*grpnumber*, *filepoolid*)
990	Gave a file pool to an SFS manager (*filepoolid*, *manager*)
1000	Removed a file pool from an SFS manager (*filepoolid*, *manager*)
1010	Gave a storage group to an SFS manager (*grpnumber*, *filepoolid*, *manager*)
1010	Removed a storage group from an SFS manager (*grpnumber*, *filepoolid*, *manager*)
1020	Removed a storage group from the list of storage groups for a file pool, from an SFS manager (*grpnumber*, *filepoolid*, *manager*)
1030	Modified CP command class (*target_userid*, *newclass*)
1040	Updated the VMDIRECT POSIX file
1050	Updated the VMDIRECT GLOBALS file
1060	Received time zone change interrupt from CP (*old_timezone_value*, *new_timezone_value*)
1070	Granted user access to SFS file spaces (*filespace_filepool*, *filespace_userid*, *target_userid*, { RD | DR | NR | WR | DW | NW }) where: ■ RD is READ ■ DR is DIRREAD ■ NR is NEWREAD ■ WR is WRITE ■ DW is DIRWRITE ■ NW is NEWWRITE

Audit Code	Description
1080	Revoked user access to SFS file spaces (*filespace_filepool*, *filespace_userid*, *target_userid*, { KD | KN | KR | NW | RW }) where: ■ KD is KEEPDIRREAD ■ KN is KEEPNEWREAD ■ KR is KEEPREAD ■ NW is NEWAUTH ■ RW is RWAUTH
1090	Accepted DIAGNOSE X’A0’ subcode X’04’ or DIAGNOSE X’88’ subcode X’08’ (*target_userid*, {VALIDATE | RULE | PASSWORD | JOURNAL })
1100	Rejected DIAGNOSE X’A0’ subcode X’04’ or DIAGNOSE X’88’ subcode X’08’ (*target_userid*, {VALIDATE | RULE | PASSWORD | JOURNAL })
1110	Accepted DIAGNOSE X’A0’ subcode X’0C’ (*target_userid, logonby_userid*, { RULE | NORULE | PASSWORD | JOURNAL })
1120	Rejected DIAGNOSE X’A0’ subcode X’0C’ (*target_userid, logonby_userid*, { RULE | NORULE | PASSWORD / JOURNAL })
1130	Minidisk duplicated *target_user*, *target_disk*, *from_user*, *from_disk*
1130	Minidisk duplicated *target_user*, *target_disk*, *from_user*, *from_disk*
1140	Accepted COUPLE ({*lanowner* | SYSTEM}, {*lanname* | *switchname*},{RULE | NORULE})
1150	Rejected COUPLE ({*lanowner* | SYSTEM}, {*lanname* | *switchname*},{RULE | NORULE})
1160	Accepted RDEVCTRL (device, {RULE | NORULE})
1170	Rejected RDEVCTRL (device, {RULE | NORULE})
1180	Accepted FOR (target_user, {RULE | NORULE})
1190	Rejected FOR (target_user, {RULE | NORULE})
1200	Accepted TRSOURCE ({RULE | NORULE})
1210	Rejected TRSOURCE ({RULE | NORULE})
1220	ENTRY ADD (target_user, type, [value])
1230	ENTRY REPLACE (target_user, type, [value])

Audit Code	Description
1240	ENTRY DELETE (target_user, type, [value])
1250	Accepted PASSCHNG (target_user, {RULE | NORULE | PASSWORD | JOURNAL})
1260	Rejected PASSCHNG (target_user, {RULE | NORULE | PASSWORD | JOURNAL})
1270	TAG ({DEFINE | MODIFY | REVOKE | QUERY | SET | SHOW | DELETE}, tag_name, [target_user])

When the AUDT Minidisk is Full

While not all CA VM:Director commands are audited, most are. (For CA VM:Secure, this includes all ACI requests generated by protected CP commands.) When an audited command is issued and the AUDT minidisk is full, CA VM:Director punches audit records to the virtual punch instead of writing them to the disk file. This ensures that no audit records are lost. For every 30 records that are punched, CA VM:Director sends the following message to the users identified on the SYSOPER records in the PRODUCT CONFIG file, the CA VM:Director console, and the issuing user:

```
AUDIT DISK FULL, WAITING FOR AUDIT EXTRACT
```

An authorized user enters the AUDITEXT command to extract the current audit records and allow further writing of audit records to disk instead of a punch spool file.

If the AUDT minidisk was full, the issuer will receive *two* files:

1. A file on the minidisk specified with the command.
2. An overflow file. The overflow file is one of the following types:
   a. A spool file in their reader when the AUDIT file has fixed length records. The spool file is named sysid OVERFLOW, where sysid is the name of the virtual machine running CA VM:Director, and contains the records punched since the AUDT minidisk filled.
   b. A disk file when a variable length record format is used for the AUDIT file. A disk file named sysid OVERFLOW is created on the A disk of the authorized user. These records can be appended to the end of the retrieved AUDIT record disk file.

When the Target Disk Is Full

When the target disk is full and you try to use the AUDITEXT command, CA VM:Director punches the entire audit file to your reader with a fileid of AUDIT EXTRACT and gives a response similar to this:

```
DMSERD107S Disk A(0191) is full
DMSVMX105S ERROR '13' WRITING FILE 'AUDIT EXTRACT A0' ON DISK.
PUN FILE 0012 FROM VMDIRECT COPY 001 NOHOLD
```

For fixed record format AUDIT files, the RDR file is in PUNCH format. For variable record format AUDIT files, the RDR file is in NETDATA format. In either case, use the RECEIVE command to load the RDR file to disk.

Note: For more information about using the VMXSRA and VMXSRB report programs to report on this data, see the sections VMXSRA Report Program and VMXSRB Report Program (see page 456).

Single System Image Considerations

When the Single System Image feature is active, a CA VM:Director server runs on each member system in the complex. Each of these servers generates an AUDIT file containing records specific to that member system. To extract the complete set of AUDIT files, a user ID authorized to issue the AUDITEXT command must log on to each member system and issue the AUDITEXT command with the LOCAL command prefix to capture the AUDIT file for that member system. Use the following command format:

```
VMDIRECT LOCAL AUDITEXT
```

CHANGE Command

Use the CHANGE command to change the name of a USER, IDENTITY, or SUBCONFIG directory entry. Changing a USER or IDENTITY entry changes the user ID. The associated user ID must be logged off for the change to work.

```
CHANGE oldentry newentry
```

Definitions

oldentry

Specifies the name of the directory entry to change.

newentry

Specifies the new directory entry name which replaces the old name.

Description

CA VM:Director does the following when you change a user ID in a USER or IDENTITY entry:

- Checks for pre-existing directory links to the new user ID and, if it finds any, automatically removes them

  Updates all directory links to minidisks owned by the old user ID to reflect the new user ID except links specified on the NOLINK record in the DASD CONFIG file

- (CA VM:Secure only) If your site uses the Rules Facility, updates all rules that refer to the original user ID to reflect the new user ID

- If the underlying file pool server supports renaming file spaces, renames the file spaces owned by the old user ID to reflect the new user ID

If CA VM:Director manages your SFS, then all file pools that are managed by the product must be active and available so that it can check them for file space that may be owned by the same user ID name as the new one to which you are changing this user ID name. If any file pool managed is not available for checking, or if it finds any file space in an available file pool owned by the new user ID name, the command will fail.

When the user ID being changed is enrolled in a CA VM:Director-controlled SFS filepool, the CHANGE command runs on a servant machine. In this case, be sure the Servant Facility is enabled; if the Servant Facility is disabled, the CHANGE command fails.

If the command tries to rename file spaces but the underlying file pool server does not support this function, the change fails.

After changing a user ID, update the following to reflect the new user ID:

- Archived files (use the CA VM:Archiver reassign function)
- Backed up data (use the CA VM:Backup restore authorization user exit)
- Tapes (using the CA VM:Tape ASSIGN command)
- APPCPASS, CONSOLE, LOGONBY, SPECIAL, and XAUTOLOG records in the directory

You cannot change the CA VM:Director user ID or a user ID specified on an IGNORE record in the DASD CONFIG file with the CHANGE command.

The CHANGE command uses the CMS FILEPOOL EXEC to rename an individual file space within a file pool. Ensure that CA VM:Director has access to the MAINT 193 minidisk on which the FILEPOOL EXEC resides.

Example

To change director entry JSMITH to JSREID, enter:

```
vmdirect change jsmith jsreid
```

Return Codes and Error Messages

The table below lists return codes and error messages for the CHANGE command.

Return Code	Message Number	Text
0	none	CHANGE command completed normally
24	038E	MISSING PARAMETER
24	039E	INVALID PARAMETER *parameter*
24	0328E	INVALID USERID *userid*
24	0357E	USER *userid* IS CURRENTLY LOGGED ON
24	0358E	USER *userid* ALREADY EXISTS
24	0531E	PARAMETER *parameter* IS TOO LONG
24	0717E	USERID *userid* IS DEFINED AS A MANAGER, AND CANNOT BE CHANGED
28	0356E	USER *userid* DOES NOT EXIST
32	0382E	USER *userid* IS ON HOLD
40	0340E	NEWUSER EXIT INDICATES USER *userid1* IS NOT DIRECTORY MANAGER FOR USER *userid2* WITH ACCOUNT *accountnumber*

Return Code	Message Number	Text
40	0343E	NO AUTHORIZED SKELETON FILES FOUND
80	0324E	ERROR *code* FROM NEWUSER EXIT
80	0464E	SCRATCH EXIT INDICATES USERID MAY NOT BE CHANGED
80	0465E	*vmdirectsvm* MAY NOT BE ASSIGNED
80	0506E	*userid* NOT ALTERED. REFERENCED IN PROFILES: *profname1 profname2* ...
7080	7080E	*userid* NAME ALREADY USED AS A FILE SPACE
7013	7013E	*userid2* ALREADY ENROLLED IN FILE POOL *filepoolid*
7111	7117E	COMMAND REJECTED -- FILE POOL *filepoolid* CANNOT PROCESS REQUEST
7236	7236E	UNEXPECTED RETURN CODE *code*, REASON CODE *reasoncode* FROM '*cslname*'

Note:

- For more information about changing a user ID directory manager, see the ASSIGN Command (see page 69).
- For more information about using the Servant Facility, see the *Administration Guide*.

CHGENTRY Command

Use the CHGENTRY command to convert a USER directory entry into an IDENTITY entry or an IDENTITY entry into a USER entry.

```
CHGENTRY entry [USER | IDENTITY]
```

Definitions

entry

This parameter names a directory entry to be changed to a different entry type.

USER

This parameter specifies that IDENTITY entry, without any BUILD statements, is to be changed into a USER entry.

IDENTITY

This parameter specifies that a USER directory entry is to be changed to an IDENTITY entry.

Authorizations

You must have authorization granted to you through the AUTHORIZ CONFIG statement, GRANT CHGENTRY. For example, to authorize MAINT to use this command, add the following AUTHORIZ CONFIG statement.

```
GRANT CHGENTRY TO MAINT
```

Description

You can convert any USER entry into an IDENTITY entry. You can convert an IDENTITY into a USER entry, but only when it contains no BUILD statements. Any BUILD statements must first be removed by deleting the associated SUBCONFIG directory entry with the DELENTRY command.

Return Codes and Error Messages

The table below lists return codes and error messages for the CHGENTRY command.

Return Code	Message Number	Text
0	0308I	ENTRY name CHANGED TO newtype
24	0038E	MISSING PARAMETER
24	0039E	INVALID PARAMETER parameter
28	0356E	USER *userid* DOES NOT EXIST
32	0382E	USER *userid* IS ON HOLD
16	0863E	SUBCONFIG name MUST BE DELETED BEFORE entry CAN BE CHANGED TO A USER

CHGMDISK Command

Use the CHGMDISK command to change or move a minidisk defined in a USER, IDENTITY, or SUBCONFIG directory entry. To change a permanent minidisk:

```
CHGMdisk entry vaddr ["Parms"] [("Options"]

Parms:

 {subpool | *}
[{size | = | *}
[{devtype | = | *}
[{volser | = | *]
[{loc | = | *}
[{mode | = | *}
[{rpw | =}
[{wpw | =}
[mpw | = ]]]]]]]

Options:

[AUnit unitsize]
[FILE [filename]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[BLKsize blksize]
[Format | NOFormat]
[Label [label]
[NOCOPY]
[Wait | NOWait]
[CAche | NOCAche]
[DEFOpt]
[MDC | NOMDC | RECORDMDC]
[WRKALleg | NOWORKALleg ]
[SYSCtl | DEVCtl | NOCtl]
[PAValias { vdev | vdev1-vdev2 | vdev.numdevs} ...]
```

Note: Options can be entered in any order.

To change a temporary minidisk:

```
CHGMdisk entry vaddr TDSK [ “Parms” ] [( “Options”]
```

```
Parms:
```

```
{size | = | *} [devtype | = | *]
```

```
Options:
```

```
[AUnit unitsize]
[FILE [filename]
[MSG | NOMSG]
[STACK [FIFO | LIFO] ]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

To change a virtual minidisk:

```
CHGMdisk entry vaddr VDSK ["Parms"] ["Options"]
```

```
Parms:
```

```
 {size|=|*}
[{FB-512 | = | *}
[{volser|=|*]
[{loc|=|*}
[{mode|=|*}
[{rpw | =} [{wpw | =} [{mpw | =}]]]]]]]
```

```
Options:
```

```
[AUnit unitsize]
[FILE [filename]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

Definitions

entry

Specifies the name of the directory entry defining the minidisk to change or move. The user ID associated with the entry must be an active user ID.

vaddr

Specifies the virtual address of the minidisk to change or move.

subpool

Specifies the subpool from which to allocate this minidisk or the subpool list to search for space to allocate for the new minidisk. The default is *MNGR, which searches the list of subpools in which the directory manager for this user ID can allocate space for the first subpool with enough space.

You can replace *subpool* with a valid CA VM:Director subpool name or list word. If you specify a subpool name, it must be defined in the DASD CONFIG file and included in the requesting or owning manager's list of subpools in the VMDIRECT MANAGERS file. ANYPOOL authorization overrides this restriction.

You can specify a subpool by name or by predefined list, which represents a variable list of subpools. The following predefined lists can be searched for available subpools. The first subpool in the list with enough space to accommodate the new minidisk is the one that is used:

Subpool list	Subpools represented
*MINE	All subpools in which you, or the person submitting this CHGMDISK command, can allocate space
*MNGR	All subpools in which the directory manager for this user ID can allocate space
*BOTH	Those in the *MNGR and *MINE lists, in that order
*ANY	Any subpool defined to CA VM:Director **Note:** Your user ID must have ANYPOOL authorization.

CA VM:Director searches subpools in a list according to the order in which the MANAGER record in the VMDIRECT MANAGERS file specifies them. It does not include in the search any subpool that is defined with the EXPLICIT attribute.

TDSK

Specifies that the minidisk to change resides in the system temporary disk (T-DISK) space. You can specify the device type of the T-DISK in the *devtype* parameter. Otherwise, the default value that the DEVTYPE record specifies is used in the VMDIRECT MANAGERS file for the owning manager.

VDSK

Specifies that the minidisk to change resides in the system virtual disk (V-disk) space. In this case, you must specify FB-512 for the *devtype* parameter and * or = for both *volser* and *loc* to use the subsequent parameters.

size

Specifies the size of the minidisk, which can be interpreted differently, depending on the *devtype* and AUNIT command specifications. The following possibilities are evaluated in the order presented:

size	*devtype* parameter	AUNIT option	Meaning of *size*
Number followed by K or M	N/A	N/A	Size in kilobytes (K) or megabytes (M), rounded up to the next allocatable unit
* or no value	N/A	N/A	Default size that the DEFSIZE record defines for the owning manager in the VMDIRECT MANAGERS file; if no DEFSIZE record exists, the default is 1M
=	N/A	N/A	The new minidisk will be the same size as the old minidisk
Number	Not specified	Specified	Number of blocks of the size specified on AUNIT
Number	Specified	Not specified	Number of 512-byte blocks for FBA devices or number of cylinders for CKD devices
Number	Not specified	Not specified	Number of 512-byte blocks

You cannot change the size of a non-CMS formatted minidisk you are moving. You may therefore specify only an equal sign for the size of a non-CMS formatted minidisk or a numeric value that is the same as its actual existing size. The actual size also becomes the default.

devtype

Specifies the device type (for example, 3380 or 3390) for the new minidisk. All device types CA VM:Director supports are valid, including those defined on the DEVTYPE record in the DASD CONFIG file.

By default, CA VM:Director determines *devtype* based on the location of the allocation. If you specify an equal sign, the device type remains the same as the original minidisk.

You cannot change the device type of a non-CMS formatted minidisk you are moving. You therefore may only specify either an equal sign or the existing device type. The default also becomes the device type of the existing minidisk.

The *devtype* parameter is a determinant in calculating the size of the minidisk. For more information, see the description of the *size* parameter, above.

volser

Specifies the volume serial number (volser) of the DASD volume on which CA VM:Director is to allocate the new minidisk. The volume must be defined in the DASD CONFIG file and the subpool (specified on the *subpool* parameter) must be defined on one of the volume extents.

By default, CA VM:Director determines *volser* based on the location of the allocation. If you specify *volser* as an equal sign, CA VM:Director allocates the new minidisk from the same volume as the original one.

loc

Specifies the location (starting cylinder for CKD devices or block number for FBA devices) at which to allocate the new minidisk on the specified volume. Specify an integer with up to seven digits for number of cylinders or blocks.

If you specify the location, you must specify the *volser* parameter as a valid volume name or an equal sign. If you specify an equal sign, CA VM:Director uses the starting location of the original minidisk, presumably on a different volume but of the same device type. If the volume name is unchanged, the allocation of the new minidisk fails, because the original minidisk still occupies that space.

If you specify an equal sign for both *volser* and *loc*, *size* and *devtype* must also be the same as the original minidisk to indicate that a new minidisk is not to be allocated. You can then change any or all of the following: the label, mode, passwords, and caching attributes.

By default, CA VM:Director allocates the minidisk according to the algorithms defined for the applicable subpool.

mode

Specifies the read/write mode for the new minidisk when the user ID logs on. CP defines the valid modes, which are R, RR, W, WR, M, MR and MW, each with an optional suffix of V, S, E, VS, VE, D, VD, SD, ED, VSD, and VED. The default for *mode* is MR. If you specify it as an equal sign, CA VM:Director uses the link mode of the original minidisk.

You can change the mode without moving the minidisk by specifying an asterisk for the subpool and an equal sign for each of the other parameters. You can also specify the original values instead of an equal sign.

rpw wpw mpw

Specifies the new read, write, and multiple link passwords, respectively, for the minidisk. Each password can be one to eight characters in length. If you specify nothing or an equal sign for any password, that password remains unchanged. To remove an existing password, you must specify the literal '$NONE$' for the password to remove and for all existing passwords following it. This literal string is a restricted keyword and should not be used for any link password.

To specify a new write password, you must specify a new read password or an equal sign. To specify a new multiple link password, you must specify new read and write passwords or equal signs.

(CA VM:Secure only) If PEF is installed, CA VM:Secure takes passwords as clear text passwords and then encrypts them.

You can change the passwords without moving the minidisk by specifying an asterisk for the subpool and an equal sign for the original values for each of the other parameters.

[AUnit *unitsize*]

Specifies an alternate allocation unit size CA VM:Director is to use when it calculates the size of the new minidisk. When this option is used, it interprets the value for the *size* parameter as the number of blocks of the size *unitsize*. It then uses this number to calculate the actual allocation size depending on the type of device used.

Possible values for *unitsize* are 512, 1024, 2048, 4096, 1K, 2K, and 4K. The default for *unitsize* is the specified or default value for BLKSIZE.

You cannot specify AUNIT for non-CMS formatted minidisks.

[BLKsize *blksize*]

Specifies the CMS block size to use when formatting the new minidisk. Possible values for *blksize* are 512, 1024, 2048, 4096, 1K, 2K, and 4K. By default, CA VM:Director uses the value specified for AUNIT; if no value is specified for AUNIT, it uses the current block size of the minidisk being moved.

You cannot specify BLKSIZE for minidisks allocated to T-DISK or for non-CMS formatted minidisks. If you specify the NOFORMAT option, the BLKSIZE option is ignored.

[CAche | NOCAche]

Enables (CACHE) or disables (NOCACHE) minidisk caching at the control unit. CA VM:Director puts the specified option on a MINIOPT record for the new minidisk.

You can combine these options with the MDC and NOMDC options to get the desired level of caching. You cannot use these options with the DASDOPT record options SYSCTL, DEVCTL, and NOCTL.

[DEFOPT]

Establishes the default caching attributes for the minidisk. This may mean that a MINIOPT record is removed or that CA VM:Director creates a MINIOPT or DASDOPT record that reflects the desired default setting for this device type on this operating system.

You cannot specify this option with any of the MINIOPT or DASDOPT options. If you do not specify this option or any of the MINIOPT or DASDOPT options, no changes are made to existing MINIOPT or DASDOPT records.

[Format | NOFormat]

Formats (FORMAT) or does not format (NOFORMAT) the old minidisk after moving it.

CA VM:Director formats the original minidisk after the move is completed using a block size of 4K and changes the label of the original minidisk to ERASED.

You must have NOFORMAT MOVE authorization to specify NOFORMAT. CA VM:Director ignores the FORMAT and NOFORMAT options for minidisks defined on T-DISK or V-DISK.

[Label [*label*]]

Specifies the minidisk label to give the new minidisk during formatting. The default is the label from the original minidisk.

You can change the label without moving the minidisk by using an asterisk for subpool, an equal sign for the rest of the parameters, and specifying the new value for the LABEL option. You can also specify the original values instead of an equal sign.

You cannot specify LABEL for non-CMS formatted minidisks.

The LABEL option and the NOCOPY option are mutually exclusive.

[MDC | NOMDC | RECORDMDC]

MDC enables full track minidisk cache for the minidisk. NOMDC disables full track minidisk cache. RECORDMDC specifies that the minidisk will use record level minidisk caching.

CA VM:Director puts the specified option on a MINIOPT record for the minidisk. You cannot use these options with the DASDOPT record options SYSCTL, DEVCTL, and NOCTL.

Note: For more information about MINIOPT and DASDOPT record options, see the IBM *CP Planning and Administration* guide for the VM release you are running.

[NOWRKALLeg | WRKALLeg]

WRKALLEG causes the working allegiance to be simulated on the minidisk. NOWRKALLEG causes no simulated working allegiance for the minidisk.

CA VM:Director puts the specified option on a MINIOPT or DASDOPT record depending on what other options are specified.

[PAValias {*vdev* | *vdev.numdevs* | *vdev1-vdev2*}]

Defines a single virtual device (vdev) or multiple virtual devices which represent alias Parallel Access Volumes for a non-fullpack minidisk base Parallel Access Volume.

CA VM:Director puts the specified option on a MINIOPT record if specified for a permanent minidisk or on a DASDOPT record if specified for a permanent fullpack minidisk defined with DEVNO.

[NOCOPY]

Moves the minidisk as specified by all other parameters of the CHGMDISK command, but does not copy the data or format the minidisk at the original or new location. Generally, use this option during a migration when it is necessary to redefine a minidisk but not copy the data. You must have NOCOPY authorization in the AUTHORIZ CONFIG file to use this option.

The LABEL option and the NOCOPY option are mutually exclusive.

[SYSCtl | DEVCtl | NOCtl]

Specifies the kinds of I/O commands that can be issued against this minidisk:

Option	I/O commands that can be issued against this minidisk
SYSCTL	I/O commands that have a global effect on subsystem resources and functions, in addition to commands related to this minidisk
DEVCTL	I/O commands that affect only the resources and functions related to this minidisk
NOCTL	No I/O commands that affect subsystem resources and functions, regardless of whether they directly relate to this minidisk

CA VM:Director puts the specified option on a DASDOPT record for the minidisk.

Use these options only for full-pack minidisks. They cannot be used with the MINIOPT record options CACHE, NOCACHE, MDC, NOMDC, and RECORDMDC.

Note: For more information about MINIOPT and DASDOPT record options, see the IBM *CP Planning and Administration* guide for the VM release you are running.

[FILE [*filename*]]

[<u>MSG</u> | NOMSG]

[STACK [<u>FIFO</u> | LIFO]]

[FIFO]

[LIFO]

[<u>Wait</u> | NOWait]

These options are explained in the Common API Command Options (see page 36).

Description

The CHGMDISK command changes or moves a minidisk for an active user ID. It can also change the label, link mode, passwords, or caching attributes of a minidisk without actually moving the minidisk.

When a CA VM:Director macro calls the CHGMDISK command, it stacks a record for use by the calling macro when it successfully moves the minidisk. The stacked record is in last in, first out (LIFO) order and contains all the information necessary to build an MDISK record to describe the minidisk new location. The stacked record has the following format:

```
vaddr devtype start size volser linkmode rpw wpw mpw
```

The *start* and *size* values are in blocks for FBA and cylinders for CKD devices.

(CA VM:Secure only) If PEF is installed, passwords are encrypted.

If another CA VM:Director macro does not call CHGMDISK, it does not stack any record.

Return Codes and Error Messages

The following table lists return codes and error messages for the CHGMDISK command.

Return Code	Message Number	Text
1	0654I	INPUT PARAMETERS RESULTED IN NO NEW VALUES: MINIDISK *userid vaddr* REMAINS UNCHANGED
2	0127E	USER *userid1* EXCEEDED USER *userid2 type* PASSWORD LIMIT {FOR *vaddr*}
4	0038E	MISSING PARAMETER
6	0531E	PARAMETER *userid* IS TOO LONG
8	0265E	NOT AUTHORIZED FOR: *authorization words*
10	0349E	PASSWORD IS INCORRECT
12	0080E	USER *userid* DOES NOT EXIST
14	0632E	MANAGER *mgrid* WAS NOT FOUND IN THE MANAGERS FILE
16	0040E	TOTAL ALLOCATION LIMIT OF *nnnnnnK* EXCEEDED
18	0632E	MANAGER *mgrid* WAS NOT FOUND IN THE MANAGERS FILE

Return Code	Message Number	Text
20	0382E	USER *userid* IS ON HOLD
22	0621E	UNEXPECTED RETURN CODE *code* FROM OPEN
23	0304E	ADDRESS *vaddr* IS AN INVALID DEVICE ADDRESS
24	0548E	MINDISK SIZE IS TOO SMALL FOR AN FB-512 DEVICE
25	0531E	PARAMETER *vaddr* IS TOO LONG
26	0313E	DEVICE *vaddr* DOES NOT EXIST
27	0120E	VOLUME *volume* NOT FOUND IN CONFIGURATION
28	0311E	DEVICE *vaddr* IS NOT A MINIDISK
29	0662E	MINIDISK *userid vaddr* IS A T-DISK AND MAY NOT BE MOVED TO OR FROM PERMANENT SPACE
30	0310E	DEVICE *vaddr* IS AN IGNORED MINIDISK
31	0561E	MINIDISK *vaddr* NOT DEFINED WITHIN *svmid* CONFIGURATION
32	0635E	YOU ARE NOT A MANAGER AND HAVE NO SUBPOOLS THAT YOU ARE AUTHORIZED TO USE
33	0636E	ANYPOOL AUTHORIZATION IS REQUIRED TO ALLOCATE FROM SUBPOOL *subpool*
34	0637E	THE MINIDISK SIZE *size* MUST BE AN INTEGER NO LONGER THAN 7 DIGITS
35	0043E	SUBPOOL *subpool* IS UNDEFINED
36	0638E	A MINIDISK SIZE OF 0 IS NOT ALLOWED
37	0719E	THE SIZE PARAMETER MAY NOT BE SPECIFIED IN BYTES FOR NON CMS FORMATTED MINIDISKS
38	0547E	INVALID DEVICE TYPE *devtype* (not found in table)
39	0338E	CANNOT CHGMDISK A CMS MINIDISK THAT HAS BEEN SAVEFD
40	0547E	INVALID DEVICE TYPE *devtype* (not CKD or FBA type)

Return Code	Message Number	Text
41	0639E	YOU ARE NOT AUTHORIZED TO ADD A CACHE'd DEVICE
42	0640E	THE VOLSER PARAMETER MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK
43	0234E	CANNOT REBLOCK A MINIDISK THAT MAY HAVE BEEN RECOMPed
44	0531E	PARAMETER *volser* IS TOO LONG
46	0653E	INFORMATION ABOUT THE *userid vaddr* MINIDISK NOT AVAILABLE; RETURN CODE *code* FROM { QRYDISK | ACCESS | LINK }
47	0573E	CANNOT LINK FOR CHANGE; VOLUME IS OFFLINE
48	0562E	MINIDISK *vaddr* NOT CHANGED, CURRENTLY IN USE
49	0585E	CANNOT MANAGE AN EMPTY *vaddr* MINIDISK
50	0640E	THE LOC PARAMETER MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK
52	0637E	THE MINIDISK SIZE *size* MUST BE AN INTEGER NO LONGER THAN 7 DIGITS
54	0574E	CANNOT SPECIFY MINIDISK LOCATION WITHOUT VOLSER
55	0225E	MINIDISK *userid vaddr* IS NOT A { T-DISK | V-DISK } AND CANNOT BE MOVED TO { T-DISK | V-DISK }
56	0655E	MINIDISK *userid vaddr* NOT MOVED; SPACE ON *volume* AT *location* IS CURRENTLY LINKED BY ANOTHER USERID
57	0704E	{ WRITE | MULT } PASSWORD SPECIFIED FOR *userid vaddr* MINIDISK REQUIRES { READ | READ/WRITE | WRITE } PASSWORDS
58	0640E	THE MODE PARAMETER MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK
59	0573E	CANNOT LINK FOR MOVE; VOLUME *volume* IS OFFLINE
60	0642E	THE SPECIFIED LINK MODE *mode* IS INVALID

Return Code	Message Number	Text
61	0554E	ERROR *code* LINKING TO MINIDISK AREA ON VOLUME *volid* AT *location* FOR
62	0640E	THE PASSWORD PARAMETER MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK
63	0531E	PARAMETER *password* IS TOO LONG
65	0625E	TOO MANY SERVANTS IN USE, COMMAND CANNOT BE EXECUTED
66	none	(non-zero return code from DIRECT_QALLOC)
67 (CA VM:Secure only)	none	(non-zero return code from PENCRYPT)
68	0643E	EXTRA PARAMETERS *parameters* WERE ENTERED AND ARE NOT ALLOWED
69	0586E	ERROR STARTING SERVANT
70	0644E	'NOWAIT' OPTION SPECIFIED BUT THE SERVANT FACILITY IS NOT ACTIVE
71	0648E	THE NOWAIT OPTION MAY NOT BE SPECIFIED WITH THE FILE OR STACK OPTIONS
72	0553E	YOU ARE NOT AUTHORIZED TO SKIP FORMATTING
73	0645E	A VALUE IS REQUIRED FOLLOWING THE *option* OPTION
74	0646E	THE SPECIFIED BLOCKSIZE *size* IS INVALID
75	0531E	PARAMETER *label* IS TOO LONG
76	0018E	INVALID OPTION *option*
77	698E	THE *option1* OPTION MAY NOT BE SPECIFIED WITH THE *option2* OPTION
78	0647E	THE *option* OPTION MAY NOT BE SPECIFIED FOR A MINIDISK ON { T-DISK | V-DISK }
79	0076S	USER ERROR *code* COPYING FILE *filename filetype filemode*
80	0364E	FILE *filename filetype* IS BEING UPDATED
81	0221E	MISSING *option* OPTION OPERAND

Return Code	Message Number	Text
82	0617E	DDR FAILED WITH RC=*code* WHILE ATTEMPTING TO MOVE THE *vaddr* DISK
83	0583I	MINIDISK *vaddr* IS NOT CMS FORMATTED, YOU CAN ONLY CHANGE THE LOCATION
85	0698E	THE *option* OPTION MAY NOT BE SPECIFIED WITH THE *option* OPTION
86	0613E	ERROR *error* COPYING FILES TO NEW DISK
87	0718E	THE { BLKSIZE | AUNIT | LABEL } OPTION MAY NOT BE SPECIFIED FOR NON CMS FORMATTED MINIDISKS
88	none	(non-zero return code from DIRECT_OBTAIN)
89	0726E	SERVANT ABENDED DURING *command* COMMAND. DIRECTORY NOT CHANGED.
90	0209E	BLOCKSIZE 800 IS NOT SUPPORTED ON FB-512 DEVICES
91	0209E	BLOCKSIZE 800 IS NOT SUPPORTED ON FB-512 DEVICES
92	0621E	UNEXPECTED RETURN CODE *code* FROM *command*
94	0480E	COPY ERROR, INPUT DISK CONTAINED *number* FILES, OUTPUT DISK HAS *number*
95	none	(non-zero return code from DIRECT UPDATE)
96	none	(non-zero return code from MAN000S0)
97	0731E	ERROR *code* REACCESSING *vmdirect vaddr* MINIDISK.
98	0383I	FROM EXIT: *text* (SCRATCH USER EXIT)
99	0383I	FROM EXIT: *text* (MDISK USER EXIT)
100	0729E	MINIDISKS OWNED BY *svmid* CANNOT HAVE THEIR LABEL CHANGED
102	0732E	YOU ARE NOT AUTHORIZED TO USE THE NOCOPY OPTION

Note:

- For explanations of the message routing and disconnect options, see Common API Command Options (see page 36).
- For MDISK statement parameter definitions, see ADDMDISK Command (see page 53). (They are equivalent to the ADDMDISK command parameters.)
- For more information about the EXPLICIT attribute, see SUBPOOL Record (see page 394).
- For more information about NOFORMAT authorizations, see the Authorizations for Command Processing table in the chapter "Authorizations" of the *Administration Guide*.
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide*.

CHGVOLNM Command

Use the CHGVOLNM command to change the volser (and all references to it) of any DASD volume controlled by CA VM:Director.

```
CHGVOLNM oldvolser newvolser {LABEL | NOLABEL}
```

When issuing the CHGVOLNM command, CA VM:Director must have OPTION D84NOPAS in its directory entry if the directory has any user IDs that have the special password of LBYONLY. Failing to have this authorization can result in errors and the command outcome can be unpredictable.

When CHVOLNM detects when more than one volume is online with the same volser, the command will exit with return code 100 and issue message 811E. To avoid this, do the following:

1. Use the CP VARY OFFLINE rdev command to vary any volumes offline, that has the same volser of the volume, you want to change. This also applies to the volumes defined as Parallel Access Volumes (PAV). *rdev* specifies the real device address of the volume.

   The volume specified by *rdev* is VARIED OFFLINE.

2. Use the CP VARY OFFLINE rdev for each PAV alias volume to make sure that only one definition of the volume is available online, so that each volser is unique.

   Each PAV alias volume specified by *rdev* is VARIED OFFLINE.

3. Execute the CHGVOLNM command to change the volser of the volume.

   The volser of the volume changes.

Definitions

oldvolser newvolser

Identifies the current volser for the volume and specifies the new volser to replace it.

LABEL | NOLABEL

Changes (LABEL) or does not change (NOLABEL) the physical volser in the volume label.

If you use LABEL, CA VM:Director detaches the volume from the system and reattaches it using the new volser. To use LABEL, the service virtual machine must have CP privilege class B or the equivalent CP authorization. No user can have an actual link to any minidisk on the volume if you run CHGVOLNM with LABEL. You cannot change the volser of a CP-owned volume.

You may use NOLABEL if the physical volser has already been changed using a separate procedure. You may also use NOLABEL if the volume is not physically available; however, since CHGVOLNM will update user directory entries to reference the new volser, users will not be able to link to their minidisks on the volume until the physical volser is renamed. You can run CHGVOLNM with NOLABEL even if users have links to minidisks on the volume.

Description

Note: The CHGVOLNM command requires a CP formatted disk label. Do not use the CHGVOLNM command against CMS-formatted minidisks.

The CHGVOLNM command allows you to change the volser of any DASD volume controlled by CA VM:Director, and to have it update all references to the volume by its new volser, including the reference in the physical label of the DASD.

CA VM:Director changes references to the old volser in the following:

- The VOLUME record in the DASD CONFIG file that identifies the volume to CA VM:Director
- All MDISK statements in all source directory entries that define minidisks on the target volume, including both minidisks CA VM:Director controls and minidisks it does not control
- Authorizations in all GRANT and WITHHOLD records to use the COMPRESS *volser* and MAP *volser* commands
- CA VM:Director control blocks, directory source files, and the object directory

Important! CHGVOLNM does not change the volser parameter on the DIRECT record in the PRODUCT CONFIG file. If you change the volser of the object directory volume, you must change that record manually; otherwise CA VM:Director will not initialize.

At its completion, CHGVOLNM sends a file called CHGVOLNM UPDATE to your A-disk. This file lists the user ID and virtual address of each MDISK statement that was updated and indicates the original volser and the new volser. If a file with this name already exists, CHGVOLNM appends the new information to the existing information in the file and indicates the date and time the changes occurred.

To prevent the CHGVOLNM command from having to wait for a lock, CA VM:Director acquires a private lock against the entire DRCT (source) disk during execution of the command. This lock prevents other directory management tasks from starting and may cause currently running tasks to wait until the command completes.

Note: If a system failure occurs while CHGVOLNM is running, the volser change may be only partially complete. You may have to manually change references to the volser that CHGVOLNM did not change before the system failure.

Examples

- To change the physical volser for a DASD volume from XA0001 to ESA001 and to update all appropriate MDISK statements, GRANT and WITHHOLD records, and internal objects to refer to the new volser, enter:

  `vmdirect chgvolnm xa0001 esa001 label`

- To change all references to the volume without changing the physical volser, enter:

  `vmdirect chgvolnm xa0001 esa001 nolabel`

CLASS Command

Use the CLASS command to change CP privilege classes in a USER or IDENTITY directory entry.

CLASS *entry classes*

Authorizations

You must have CLASS authorization through a separate GRANT CLASS record in the AUTHORIZ CONFIG file for each class you grant to a user ID.

Definitions

entry classes

Specifies the directory entry and the classes to assign. Each class is represented by a single character; multiple classes cannot be separated by blanks.

Description

IBM or your system programming staff defines valid privilege classes. You can assign these privilege classes to user IDs you manage.

The classes you assign to a user ID through this command replace all classes currently assigned to this user ID; class assignments in sequential CLASS commands or in a CLASS command after the initial class assignments are not cumulative.

This command does not change the current login session. New class privileges assigned through the CLASS command take effect at the next logon.

Examples

Directory manager WOODYB is authorized to assign CP privilege classes B, C, and G to the user IDs he manages.

- WOODYB wants to change user ID NORMP privilege classes from F to B, C, and G:

  vmdirect class normp bcg

- WOODYB wants to add privilege classes B and C to user ID CLIFFC, who already has privilege class G:

  vmdirect class cliffc bcg

CMD Command

Use the CMD command to route another command to an Agent product server in a Single System Image environment. The routed command will execute on the specified Agent server.

`CMD` *`nodename command`*

Definitions

nodename

Is the member name running an Agent server.

command

Is the command string to execute on the Agent server running on the specified member node.

Description

The CMD command sends a command to an Agent node for processing. In an SSI environment, commands issued using the product command MODULE (usually VMDIRECT MODULE or VMDIRECT MODULE) execute on the Master product server.

Note: For more information on SSI command processing, see “Command Processing in an SSI Complex” in the *Administration Guide*.

The following commands are supported for execution on an Agent server:

- ABEND
- ACITRACE
- AUDITEXT
- CMS
- DEBUG
- END
- MAP
- QCPCFG
- QLOCK
- QPCB
- QSTART
- REBUILD
- SYSWORD
- TAKEOVER
- TRACE
- VERSION

CMS Command

Use the CMS command to run a CMS subset command, CP command, or EXEC on the CA VM:Director service virtual machine.

```
CMS {cmscommand | CP command | EXEC name}
```

Definition

cmscommand

A CMS subset command (not a user area module).

CP *command*

A CP command.

EXEC *name* [*parms*]

An EXEC, including any necessary parameters to pass to the EXEC.

Description

When the CMS command transmits a CMS subset to CA VM:Director, the CMS subset active flag is set to prevent the execution of any user area modules. However, the CMS subset environment is not entered, and control always returns to CA VM:Director following the execution of the specified command.

The CMS command cannot execute a CMS or CP command that attempts a console read; it terminates abnormally if you give it such a command. If you must transmit such a command, build an EXEC that stacks the necessary command or commands before invoking it. For example, if you want to transmit the XEDIT command, build an EXEC that includes a final FFILE or QQUIT, before invoking XEDIT.

If the EXEC interpreter is being invoked, the command must begin with EXEC. CA VM:Director ignores the IMPEX flag setting.

If you transmit a CP command and the IMPCP flag is off, the command must begin with CP. If the IMPCP flag is on, the command need not begin with CP. CA VM:Director treats the command first as if it is a CMS command and then a CP command.

Note: For a description of the CMS SET IMPEX command and the CMS SET IMPCP command, see the IBM *CMS Commands and Utilities Reference* manual for your release of VM.

COMPRESS Command

Use the COMPRESS command to reduce fragmentation of allocated disk space.

`COMPRESS` *`volser number subpool1`* `[`*`subpool2`* `[`*`limit`*`]]`

Definitions

volser

The volume serial number of the DASD containing the extent to compress.

number

The starting cylinder or block number of the extent to compress. This number is the same as the starting number defined on an EXTENT record in the DASD CONFIG file.

subpool1

The name of the subpool to use when allocating space within the extent being compressed. Choose a subpool that is valid for the extent being compressed that has a large allocation limit or no allocation limit.

subpool2

The name of the subpool from which to allocate space to hold the minidisk temporarily. The default name is the same as the *subpool1* name.

limit

The number of 15-second intervals to wait before bypassing a minidisk that is not available. The command waits if other virtual machines have links to a minidisk that needs to be moved. Before waiting, the command sends these users messages asking them to detach the link. The minimum number of wait intervals is 0; the maximum is 2147483647. The default is 10 intervals.

Description

The COMPRESS command reduces and controls excessive fragmentation of disk space that is under CA VM:Director management.

All minidisks within the extent are considered movable, provided the minidisk is not in use and moving or scratching the minidisk is not restricted by the optional SCRATCH user exit.

It does this by trying to move a minidisk from the high end of the extent to fill a free area at the low end of the extent. The COMPRESS command looks for a minidisk that exactly fits the free area first. If it cannot find a matching minidisk, it looks for a minidisk that can fit in the free area. If it cannot find a minidisk that fits in the free area, the command moves the minidisk immediately following the free area to a temporary area, then copies it from the temporary area to its new location in the free area.

The compression process continues until there are no more free spaces in the extent or until the remaining free space is concentrated in one free area. When the compression process is completed, the command formats all free space to erase any data left from a minidisk that was moved during the compression.

If you are compressing the extent where your own 191 minidisk resides, you must detach it before issuing the COMPRESS command. Any minidisk can be accessed temporarily as your A-disk.

If a minidisk is in use, COMPRESS sends messages to the user ID to detach the minidisk or log off. If the user ID does not detach the minidisk or log off within the time limit, COMPRESS skips over the minidisk and continues the compression.

COMPRESS can move recomputed minidisks as long as there are no links to them. COMPRESS can move SQL minidisks as long as SQL is shut down and there are no links to the minidisks.

Examples

- You want to compress an extent on DASD volume USR01. The extent begins at block number 423000 and ends at block number 450000. The extent is located in subpool MAIN and you want to use the subpool TEMP for temporary minidisk storage, and you will wait 150 seconds for a minidisk that is not yet available (10 time intervals) before passing it. Enter the following command:

  `vmdirect compress usr01 423000 main temp`

CA VM:Director responds:

```
MOVING USER ‘ASTA’ MINIDISK ‘191’ TEMPORARILY TO VOLUME ‘USR02’
MOVING USER ‘ASTA’ MINIDISK ‘191’ TO BLOCK 423000
WAITING FOR USER ‘CINDY’ TO DETACH MINIDISK ‘191’
WAITING FOR USER ‘CINDY’ TO DETACH MINIDISK ‘191’
WAITING FOR USER ‘CINDY’ TO DETACH MINIDISK ‘191’
MOVING USER ‘CINDY’ MINIDISK ‘191’ TEMPORARILY TO VOLUME ‘USR02’
MOVING USER ‘CINDY’ MINIDISK ‘191’ TO BLOCK 425000
FORMATTING 184 BLOCKS OF FREE SPACE AT BLOCK 425800
THERE WAS A TOTAL OF 2 MINIDISK MOVES RESULTING IN 1 FREE AREA
```

- In the previous example, CA VM:Director waited for user ID CINDY to detach her 191 minidisk. To immediately bypass minidisks that need to be moved but which are currently linked, enter:

  vmdirect compress usr01 423000 main main 0

Note: For an explanation of extents, see the EXTENT Record (see page 370).

CONFIG Command

Use the CONFIG command opens user-accessible configuration files for editing or displays the SFS Configuration Menu.

```
CONFIG [Authoriz | Dasd | Product | Security | SFS]
```

Definitions

Authoriz

Opens the AUTHORIZ CONFIG file in XEDIT. You can edit all of the records in this file while CA VM:Director is running.

Dasd

Opens the DASD CONFIG file in XEDIT. You can edit all of the records in this file while CA VM:Director is running.

Product

Opens the PRODUCT CONFIG file in XEDIT. You can update most of the records in this file while CA VM:Director is running.

No changes are allowed for the following record types:

- ACCESS
- ENCRYPT (CA VM:Secure only)
- RESID
- SSINODE
- DIRECT

Security

Opens the SECURITY CONFIG file in XEDIT. You can edit all of the records in this file while CA VM:Director is running except the AUTOPASS and (CA VM:Secure only) ENABLE records.

SFS

Displays the SFS Configuration Menu.

Description

When you want to make changes to any of the four product configuration files, the CONFIG command opens the specified configuration file in XEDIT so that you can review or change the CA VM:Director configuration. You can change most of its configuration while it is running.

When you want to make changes to the SFS configuration, the CONFIG SFS command displays the SFS Configuration Menu, which allows you to change the configuration for SFS.

If you do not include a parameter on the CONFIG command, the System Configuration Menu displays, from which you can choose the configuration file to open or the SFS Configuration Menu.

When you save a configuration file you have modified, the system verifies your changes and puts them in effect immediately. If it finds a nonexistent user ID on a record when it updates the file, it writes warning messages on your console but still updates the file.

When you exit the SFS Configuration Menu after changing the configuration, CA VM:Director puts those changes in effect immediately.

Note: For a complete description of the records in each of the CA VM:Director configuration files, see Configuration File Reference (see page 351).

In the *Administration Guide* see the following:

- The chapter "Configuring CA VM:Director for SFS" for a complete description of the tasks required for configuring SFS administration
- The chapter "Configuration Files and Records" for an explanation of the effects of changing records in the configuration files.

CPFMTXA Command

Use the CA VM:Director CPFMTXA command to update the allocation map of the object directory volume without shutting down CA VM:Director.

```
CPFMTXA parameters
```

Definitions

parameters

Provide any valid CPFMTXA command parameters for the re-allocation function. If no parameters are specified, the requesting userid is prompted by the CPFMTXA command for input. Refer to the IBM CP Commands and Utilities Reference manual for valid CPFMTXA re-allocation parameters. To avoid problems, we recommend you place a note in your CPFMTXA utility documentation to remind staff to use the CA VM:Director CPFMTXA command when allocation changes are being made to the volume containing your object directory.

Authorizations

You must have authorization granted to you through the AUTHORIZ CONFIG statement, GRANT CPFMTXA. For example, to authorize MAINT to use this command, add the following AUTHORIZ CONFIG statement.

```
GRANT CPFMTXA TO MAINT
```

Description

CA VM:Director maintains an in-storage copy of the CP allocation map for the disk which contains the CP Object Directory. As the Object Directory is updated by changes, the CP allocation map is updated in memory and on disk to mark Object Directory cylinders or blocks as either free or in-use. If a system administrator attempts to update the allocation map on the volume containing the Object Directory, using the IBM CPFMTXA program, then CA VM:Director will cause those changes to be lost whenever it re-writes the allocation map from the in-storage copy.

The CA VM:Director CPFMTXA command allows a system administrator to update the allocation map on the volume containing the Object Directory, without taking CA VM:Director down.

Use extreme caution when using this command. The user must be completely familiar with the use of the IBM CPFMTXA command for changing volume allocation before attempting to use it. This CA VM:Director command runs the IBM CPFMTXA command in the issuer virtual machine. CPFMTXA calls the ICKDSF MODULE. Make sure that the MODULE file is present in virtual machine using this command. For more information, see the IBM *Device Support Facilities User's Guide and Reference*. In an SSI environment, the issuer must be logged on to the SSI member system whose Object Directory volume allocation map is being updated.

The CA VM:Director CPFMTXA performs these actions in the specified order:

1. Any in-flight updates of the Object Directory are completed, and the in-memory copy of the CP allocation map is written to disk.
2. A lock is established to prevent any subsequent modifications to the Object Directory and the associated allocation map by any CA VM:Director command or process.
3. The IBM CPFMTXA command is executed with the specified parameters. The execution occurs in the virtual machine which issued the CA VM:Director CPFMTXA command, not in the CA VM:Director server machine. The command issuer is responsible for establishing the necessary write access to a minidisk which covers the area from the beginning of the real volume (cylinder zero or block 0) to at least the end of the allocation map. CA VM:Director does not validate any of the parameters to the CPFMTXA command, and just passes them to the IBM CPFMTXA command as specified.
4. At the end of execution of the IBM CPFMTXA command, CA VM:Director reads in and analyzes the allocation map. A CA VM:Director REBUILD operation is then executed to consolidate free space within the object directory and verify any changes. After validating all changes to the allocation map, the in-storage copy of the allocation map is refreshed, and the lock established in step 2 is released so that subsequent modifications of the allocation map by CA VM:Director may proceed.

All formatting of the volume should be done outside of CA VM:Director so that its performance is not impacted. Only re-allocations to the object directory volume should take place through the CPFMTXA command. At times, the execution of the IBM CPFMTXA command may complete successfully with a return code zero, but the REBUILD phase may have errors. For instance, if you change directory cylinders (DRCT) to non-directory cylinders, the CPFMTXA allocation will end successfully but CA VM:Director will abend during the REBUILD phase.

The following describes types of changes that can be made with CPFMTXA ALLOCATE processing and their impact on CA VM:Director processing:

1. Allocation is changed for the volume but no changes were made to Object Directory cylinders or blocks (type DRCT). Normal operation of CA VM:Director is resumed. Keep in mind that CP will not recognize any non-directory allocation changes on the volume until the volume is attached to the system. This usually happens only at IPL time. Care should be taken with the timing of these changes so that they are performed just prior to system IPL.
2. Non-directory areas have been changed to type directory (DRCT). CA VM:Director will start using the new cylinders or blocks to contain pages of the Object Directory. Normal operation of CA VM:Director is resumed.
3. Directory cylinders or blocks have been changed to non-directory type. CA VM:Director operation cannot continue, since the Object Directory will have been wiped out by this change. CA VM:Director will ABEND with code RXB101 after issuing message 699W. In the event of this error, it may be possible to restart CA VM:Director quickly with the SOURCE start option, so that the directory are is re-written. If this is not possible, then the Object Directory and allocation map must be restored from backups.

DELENTRY Command

Use the DELENTRY command to delete a USER, IDENTITY, SUBCONFIG, or PROFILE directory.

```
DELEntry entry [("Options"]

Options:

[FILE [filename]]
[Format | NOFormat]
[HOLD]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[Wait | NOWait]
```

Note: Options can be entered in any order.

To delete a directory profile:

```
DELEntry entry (PROFILE "Options"

Options:

[FILE [filename]]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

Authorizations

If a user ID has minidisks to be deleted, you must have DELMDISK authorization through a GRANT DELMDISK record in the AUTHORIZ CONFIG file.

If you want to leave minidisks that have been deleted from a directory entry unformatted, you must have NOFORMAT DELETE authorization through a GRANT NOFORMAT DELETE record in the AUTHORIZ CONFIG file.

Definitions

entry

Specifies the name of the directory entry to delete.

***entry* (PROFile**

Specifies the name of the directory profile to delete.

[Format | NOFormat]

Formats (FORMAT) or does not format (NOFORMAT) the minidisks defined in the directory entry before deleting them. You must have the appropriate NOFORMAT authorization to specify NOFORMAT. (If you include this DELENTRY command in a program, the user ID that submits the program for execution must have the appropriate NOFORMAT authority.)

Formatted minidisks are formatted using a block size of 4K, and their labels are changed to ERASED.

CA VM:Director ignores this option for minidisks defined on T-DISK or V-DISK.

[HOLD]

Indicates that the specified directory entry is currently on hold. This option is required if you want to delete a directory entry on hold.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[LIFO]

[FIFO]

[Wait | NOWait]

These options are explained in the Common API Options (see page 36).

Description

The DELENTRY command lets you delete the directory entry for a USER, IDENTITY, SUBCONFIG, or PROFILE regardless of whether it is active or on hold. When it deletes the directory entry for a user ID, it also deletes the:

- User ID minidisks using the DELMDISK command
- User ID from all file pools in which it was enrolled

- User ID from the VMDIRECT MANAGERS file

(CA VM:Secure only) If your site uses the Rules Facility, deletes the USRRULES file for the user ID and any references to the user ID in any other rules file

Note: When you use the DELENTRY command and delete a minidisk that is on a pack that is under CA VM:Director control, CA VM:Director also deletes all links to that deleted minidisk from the directory.

You can delete an active user's directory entry only under the following conditions:

- User ID is not currently logged on
- None of the user ID minidisks is linked
- No directory profile specifies a link to any of its minidisks
- All file pools in which the user is enrolled are active

You can delete a directory profile only if no directory entry refers to it with an INCLUDE statement. Use the QUERY PRFUSERS command to list all users of a directory profile.

When you delete a SUBCONFIG directory entry, the associated BUILD statement in the related IDENTITY directory entry is removed.

When you delete an IDENTITY directory entry, any related SUBCONFIG directory entry is also deleted.

Example

To delete user ID NICKT, which is on hold, format NICKT's deleted minidisks, and send messages generated by the deletion to the default CMS file (DELENTRY RESULTS A), enter:

```
vmdirect delentry nickt (hold format file
```

Return Codes and Error Messages

The table below lists immediate termination return codes and error messages for the DELENTRY command.

Return Code	Message Number	Text
2	0018E	INVALID OPTION *option*
4	0644E	'NOWAIT' OPTION SPECIFIED BUT THE SERVANT FACILITY IS NOT ACTIVE

Return Code	Message Number	Text
6	0648E	THE NOWAIT OPTION MAY NOT BE SPECIFIED WITH THE STACK OR FILE OPTIONS
8	0553E	YOU ARE NOT AUTHORIZED TO SKIP FORMATTING
10	0698E	THE PROFILE OPTION MAY NOT BE SPECIFIED WITH THE HOLD OPTION
12	0038E	MISSING PARAMETER
14	0531E	PARAMETER *userid* IS TOO LONG
15	0328E	INVALID USERID *userid*
16	0265E	NOT AUTHORIZED FOR: DELENTRY *userid*
17	7068E	YOU ARE NOT AUTHORIZED TO WORK WITH *userid*
18	0127E	USER *userid1* EXCEEDED USER *userid2* CA VM:Director LOGON PASSWORD LIMIT {FOR *vaddr*}
20	0349E	PASSWORD IS INCORRECT
22	0080E	USER *userid* DOES NOT EXIST
24	0435E	USER *userid* IS NOT ON HOLD
25	0270W	PROFILE *entryname* DOES NOT EXIST
26	0382E	USER *userid* IS ON HOLD
28	0643E	EXTRA PARAMETER(S) *parameters* WERE ENTERED AND ARE NOT ALLOWED
29	0254E	PROFILE *profname* IS USED BY *number* USERS AND MAY NOT BE REMOVED
30	0383I	FROM EXIT: SCRATCH *userexit*
32	0357E	USER *userid* IS CURRENTLY LOGGED ON
34	0621E	UNEXPECTED RETURN CODE *code* FROM OPEN
36	0506E	*userid* NOT ALTERED. REFERENCED IN PROFILES: *profname1 profname2* ...
38	0710E	MANAGER *mgrid* STILL HAS USERIDS DEFINED ON THE { DRCT | HOLD DISK }

Return Code	Message Number	Text
40	0711E	FILE *filename filetype filemode* CANNOT BE UPDATED BECAUSE IT IS LOCKED BY ANOTHER USER
42	0712E	AN ERROR OCCURRED WHILE ATTEMPTING TO PROCESS THE *filename filetype* FILE
44	0730E	USER *userid* NOT REMOVED, MINIDISK *mdisk* IS CURRENTLY IN USE
46	0653E	INFORMATION ABOUT THE *userid vaddr* MINIDISK NOT AVAILABLE; RETURN CODE *code* FROM { QRYDISK | ACCESS | LINK }
50	0573E	CANNOT LINK FOR *action*; VOLUME *volser* IS OFFLINE
7223	7223E	ERROR code FROM command

Note:

- For explanations of the message routing and disconnect options, see Common API Command Options (see page 36).
- For information about how to check on links to a minidisk if you cannot delete a minidisk because it is linked, see DISPLINK Command (see page 127).
- For an explanation of the QUERY PRFUSERS command, see QUERY Command (see page 303).
- For NOFORMAT authorizations, see Authorizations for Command Processing table in the chapter "Authorizations" of the *Administration Guide*.
- *Message Reference Guide* for complete descriptions of all messages listed in this section.

DELETE Command

Use the DELETE command to delete an SFS file space owned by a user ID.

```
DELETE userid [filepool | ALL | *] [("Options"]

Options:

[FILE [filename]]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

Authorizations

You must be the SFS manager for the user ID and you must be authorized to allocate space in the affected file pools, or you must be an SFS administrator.

Definitions

userid

Specifies the user ID whose file space you want CA VM:Director to delete. The user ID must be active.

[*filepool* | ALL]

The name of an existing file pool from which to delete the user's file space, or ALL to delete the user's file space for all file pools that CA VM:Director manages.

The specified file pool must be one that CA VM:Director manages and in which CA VM:Director knows the user ID owns file space. ALL deletes the user's file space from all file pools managed by CA VM:Director that the user is known to be enrolled in.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]

[FIFO]

[LIFO]

These options are explained in the Common API Command Options (see page 36).

Description

The DELETE command deletes an active user ID file space from one or all file pools that CA VM:Director manages. If you want to delete file space that belongs to a user ID on hold, you must reactivate it first.

Note: CA VM:Director runs the DELETE command on a servant machine. Before issuing the DELETE command, be sure the Servant Facility is enabled; if the Servant Facility is disabled, the DELETE command fails. For more information about using the Servant Facility, see the *Administration Guide*.

Deleting a user's file space deletes all data that user ID has in the specified file pool (or in all file pools that CA VM:Director manages if you specify ALL). Verify, before you delete a user’s file space, that the user does not need the file space or any data currently stored in that file pool.

The file pool you specify on the DELETE command must be active and available; otherwise the command will fail. If you specify all file pools in which the user ID is enrolled (ALL), then all file pools must be active and available; if any one of them is not, the command will fail.

Deleting a file space makes that file space available for reallocation. It is the opposite of enrolling a user with a particular file space.

Return Codes and Error Messages

The following table lists return codes and error messages for the DELETE command.

Return Code	Message Number	Text
39	0039E	INVALID PARAMETER *parameter*
99	0099I	*command* COMMAND CANCELED
328	0328E	INVALID USERID *userid*
7002	7002E	YOU ARE NOT AUTHORIZED TO MANAGE USER *userid*
7003	7003E	SFS ERROR REASON CODE 543 DURING DELETE FOR *userid filepoolid*
7022	7022E	YOU ARE NOT AUTHORIZED TO MANAGE *userid* IN *filepoolid*
7025	7025E	userid NOT YET ENROLLED IN filepoolid
7028	7028E	USERID REQUIRED

Return Code	Message Number	Text
7068	7068E	YOU ARE NOT AUTHORIZED TO WORK WITH *userid*
7082	7082E	REQUIRED FIELD *fieldname* NOT PROVIDED
7098	7098E	UNABLE TO VERIFY STATUS OF FILESPACE *userid*. ONE OR MORE FILEPOOLS ARE UNAVAILABLE

Note:

- For explanations of the message routing options, see Common API Command Options (see page 36).
- For more information about reducing (or increasing) a user's allocation in a file pool without deleting it entirely, see MODIFY Command (see page 275).
- For more information about using the Servant Facility, see the *Administration Guide*.
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide*.

DELMDISK Command

Use the DELMDISK command to delete a minidisk defined within a USER, IDENTITY, or SUBCONFIG directory entry.

```
DELMdisk entry vaddr [("Options"]

Options:

[FILE [filename]]
[Format | NOFormat]
[MSG | NOMSG]
[STACK [FIFO| LIFO]]
[FIFO]
[LIFO]
[Wait | NOWait]
```

Note: Options can be specified in any order.

Authorizations

You must have DELMDISK authorization through a GRANT DELMDISK record in the AUTHORIZ CONFIG file.

If you want to leave the contents of minidisks that have been deleted intact, then you must have NOFORMAT DELETE authorization through a GRANT NOFORMAT DELETE record in the AUTHORIZ CONFIG file.

Definitions

entry vaddr

Specifies the owner and virtual address of the minidisk to delete. The directory entry must not be in HOLD status.

[Format | NOFormat]

Formats (FORMAT) or does not format (NOFORMAT) the specified minidisk after deleting it. Formatted minidisks are formatted using a block size of 4K, and their labels are changed to ERASED. You must have NOFORMAT or NOFORMAT DELETE authorization to specify NOFORMAT. (If you include this DELMDISK command in a program, the user ID that submits the program for execution must have NOFORMAT DELETE authority.)

CA VM:Director ignores this option for minidisks defined on T-DISK or V-DISK.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[Wait | NOWait]

These options are explained in the Common API Commands Options (see page 36).

Description

The DELMDISK command deletes minidisks USER, IDENTITY, or SUBCONFIG directory entries. You can delete a user's minidisks only under the following conditions:

- User ID is not currently logged on
- Minidisks you want to delete are not linked
- No directory profile specifies a link to any of the minidisks you want to delete

Note: When you use the DELMDISK command to delete a minidisk that is on a pack that is under CA VM:Director control, CA VM:Director also deletes links to that deleted minidisk from the directory.

Example

To delete the NICKT 193 minidisk without formatting it, enter:

```
vmdirect delmdisk nickt 193 (noformat
```

Return Codes and Error Messages

The table below lists return codes and error messages for the DELMDISK command.

Return Code	Message Number	Text
2	0127E	USER *userid1* EXCEEDED USER *userid2 type* PASSWORD LIMIT
4	0038E	MISSING PARAMETER
6	0531E	PARAMETER *userid* IS TOO LONG
8	0265E	NOT AUTHORIZED FOR: DELMDISK *userid*
10	0349E	PASSWORD IS INCORRECT

Return Code	Message Number	Text
12	0080E	USER *userid* DOES NOT EXIST
20	0382E	USER *userid* IS ON HOLD
22	0633E	ONE OR MORE ERRORS WERE DETECTED WHILE PROCESSING THE UPDATE OF USERID *userid*
23	0304E	ADDRESS *vaddr* IS AN INVALID DEVICE ADDRESS
25	0531E	PARAMETER *vaddr* IS TOO LONG
26	0313E	DEVICE *vaddr* DOES NOT EXIST
28	0634E	DIRECTORY PROFILE LINKS ALREADY EXIST FOR DEVICE *vaddr*
29	0311E	DEVICE *vaddr* IS NOT A MINIDISK
30	none	non-zero return code from DIRECT_PRFLINKS
31	0561E	MINIDISK *vaddr* NOT DEFINED WITHIN *svmid* CONFIGURATION
32	0310E	DEVICE *vaddr* IS AN IGNORED DEVICE
41	0639E	YOU ARE NOT AUTHORIZED TO DELETE A CACHE'd DEVICE
46	0653E	INFORMATION ABOUT THE *userid vaddr* MINIDISK NOT AVAILABLE; RETURN CODE *code* FROM { QRYDISK | ACCESS | LINK }
48	0562E	MINIDISK *vaddr* NOT *action*, CURRENTLY IN USE
49	0339E	MINIDISK *vaddr* HAS BEEN PROCESSED BY THE CMS SAVEFD COMMAND
50	0573E	CANNOT LINK FOR FORMAT; VOLUME IS OFFLINE
68	0643E	EXTRA PARAMETER(S) *parameters* WERE ENTERED AND ARE NOT ALLOWED
70	0644E	'NOWAIT' OPTION SPECIFIED BUT THE SERVANT FACILITY IS NOT ACTIVE
71	0648E	THE NOWAIT OPTION MAY NOT BE SPECIFIED WITH THE STACK OF FILE OPTIONS
72	0553E	YOU ARE NOT AUTHORIZED TO SKIP FORMATTING
76	0018E	INVALID OPTION *option*
78	0647E	THE { WAIT | FORMAT } OPTION MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK

Return Code	Message Number	Text
80	0364E	FILE *filename filetype* IS BEING UPDATED
89	0726E	SERVANT ABENDED DURING *command* COMMAND. DIRECTORY NOT CHANGED.
95	none	non-zero return code from DIRECT UPDATE
96	none	non-zero return code from MAN000S0
97	none	non-zero return code from MAN000R0
98	0383I	FROM EXIT: *text* (SCRATCH USER EXIT)

Note:

- For explanations of the message routing and disconnect options, see Common API Command Options (see page 36).
- For more information about how to check on links to a minidisk if you cannot delete a minidisk because it is linked, see DISPLINK Command (see page 127).
- For NOFORMAT authorizations, see the Authorizations for Command Processing table in the chapter "Authorizations" of the *Administration Guide.*
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide*.

DISPLINK Command

Use the DISPLINK command to display the active links to a minidisk.

Displink *userid* [*vaddr*]

Definitions

***userid* [*vaddr*]**

Specifies the owner and device address of the minidisk whose active links you want CA VM:Director to display. You cannot specify your own user ID. The default virtual address is 191.

Description

To determine the active links to the specified minidisk, the DISPLINK command links the minidisk from a servant virtual machine (or your virtual machine if your site is not using the Servant Facility), issues the CP QUERY LINKS command, then detaches the minidisk.

If you find that you cannot manipulate a minidisk because it is still in use, you can use the DISPLINK command to determine which user IDs have LINKs to that minidisk.

Example

To display active links to the FINANCE 191 minidisk, enter:

vmdirect displink finance 191

CA VM:Director responds:

```
PLANNING 100 R/O
FINANCE 191 R/W
```

In an SSI environment, the linking ID member system location is appended as follows:

```
PLANNING 100 R/O FROM SYSTEM1
FINANCE  191 R/W FROM SYSTEM2
```

DUPENTRY Command

Use the DUPENTRY command to create a directory entry for a new user ID, based on the directory entry for an existing user ID.

```
DUPEntry userid1 userid2 [ ( "Options" )

Options:

[MANager mgrid]
[FILE [filename]]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

Authorizations

You must have both DUPENTRY authorization and DUPMDISK authorization through GRANT DUPENTRY, GRANT DUPMDISK, GRANT ADDMDISK, and GRANT NOFORMAT records in the AUTHORIZ CONFIG file.

In order to use the MANAGER option, you must have ASSIGN authorization through a GRANT ASSIGN record in the AUTHORIZ CONFIG.

DUPENTRY command authorization has two parameters, the template user ID and the duplicated user ID.

The two parameters follow the specifications documented in the *Administration Guide*, in the Authorizations chapter. Information for the GRANT and WITHHOLD configuration file records can be found in Configuration File Reference (see page 351).

Definitions

userid1

Specifies the name of an existing user ID to be used as a pattern or template for creating a new user ID directory entry. The user ID must exist.

userid2

Specifies the name of a user ID to be created. The user ID must not exist.

[MANager *mgrid*]

Specifies the directory manager for the new user ID. The manager must be defined in the VMDIRECT MANAGERS file. If you do not specify this option, you become the manager of the new user ID. If '=' is specified, then the manager of userid1 becomes the manager of userid2.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]

[FIFO]

[LIFO]

[Wait | NOWait]

These options are explained in Common API Command Options (see page 36).

Description

The DUPENTRY command creates a directory entry for a new user ID based on an existing user ID, which is processed as a template or pattern. With some exceptions documented below, such as unique or dedicated resources, the new user ID is a duplicate of the template user ID.

The same devices and options are present, and the minidisks are exact copies of the disks in the template user ID.

When creating a directory entry for a user ID, CA VM:Director first creates a basic directory entry. If CA VM:Director detects errors it does not create the directory entry, and does not create any minidisks.

If CA VM:Director does not detect errors, it creates minidisks by duplicating the disks from the template user ID.

When creating the directory entry for the new user ID, CA VM:Director calls the NEWUSER User Exit.

Note: For more information about implementing the NEWUSER user exit, see User Exit Reference (see page 405).

The DUPENTRY command calls the DUPMDISK command for each minidisk device in the template. New minidisks, each containing an exact copy of the data on each template minidisk, are created in the new user ID.

Note: When duplicating a user ID, the DUPENTRY command ignores any passwords specified on the USER statement in the template user ID and sets the initial logon password to be the same as the new user ID.

The password for the new user ID can be reset during initial LOGON or by using the PASSWORD, MANAGE, or MAINT MANAGE commands.

However, the special passwords LBYONLY or AUTOONLY are propagated to the new user ID, if found in the template.

New Minidisks

CA VM:Director creates minidisks for the new user ID based on MDISK statements in the template user ID directory entry. The data from the template user ID minidisks is copied to the new user ID minidisks. Minidisk access passwords from the template are not carried forward to the new user ID.

Passwords for minidisk access may be added using the CHGMDISK, MANAGE, MAINT MANAGE, or USER commands.

Exceptions

There are certain definitions which may be present in a template user ID directory entry that are used to configure unique resources. For example, a DEDICATE statement specifies that a device is allocated for sole use of a specific virtual machine. This type of unique resource is not copied to the configuration of the user ID being created by this command, because the unique resource cannot be shared between the template user ID and the new user ID.

There are also status records present in the user directory entry for the template user ID which have no meaning in the new user ID. An example might be the record of the last time the template user ID was logged on.

This status information is not copied to the new user ID.

If you are using CA VM:Secure RULES, then no RULES for the template user ID are carried over to the new user ID. The appropriate access rules must be added manually for the new user ID.

Here is the list of the records or record parameters which receive special processing during execution of this command. These statements are not copied, but instead are omitted from the new user ID directory entry and documented by response message VMX0838I.

- A DEDICATE statement defines a device which is accessed by only a single user ID. Such a device is not sharable. An alternate device could be dedicated if necessary.
- A MACID parameter on a NICDEF statement defines a unique identifier for a virtual network interface card device. An alternate MACID should be assigned if needed.
- An APPCPASS statement is not carried over because the authorization it grants may not be appropriate.
- It is not possible to determine whether parameters defined on a MINIOPT or DASDOPT statement for a minidisk in the template user ID should be applied to the copy of the minidisk created by this command. These MINIOPT or DASDOPT statements are not carried over to the new image.
- Any minidisk defined with the END specification is not copied to the new user ID directory entry. Full volume minidisks should be added manually, if appropriate.
- MDISK passwords are removed with the exception for the case where the read password; the read and write password; or the read, write, and multiple passwords are set to 'ALL'.

Special comment records are used to record status for an individual user Special comment records are used to record status for an individual user Id and are not copied over to the new image.

The following comment records are not copied:

- *PWnn statements contain password phrase information.

- *ED= records contain a record of the last modification made to a directory entry.
- *EX= is present when the user ID password has expired.
- *FP= indicates that a user ID is enrolled in a Shared File System filepool.
- *HS= contains history status. A blank *HS= record is substituted for each *HS=.
- *LA= contains the time of the last AUTOLOG of a user ID.
- *LL= contains the time of the last LOGON of a user ID.
- *LT= contains the identifier of the last IPv6 terminal used to LOGON to the user ID.
- *NB= is set when the PASSWORD command is used with the BYONLY option.
- *SP= specifies that a PROFILE EXEC is to be created.
- *US= sets information passed to a POSTRULE User Exit.
- The date field is removed from any *PW= statements, in order to preserve the specification that previously used passwords are to be remembered for later processing by a PASSWORD User Exit. The date field records when the password was updated, and does not apply to the created user ID.

Return Codes and Error Messages

The following table lists messages for errors that do not cause immediate termination. When DUPENTRY executes the DUPMDISK command to copy a minidisk from the template user ID to the user ID being created, the 0323E message is generated when DUPMDISK returns an error indication. The 0323E message is preceded by messages from the DUPMDISK command. If there are additional minidisks to duplicate, DUPENTRY will continue attempting minidisk duplication.

Note: For more information about the problem with a particular minidisk, see the DUPMDISK command (see page 136).

DUPENTRY Command: Error Messages (No Immediate Termination)

Message Number	Message Text
0323E	ERROR return code EXECUTING DUPMDISK vaddr

DUPENTRY Command: Error Messages (Immediate Termination)

The following table lists return codes and messages for errors that cause immediate termination.

Return Code	Message Number	Message Text
4	0038E	MISSING PARAMETER
6	0531E	PARAMETER parm IS TOO LONG
8	0265E	NOT AUTHORIZED FOR: authorization words
12	0080E	USER userid DOES NOT EXIST
13	0707E	SPECIAL ASSIGN AUTHORIZATION IS REQUIRED TO USE THE MANAGER OPTION
15	0328E	INVALID USERID userid
17	0127E	user|terminal requestor EXCEEDED user|terminal requestor VMXLOGON PASSWORD LIMIT
21	0364E	FILE filename filetype filemode IS BEING UPDATED.
22	0358E	USER filename ALREADY EXISTS
23	0041E	USER userid IS NOT A DIRECTORY MANAGER
25	0690E	A USERID VALUE IS REQUIRED FOLLOWING THE MANAGER OPTION
26	0358E	USER filename ALREADY EXISTS ON HOLD
30	0643E	EXTRA PARAMETERS parameters WERE ENTERED AND ARE NOT ALLOWED
31	0463E	MANAGER mgrid's USERID LIMIT OF nnnnnn HAS BEEN REACHED
54	0633E	ONE OR MORE ERRORS WERE DETECTED WHILE PROCESSING THE UPDATE OF USERID userid
58	0363E	USER userid REJECTED BY INSTALLATION EXIT ROUTINE

Return Code	Message Number	Message Text
60	0341E	NEWUSER EXIT REJECTED USER userid AND ACCOUNT account
62	0689E	NEWUSER EXIT REJECTED GROUPNAME group FOR USER userid
70	0644E	'NOWAIT' OPTION SPECIFIED BUT THE SERVANT FACILITY IS NOT ACTIVE
71	0648E	THE NOWAIT OPTION MAY NOT BE SPECIFIED WITH THE STACK OR FILE OPTION
76	0018E	INVALID OPTION option

DUPENTRY Authorization Definition

Authorization to the DUPENTRY command is defined through the GRANT and WITHHOLD AUTHORIZ CONFIG configuration file records.

DUPENTRY command authorization has two parameters, the template user ID and the duplicated user ID.

The two parameters follow the specifications documented in the *Administration Guide*, in the Authorizations chapter. Information for the GRANT and WITHHOLD configuration file records can be found in the Configuration File Reference (see page 351).

A user ID can use DUPENTRY only if they have authorization for DUPENTRY, DUPMDISK, ADDMDISK and NOFORMAT. To use the MANAGER option, the user ID must also have ASSIGN authorization.

Example

- Allow user ID CLONER to duplicate new user IDs specifying any template user ID and any duplicated user ID.

  grant dupentry to cloner

- Allow user ID CLONER2 to duplicate any new user ID but only from the LINMAST2 template user ID.

  grant dupentry linmast2 * to cloner2

- Allow user ID CLONER3 to duplicate new user IDs that begin with 'LIN3' from template user IDs that begin with 'LIN3M'.

  grant dupentry lin3m* lin3* to cloner3

- Prevent the group GENUSER from issuing the DUPENTRY command.

  withhold dupentry from genuser (group

DUPMDISK Command

Use the DUPMDISK command to create an exact duplicate of a specified minidisk.

To create a duplicate minidisk:

```
DUPMdisk “Source” “Target” ["Parms"] [("Options"]
```

```
Source:
```

sourceentry sourcedisk

```
Target:
```

targetentry targetdisk

```
Parms:

"Mdisk Parms" | "Devno Parms"

Mdisk Parms:

 {subpool | *MNGR | *}
[{volser|=|*}
[{loc|=|*}
[{mode|=|W}
[{rpw|=}
[{wpw|=}
[mpw|=]]]]]]

Devno Parms:

 DEVNO rdev
[*
[{mode|=|W}
[{rpw|=}
[{wpw|=}
[mpw|=]]]]]
```

```
Options:

[FILE [filename]]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[Wait | NOWait]
```

Note: Options can be specified in any order.

Authorizations

You must have DUPMDISK authorization through a GRANT DUPMDISK record in the AUTHORIZ CONFIG file.

To use the DUPMDISK command on user IDs other than your own:

- For the *sourceuser*, you must have a GRANT DUPMDISK OVER *userid* TO *userid* record in the AUTHORIZ CONFIG file.
- For *targetuser*, you must have a GRANT ADDMDISK OVER *userid* TO *userid* record in the AUTHORIZ CONFIG file.

Note: For more information about Authorizations for selective User Ids, see the section, Granting Authorizations to Use Commands on Only Some User IDs in the chapter "Authorizations" of the *Administration Guide.*

Either you or the directory manager that will own the user ID must be authorized to allocate from the subpool in which you specify the minidisk to be duplicated. It is not required that both you and the owning manager be authorized. Alternately, either one of you can have ANYPOOL authorization, which allows you to allocate space from any subpool.

Definitions

sourceentry

Specifies the USER, IDENTITY, or SUBCONFIG directory entry defining the minidisk to be used as the source. The directory entry must not be in HOLD status.

sourcedisk

Specifies the virtual address of the minidisk to be used as the source.

targetentry

Specifies the USER, IDENTITY, or SUBCONFIG directory entry that owns the new duplicate (target) minidisk. The directory entry must not be in HOLD status.

targetdisk

Specifies the virtual address of the new duplicate minidisk.

subpool

Specifies the subpool from which to allocate space for the duplicate (target) minidisk or the subpool list to search for space to allocate for the duplicate minidisk. The default is *MNGR, which searches the list of subpools in which the directory manager for this user ID can allocate space for the first subpool with enough space.

If you specify the subpool by name, that subpool must be defined in the DASD CONFIG file and included in your or the owning manager's list of subpools in the VMDIRECT MANAGERS file. ANYPOOL authorization overrides this restriction. The following predefined lists can be searched for an available subpool. The first subpool in the list with enough space to accommodate the duplicate minidisk is the one that is used:

Subpool List	Subpools Represented
*MINE	All subpools in which you, or the person submitting this DUPMDISK command, can allocate space
*MNGR	All subpools in which the directory manager for this user ID can allocate space
*BOTH	Those in the *MNGR and *MINE lists, in that order
*ANY	Any subpool defined to CA VM:Director. **Note:** Your user ID must have ANYPOOL authorization.

CA VM:Director searches subpools in a list according to the order in which the MANAGER record in the VMDIRECT MANAGERS file specifies them. It does not include in the search any subpool that is defined with the EXPLICIT attribute.

volser

Specifies the volume serial number (volser) of the DASD volume on which to allocate the duplicate minidisk. The volume must be defined in the DASD CONFIG file and the subpool (specified on the *subpool* parameter) must be defined on one of the volume extents.

By default, CA VM:Director determines *volser* based on the location of the allocation. If you specify *volser* as an equal sign, CA VM:Director allocates the new minidisk from the same volume as the original one. However, if you specify a *volser*, you must also specify a location (*loc*).

loc

Specifies the location (starting cylinder or block) at which to allocate the duplicate minidisk on a given volume. This location can be any integer up to seven digits for number of cylinders or blocks.

If you specify the location, you must specify the *volser* parameter as a valid volume name or an equal sign. If you specify an equal sign, CA VM:Director uses the starting location of the original minidisk, presumably on a different volume but of the same device type. If the volume name is unchanged, the allocation of the new minidisk fails, because the original minidisk still occupies that space.

By default, CA VM:Director allocates the minidisk according to the algorithms defined for the applicable subpool.

Specifying an equal sign for both *volser* and *loc* causes the allocation of the new minidisk to fail, because the new minidisk still occupies that space.

DEVNO

Creates a permanent fullpack minidisk defined by a real device address. CA VM:Director does not control this type of minidisk and does not map the internal minidisk structure. You must specify * for the *loc* parameter to use the subsequent parameters.

rdev

Specifies the real device address of the volume on which to define the minidisk. No validation of this address is done until the user logs on and CP verifies the presence of the minidisk. The *rdev* parameter is used in the position of the *size* parameter in other versions of the parameter list. If you do not specify an *rdev* value or if you specify an *, CA VM:Director uses the virtual device address specified for *targetdisk*.

mode

Specifies the read/write mode for the duplicate minidisk when the user logs on. CP defines the valid modes, which are R, RR, W, WR, M, MR, and MW, each with an optional suffix of V, S, E, VS, VE, D, VD, SD, ED, VSD, and VED. The default mode is W.

rpw wpw mpw

Specifies the read, write, and multiple link passwords, respectively, for the duplicate minidisk.

Each password can be one to eight characters in length. These parameters do not have default values, and you cannot specify an asterisk for them. If one of the passwords is not specified, CA VM:Director places no value on the generated MDISK statement for that password and does not allow anyone to link to that minidisk with the respective link mode.

(CA VM:Secure only) If PEF is installed on your system, CA VM:Director encrypts the passwords.

[<u>Wait</u> | NOWait]

[<u>MSG</u> | NOMSG]

[STACK [<u>FIFO</u> | LIFO]]

[FIFO]

[LIFO]

[FILE [*filename*]]

These options are explained in the Common API Command Options (see page 36).

Description

The DUPMDISK command lets you create an exact duplicate of a specified existing minidisk, or copy the contents of a source minidisk to a matching target minidisk. The resulting minidisk will be an exact duplicate of the input minidisk, having the same size, device type, label, and other attributes.

If any allocation parameters (*subpool, volser,* or *loc*) are specified, then a new target minidisk will be created. An error will occur if the target minidisk already exists.

If no allocation parameters are specified, and the target minidisk does not exist, then default values for the allocation parameters will be used to create the target minidisk.

If no allocation parameters are specified, and the target minidisk exists, then the data will be copied from the source minidisk to the target. An error will occur if the source and target minidisk device type and size do not match.

While the command can be used to duplicate a CMS minidisk, its real benefit is in cloning Linux images on VM. By first making a master image and then using the DUPMDISK command to duplicate the directory entry and the minidisks, you can quickly set up multiple identical Linux images.

When a CA VM:Director macro calls the DUPMDISK command, it stacks a record for the calling macro's use when it successfully duplicates the minidisk. The stacked record is in last in, first out (LIFO) order and contains all the information necessary to build an MDISK record to describe the duplicated minidisk's location. The stacked record has the following format:

```
vaddr devtype start size volser linkmode rpw wpw mpw
```

The *start* and *size* values are in blocks for FBA and cylinders for CKD devices.

(CA VM:Secure only) If PEF is installed, passwords are encrypted.

Examples

- To make an exact copy of BOBBY 291 to the MAIN 10 minidisk with read/write access, enter:

  vmdirect dupmdisk bobby 291 main 10 * * * mr

- To make an exact copy of BOBBY 291 to the MAIN 10 minidisk with read/write access, allocating from the default subpool of BOBBY's directory manager, enter:

  vmdirect dupmdisk bobby 291 main 10 *mngr * * mr

- To make an exact copy of BOBBY 291 to the MAIN 10 minidisk on volume vmdsk1 at cylinder 15 with read/write access, enter:

  vmdirect dupmdisk bobby 291 main 10 * vmdsk1 15 mr

Return Codes and Error Messages

The following table lists return codes and error messages for the DUPMDISK command:

DUPMDISK Command: Return Codes and Error Messages

Return Code	Message Number	Text
2	0127E	USER *userid1* EXCEEDED USER *userid2* type PASSWORD LIMIT [FOR vaddr]
4	0038E	MISSING PARAMETER
6	0531E	PARAMETER *userid* IS TOO LONG
8	0265E	NOT AUTHORIZED FOR *authorization words*
10	0349E	PASSWORD IS INCORRECT
12	0080E	USER *userid* DOES NOT EXIST
14	0632E	MANAGER *mgrid* WAS NOT FOUND IN THE MANAGERS FILE
16	0040E	TOTAL ALLOCATION LIMIT OF *number* EXCEEDED
18	0632E	MANAGER *mgrid* WAS NOT FOUND IN THE MANAGERS FILE
20	0382E	USER *userid* IS ON HOLD
22	0621E	UNEXPECTED RETURN CODE *code* FROM OPEN

Return Code	Message Number	Text
23	0304E	ADDRESS *vaddr* IS AN INVALID DEVICE ADDRESS
25	0531E	PARAMETER *vaddr* IS TOO LONG
26	0313E	DEVICE *device* DOES NOT EXIST
27	0549E	DEVICE *device* ALREADY EXISTS
28	0634E	DIRECTORY PROFILE LINKS ALREADY EXIST FOR DEVICE *device*
30	none	*non-zero return code from* DIRECT_PRFLINKS
32	0635E	YOU ARE NOT A MANAGER AND HAVE NO SUBPOOLS THAT YOU ARE AUTHORIZED TO USE
33	0636E	ANYPOOL AUTHORIZATION IS REQUIRED TO ALLOCATE FROM SUBPOOL *subpool*
34	0637E	THE MINIDISK SIZE *size* MUST BE AN INTEGER NO LONGER THAN 7 DIGITS
35	0043E	SUBPOOL *subpool* IS UNDEFINED
36	0638E	A MINIDISK SIZE OF 0 IS NOT ALLOWED
37	0831E	*Userid1 vaddr1* AND *userid2 vaddr2* MUST HAVE MATCHING SIZE AND DEVICE TYPE.
38	0547E	INVALID DEVICE TYPE *devtype* (not found in table)
39	0119E	VOLUME ‘*volser*’ IS NOT A *devtype* DEVICE
40	0547E	INVALID DEVICE TYPE *devtype* (not CKD or FBA type)
41	0639E	YOU ARE NOT AUTHORIZED TO ADD A CACHE’d DEVICE
42	0640E	THE *parm* PARAMETER MAY NOT BE SPECIFIED FOR A MINIDISK ON DEVNO
44	0531E	PARAMETER *volser* IS TOO LONG
52	0637E	THE MINIDISK SIZE *size* MUST BE AN INTEGER NO LONGER THAN 7 DIGITS
54	0574E	CANNOT SPECIFY MINIDISK LOCATION WITHOUT VOLSER
58	0640E	THE MODE PARAMETER MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK | V-DISK
60	0642E	THE SPECIFIED LINK MODE *mode* IS INVALID

Return Code	Message Number	Text
61	0531E	PARAMETER *password* IS TOO LONG
62	0640E	THE PASSWORD PARAMETER MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK | V-DISK
65	0625E	TOO MANY SERVANTS IN USE, COMMAND CANNOT BE EXECUTED
66	none	*non-zero return code from* DIRECT_QALLOC
67 (CA VM:Secure only)	none	*non-zero return code from* PENCRYPT
68	0643E	EXTRA PARAMETERS *parameters* WERE ENTERED AND ARE NOT ALLOWED
69	0586E	ERROR STARTING SERVANT
70	0644E	NOWAIT OPTION SPECIFIED BUT THE SERVANT FACILITY IS NOT ACTIVE
71	0648E	THE NOWAIT OPTION MAY NOT BE SPECIFIED WITH THE STACK OR FILE OPTIONS
76	0018E	INVALID OPTION *option*
77	0698E	THE *option* OPTION MAY NOT BE SPECIFIED WITH THE *option* OPTION
78	0647E	THE WAIT AND FORMAT OPTIONS MAY NOT BE SPECIFIED FOR A MINIDISK ON T-DISK
79	0076S	USER ERROR '*nnn*' COPYING FILE '*fileid*'
80	0364E	FILE *filename filetype* IS BEING UPDATED
81	0220E	INVALID *option* OPTION OPERAND *parameter*
82	0617E	DDR FAILED WITH RC= *code* WHILE ATTEMPTING TO MOVE THE *vaddr*1 DISK
88	none	*non-zero return code from* DIRECT_OBTAIN
90	0209E	BLOCKSIZE 800 IS NOT SUPPORTED ON FB-512 DEVICES
91	0209E	BLOCKSIZE 800 IS NOT SUPPORTED ON FB-512 DEVICES
95	none	*non-zero return code from* DIRECT UPDATE
96	none	*non-zero return code from* MAN000S0
98	none	*non-zero return code from* MAN000R0

Return Code	Message Number	Text
103	0383I	FROM EXIT: *exit message*
1nnn	0200E	UNEXPECTED ERROR ‘*nnn*’ LINKING USER ‘*userid*’ MINIDISK ‘*vaddr*’
-231	0726E	SERVANT ABENDED DURING command COMMAND. DIRECTORY NOT CHANGED

Note:

- For explanations of the message routing and disconnect options, Common API Command Options (see page 36)
- For more information about the EXPLICIT attribute, see SUBPOOL Record (see page 394).
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide*.

EDIT Command

Use the EDIT command to open a USER, IDENTITY, or SUBCONFIG directory entry source for editing or to create a USER or IDENTITY directory entry.

```
EDIT entry [prototype]
```

Authorizations

You must have EDIT authorization through a GRANT EDIT record in the AUTHORIZ CONFIG file.

Definitions

entry

The name of the directory entry you want CA VM:Director to create or modify.

prototype

The name of a prototype directory entry on which to base the new directory entry when you are creating a user ID. The prototype is required when you are creating a directory entry, and not allowed when you are editing an existing directory entry.

Description

The EDIT command invokes your PROFILE XEDIT (if one exists) and allows you to modify an existing directory entry or to create a new directory entry for a user ID.

When you create a directory entry, you must use a prototype directory entry as the basis for the new one.

CA VM:Director copies the prototype directory entry to the new user ID name, then does the following to the copy (the new directory entry):

- Changes the user ID and logon password to match the new user ID
- Removes any MDISK statements from the new directory entry
- Resets any password history comments (*PW=) to today

If the directory entry includes an account comment (*AC=), change the account numbers it specifies to suit the new user ID.

When you edit an existing directory entry, you can change any statement in the entry except MDISK or BUILD statements, as long as the change results in a directory entry that meets VM standards for directory control statements. MDISK statements for controlled volumes may not be added. MDISK statements may be deleted which causes the minidisk to be deleted. BUILD statements may neither be added nor deleted using EDIT.

(CA VM:Secure only) If PEF is installed at your site, you also cannot change logon passwords (which are encrypted). No error messages are generated if you try to change a logon password, but CA VM:Director saves the encrypted logon password and restores it to its original value when you exit the directory entry

Changing special comment records starting with "*PW00" records can produce have unpredictable results, including making it impossible to log on. Changing these records must be only done under the direction of Technical Support.

Important! The EDIT command cannot perform DASD management. Do not use it to add, remove, or change MDISK records that are under CA VM:Director control in the DASD CONFIG file.

If the directory entry contains an INCLUDE statement (to reference the statements of a directory profile in the directory entry), the EDIT command displays this INCLUDE statement but does not display the contents of the included directory entry. You can delete the INCLUDE statement so that the directory entry no longer references the directory profile.

You can also change the POSIX-related statements: POSIXINFO, POSIXGLIST, and POSIXOPT.

Note: For syntax and usage information for the POSIXINFO, POSIXGLIST, and POSIXOPT directory control statements, see the IBM *CP Planning and Administration* guide for your release of VM.

If CA VM:Director detects syntax errors in the directory entry, it displays diagnostic messages and asks whether you want to fix the errors. If you choose not to correct the errors, the CP directory is not updated. If you enter the XEDIT SAVE command and then the QUIT or QQUIT command, the CP directory is updated.

You cannot issue the EDIT command from the CA VM:Director console.

Example

To edit the directory entry for user entry ERNIEP, enter:

`vmdirect edit erniep`

The directory entry for ERNIEP may look something like this:

```
USER ERNIEP RELIC 1M 3M 64
INCLUDE TESTPRO1
OPTION DEVINFO
*FL= N
ACCOUNT 58000099 POLO
*PW=
LINK TESTUSR1 191 19C RR
```

Note the INCLUDE statement, which references the directory profile TESTPRO1 but does not display the contents of this directory profile.

Note:

- For more information about displaying the contents of an included directory profile in a directory entry when editing the directory entry, see EDX Command (see page 148).
- For information about directory profiles and directory entries in CA VM:Director, see the chapter "Directory Profiles" in the *Administration Guide*.
- For a description of directory control statements, the IBM *CP Planning and Administration* guide.

EDX Command

Use the EDX command to open an existing USER or IDENTITY directory entry for editing or to create a directory entry, expanding to display the statements contained in the included directory profile.

```
EDX entry [prototype]
```

Authorizations

You must have EDIT authorization through a GRANT EDIT record in the AUTHORIZ CONFIG file.

Definitions

entry

The user ID whose directory entry you want CA VM:Director to create or modify.

prototype

The prototype directory entry on which to base the new directory entry if you are creating a user ID. The prototype is required if you are creating a directory entry and not allowed if you are editing an existing directory entry.

Description

The EDX command invokes the issuer's PROFILE XEDIT (if one exists) and performs the same function as the EDIT command except that it also displays and lets you edit statements from the included directory profile, if there is one. This is in addition to displaying the INCLUDE statement itself.

You also change the POSIX-related statements: POSIXINFO, POSIXGLIST, and POSIXOPT. For syntax and usage information for the POSIXINFO, POSIXGLIST, and POSIXOPT directory control statements, see the IBM *CP Planning and Administration* guide for your release of VM.

If you edit the statements that come from the directory profile, the updated directory profile statements are copied to the directory entry so that the changes are local; the directory profile remains unchanged.

If you delete the INCLUDE statement from a directory entry, the EDX command inserts all statements from the directory profile in the directory entry, and the user ID no longer references the directory profile. Any changes you make to the statements that came from the directory profile are now local to this directory entry only.

Example

To edit the directory entry for user ID ERNIEP, making the statements from the directory entry's INCLUDE statement available, enter:

vmdirect edx erniep

All statements that do not conflict are displayed as follows:

```
USER ERNIEP RELIC 1M 3M 64
INCLUDE TESTPRO1
OPTION SETORIG
*FL= N
ACCOUNT 58000099 POLO
*PW=

IPL CMS PARM AUTOCR
CONSOLE 009 3215
SPOOL 00C 2540 READER*
SPOOL 00D 2540 PUNCH A
SPOOL 00E 1403 A
LINK CMS4 190 190 RR
LINK YDISK 19E 19E RR
LINK YDISK 1BF 1BF RR

LINK TESTUSR1 191 19C RR
```

The statements in bold come from the included directory profile. These are the only statements that differ from the example shown for the EDIT command.

Note:

- For more information about displaying a directory entry without the contents of an included directory profile, see EDIT Command (see page 145).
- For more information about directory profiles and directory entries in CA VM:Director, see the chapter "Directory Profiles" in the *Administration Guide*.
- For a description of directory control statements, the IBM *CP Planning and Administration* guide.

END Command

Use the END command to cause normal termination of CA VM:Director system operation.

```
END [FORCE]
```

Authorizations

You must have END or END FORCE authorization through a GRANT END FORCE record in the AUTHORIZ CONFIG file to be able to terminate CA VM:Director immediately.

To allow a user to terminate CA VM:Director, only after all current processes complete, give that user END NOFORCE authorization through a GRANT record in the AUTHORIZ CONFIG file.

Definitions

FORCE

Terminates CA VM:Director immediately, without waiting for active processes to complete.

Description

The END command causes normal termination of CA VM:Director system operation, either immediately or after all currently running processes complete. If it waits for current processes to complete before terminating CA VM:Director, it prevents other processes from starting.

Forcing an immediate termination of CA VM:Director can cause unpredictable results if END terminates it during active processes that modify minidisk definitions. For example, the minidisk may have been in the middle of being moved and the user may have temporary links to the new disk space. If you are using the Servant Facility when this happens, these results are automatically cleared during the next initialization of CA VM:Director. If you are not using the Servant Facility, you can clear up these results by having the user ID that was running the interrupted process log off and log back on.

Examples

1. You need to terminate CA VM:Director to complete some maintenance work during the next hour. Enter the END command from the CA VM:Director console:

   end

CA VM:Director displays the following messages:

```
11:03:16 VMDIRECT 09 VMXCMD1110I Enter: END
11:03:16 VMDIRECT 09 VMXCMD1111I Beginning command: END
11:03:18 VMDIRECT 09 VMXSER0568I Servant VMX$0001 has been deleted.
11:03:18 VMDIRECT 09 VMXCMD1112I Ending command SERVANT with completion code 0.
USER DSC LOGOFF  AS   VMX$0001      USERS=4
11:03:25 VMDIRECT 09 VMXEND1103I CA VM:Director ended: 19Dec06 11:03:25.
11:03:25 VMDIRECT  09 VMXEND1175I Maximum virtual storage used was 397K out of 725K
bytes.
```

2. You need to terminate CA VM:Director. End user DOROTHY is in the middle of the USER command. Enter the END command from the CA VM:Director console:

   end

   CA VM:Director displays the following messages:

```
16:36:15 VMDIRECT 41 VMXCMD1110I Enter: END
16:36:15 VMDIRECT 41 VMXCMD1111I Beginning command: END
16:36:16 VMDIRECT 41 VMXLCK1196I Waiting for DOROTHY SS2 (USER) to release a shared
lock on resource: VMNUC-MACRO-VMDIRECT-USER.
After user DOROTHY completes the USER command, the following messages display:
16:41:21 DOROTHY 40 VMXCMD1112I Ending command USER with completion code 0.
16:41:21 VMDIRECT 41 VMXEND1103ICA CA VM:Director ended: 11Dec06 16:41:21.
16:41:22 VMDIRECT 41 VMXEND1175I Maximum virtual storage used was 1.1M out of 40M
bytes.
```

ENROLL Command

Use the ENROLL command to enroll a user ID in an SFS file pool.

```
ENROLL userid [“Parms”] [(“Options”]

Parms:

{blocks | *} [{storgrp | *} [filepool|*]]

Options:

[FILE [filename]]
[IPLstmt]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

Authorizations

You must be the SFS manager for the user ID and you must be authorized to allocate space in the affected file pools, or you must be an SFS administrator.

Definitions

userid

Specifies the user ID to enroll. The user ID must be active.

[*blocks* [*storgrp* [*filepool*]]]

Specifies the number of 4K blocks to allocate to the user ID, the user storage group from which to allocate this space, and the file pool from which to allocate the space. The user storage group and file pool must already be defined to CA VM:Director. The defaults for *blocks*, *storgrp*, and *filepool* are your default enrollment values as specified in the CA VM:Director configuration for SFS.

[IPLstmt]

Updates the IPL statement in the user ID's directory entry to make this file pool the user ID's default one (active when a user logs on to this user ID). If a sample profile is available or specified on a *SP= special comment in the directory entry, that profile is copied to the top-level directory for the user as PROFILE EXEC. If there is already a PROFILE EXEC file in the user's top-level directory, a sample profile specified on a *SP= special comment is not copied.

[FILE [*filename*]]

[<u>MSG</u> | NOMSG]

[STACK [<u>FIFO</u> | LIFO]]

[FIFO]

[LIFO]

These options are explained in Common API Command Options (see page 36).

Description

The ENROLL command enrolls an active user ID in an SFS file pool that is managed by CA VM:Director.

You can use your enrollment default values (unless you are the SFS administrator, in which case you must specify all values required) for the file pool, the user storage group, and the allocation size for this user ID, or you can specify values. If you specify a file pool from which to allocate space to this user ID, it must be one from which you are authorized to allocate file space through CA VM:Director; if you specify a user storage group, it must also be one from which you are authorized to allocate space.

When you allocate space, CA VM:Director checks that the amount you are trying to allocate does not exceed your allocation limit for this user storage group or the allocation limit for the user storage group itself. If either limit will be violated by this allocation, the command fails. If no limit is violated by your allocation, that amount of space is allocated to the user ID.

Return Codes and Error Messages

The table below lists return codes and error messages for the ENROLL command.

Return Code	Message Number	Text
99	0099I	*command* COMMAND CANCELED
328	0328E	INVALID USERID *userid*
7008	7008E	INVALID KEY *keyname*

Return Code	Message Number	Text
7015	7015I	userid ENROLLED IN FILE POOL filepoolid
7021	7021E	IPL RECORD UPDATED FOR *userid*
7028	7028E	USERID REQUIRED
7116	7116E	NO DEFAULT STORAGE GROUP AVAILABLE
7246	7246E	NO DEFAULT BLOCKS AVAILABLE

Note:

- For explanations of the message routing and disconnect options, see Common API Command Options (see page 36).
- For more information about checking your default allocation values (the QUERY SFS command), see QUERY Command (see page 303).
- For more information about SFS allocation limits for SFS managers and for user storage groups, see the chapter "SFS Overview" in the *Administration Guide*.
- For more information about special comment, see *SP= Special Comment in the chapter "Special Comment Reference" in the *Administration Guide*.
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide*.

ENTRY Command Reference

To update or query a single attribute or statement in any type of directory entry, issue the ENTRY command. The ENTRY command provides subcommands for processing almost all of the valid directory statements. The subcommands correspond to the IBM CP user directory control statements that they affect. The following entries provide the detailed usage for each valid subcommand.

ENTRY Command

ACCOUNT Subcommand

Use the ENTRY ACCOUNT command to add or remove account code values from a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname Account {"Parms"} [( "Options"]


Parms:

  acctnum1 [acctnumn] ...
| DELETE {* | acctnum1 [acctnumn] ...}
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

acctnum1

Defines the primary account number. An account number can be from 1 to 8 characters long.

acctnumn

Defines up to seven alternate account numbers.

DELETE

Removes the specified account numbers from the entry or removes all account numbers if ‘*’ is specified.

?

Returns all account code numbers from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

A maximum of eight account numbers may be defined in the entry. For details on the ACCOUNT statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

APPCPASS Subcommand

Use the ENTRY APPCPASS command to add or remove or replace an APPCPASS statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname APPCpass {"Parms"} [( "Options"]


Parms:

  gate_lu gate_known_lu userid {password | DELETE}
| ?


Options:

[MSG | NOMSG]
[FILE [filename]]
[“Stack Options”]
[STEM stemname]


Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the matching statement from the entry.

?

Returns all APPCPASS statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on an APPCPASS statement. For details on the APPCPASS statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

AUTOLOG Subcommand

Use the ENTRY AUTOLOG command to add or remove autolog user IDs from a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname AUTOlog {"Parms"} [( "Options"]


Parms:

  userid1 [useridn] ...
| DELETE {* | userid1 [useridn] ...}
| ?


Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]


Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

useridn

Specifies the names of up the user IDs allowed to AUTOLOG the entry user ID.

DELETE

Removes the specified autolog user IDs from the entry or removes all autolog user IDs if '*' is specified.

?

Returns all autolog user IDs from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the AUTOLOG statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

CLASS Subcommand

Use the ENTRY CLASS command to define or remove account the string of privilege class values from a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname CLass {"Parms"} [( "Options"]


Parms:

  privilegeclasses
| DELETE
| ?


Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]


Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

privilegeclasses

Specifies the class letter string consisting of up to 32 characters The valid characters are a through Z and 1 through 6.

DELETE

Removes the privilege classes from the directory entry.

*****

Removes the privilege classes from the directory entry.

?

Returns the privilege classes from the directory entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the privilege classes on a CLASS, USER, or IDENTITY statement, refer to the definitions in the sections on those statements in the IBM *CP Planning and Administration* publication for your release of VM.

COMMAND Subcommand

Use the ENTRY COMMAND command to add or remove a COMMAND statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname COMMAND "Parms" [( "Options"]

Parms:

ordinal { cp_cmd | DELETE | ?}

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

ordinal

Specifies an integer number defining the position of a COMMAND statement relative to other COMMAND statements in the directory entry. The first COMMAND statement in an entry has ordinal value ‘1’.

cp_cmd

Specifies the CP command line to be inserted or deleted from the directory entry. Adding a COMMAND statement with ordinal value ‘1’ makes it the first COMMAND statement in the entry. Adding a COMMAND statement with ordinal value ‘3’ inserts it after the COMMAND statement with ordinal value ‘2’.

DELETE

Removes the COMMAND statement identified by its ordinal number.

?

Returns all COMMAND statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the COMMAND statement in a directory entry refer to the definitions in the section about that statement type in the IBM *CP Planning and Administration* publication for your release of VM. COMMAND and CMD are treated as identical statement types.

CONSOLE Subcommand

Use the ENTRY CONSOLE command to add or remove or replace a CONSOLE statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname Console {"Parms"} [( "Options"]

Parms:

vdev {3215 | 3270} [class [secuser [OBServer]]]
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the matching statement from the entry.

?

Returns the CONSOLE statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a CONSOLE statement. For details on the CONSOLE statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

CPU Subcommand

Use the ENTRY CPU command to add or remove or replace a CPU statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname CPU cpuaddr {"Parms"} [( "Options"]

Parms:

  [BASE] [CPUID bbbbbb] [NODEDicate | DEDicate]
| DELETE
| ?


Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
```

| STACK [FIFO | LIFO]

Definitions

entryname

Specifies the name of the directory entry to process.

cpuaddr

Specifies the hexadecimal CPU address.

DELETE

Removes the matching statement from the entry.

?

Returns all CPU statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a CPU statement. For details on the CPU statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

ENTRY Command

CRYPTO Subcommand

Use the ENTRY CRYPTO command to add or remove or replace a CRYPTO statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname CRYPto {"Parms"} [( "Options"]

Parms:

  "First Crypto Statement Parms"
| "Subsequent Crypto Statement Parms"
| DELETE
| ?

First Crypto Statement Parms:

  APVIRTual
| DOMAIN domain1 [domainn] ... [APDEDicated aps1 [apsn] ...]

Subsequent Crypto Statement Parms:

  DOMAIN domain1 [domainn] ... [APDEDicated aps1 [apsn] ...]


Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes all CRYPTO statements from the entry.

?

Returns all CRYPTO statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a CRYPTO statement. For details on the CRYPTO statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

DATEFORMAT Subcommand

Use the ENTRY DATEFORMAT command to add or remove or replace a DATEFORMAT statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname DATEFormat {"Parms"} [( "Options"]

Parms:

  SHOrtdate
| FULldate
| ISOdate
| SYSdefault
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the DATEFORMAT statement from the entry.

?

Returns the DATEFORMAT statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a DATEFORMAT statement. For details on the DATEFORMAT statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

DEDICATE Subcommand

Use the ENTRY DEDICATE command to add or remove or replace a DEDICATE statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname Dedicate vdev {"Parms"} [( "Options"]

Parms:

  "Dasd Parms" [R/O] [NOQIOASSIST] [USERACCESSED userid]
| "Tape Parms"
| "Other Parms"
| DELETE
| ?

Dasd Parms:

  rdev [[VOLID] volid]
| [[VOLID] volid]

Tape Parms:

rdev [R/O] "Tape User Parms"

Tape User Parms:

  [SINGLEUSER] [NOASSIGN] [NOQIOASSIST]
|  MULTUSER    [NOASSIGN] [NOQIOASSIST]

Other Parms:

rdev [NOQIOASSIST] [USERACCESSED userid]


Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

vdev

Uniquely identifies the virtual device address.

DELETE

Removes the matching statement from the entry.

?

Returns the matching statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a DEDICATE statement. For details on the DEDICATE statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

DISTRIB Subcommand

Use the ENTRY DISTRIB command to add or remove the distribution code value from a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname DISTrib {"Parms"} [( "Options"]

Parms:

  distcode
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

distcode

Specifies the value of the distribution code.

DELETE

Removes the distribution code from the entry.

?

Returns the distribution code value from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the distribution code value on an ACCOUNT statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

D8ONECMD Subcommand

Use the ENTRY D8ONECMD command to add or remove or replace a D8ONECMD statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname D8ONECMD {"Parms"} [( "Options"]

Parms:

  {FAIL | LOG | OFF} [UNLOCK | LOCK]
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the D8ONECMD statement from the entry.

?

Returns the D8ONECMD statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a D8ONECMD statement. For details on the D8ONECMD statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

IOPRIORITY Subcommand

Use the ENTRY IOPRIORITY command to add or remove or replace the IOPRIORITY statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname IOPRIORity {"Parms"} [( "Options"]

Parms:

  {ABSolute | RELative} low [high]
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the IOPRIORITY statement from the entry.

?

Returns the IOPRIORITY statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on an IOPRIORITY statement. For details on the IOPRIORITY statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

IPL Subcommand

Use the ENTRY IPL command to add or remove or replace the IPL statement in a USER, IDENTITY, SUBCONFIG or PROFILE directory entry.

```
ENTRY entryname Ipl {"Parms"} [( "Options"]

Parms:

  {namesys | vdev} [LOADParm load_parameters] [PARM parameters]
| fcp_vdev [LOADParm load_parameters]
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the IPL statement from the entry.

?

Returns the IPL statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on an IPL statement. For details on the IPL statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

IUCV Subcommand

Use the ENTRY IUCV command to add or remove or replace an IUCV statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname IUCV {"Parms"} [( "Options"]

Parms:

  {"Iucv Type"} [Priority] [Msglimit limit]
| *ASYNCMD [Msglimit limit]
| *IDENT "Ident Parms"
| DELETE
| ?

Iucv Type:

  ALLOW
| ANY
| gatewayid
| resourceid
| userid
| *ACCOUNT
| *BLOCKIO
| *CCS
| *CONFIG
| *CRM
| *LOGREC
| *MONITOR
| *MSG
| *MSGALL
| *RPI
| *SCLP
| *SIGNAL
| *SPL
| *SYMPTOM
| *VMEVENT
| *VSWITCH

Ident Parms:

  GATEWAY GATEWAY
| name {LOCAL | GLOBAL | GATEWAY}
| RESANY {LOCAL | GLOBAL}

Options:

[MSG | NOMSG]
```

```
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the matching statement from the entry.

?

Returns the matching statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on an IUCV statement. For details on the IUCV statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

LINK Subcommand

Use the ENTRY LINK command to add or remove or replace a LINK statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname Link "Parms" [( "Options"]

Parms:

  owner owner_vdev vdev [mode | DELETE | ?]

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the matching statement from the entry.

?

Returns the matching statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a LINK statement. For details on the LINK statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

LOADDEV Subcommand

Use the ENTRY LOADDEV command to add or remove or replace a LOADDEV statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname LOADDEV {"Parms"} [( "Options"]

Parms:

  BOOTprog {bootprog_number  | DELETE | ?}
| BR_LBA   {hhhhhhhhhhhhhhhh | DELETE | ?}
| LUN      {hhhhhhhhhhhhhhhh | DELETE | ?}
| PORTname {hhhhhhhhhhhhhhhh | DELETE | ?
| SCPdata {'text' | HEX 'hh..hh' | DELETE | ?}

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the matching statement from the entry.

?

Returns the matching statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a LOADDEV statement. For details on the LOADDEV statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

LOGONBY Subcommand

Use the ENTRY LOGONBY command to add or remove logonby user ID values from a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname LOGONBY {"Parms"} [( "Options"]

Parms:

  userid1 [useridn] ...
| DELETE [userid1 [useridn] ... | *]
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

useridn

Specifies the names of the user IDs allowed to logon to the entry user ID using the BY option on the LOGON command.

DELETE

Removes the specified logonby user IDs from the entry. If '*' is selected, DELETE removes all logonby user IDs.

?

Returns all logonby user IDs from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the LOGONBY statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

MACHINE Subcommand

Use the ENTRY MACHINE command to add or remove or replace a MACHINE statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname MACHine {"Parms"} [( "Options"]

Parms:

  {ESA | XA | XC} [maxcpu]
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the MACHINE statement from the entry.

?

Returns the value of the MACHINE statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a MACHINE statement. For details on the MACHINE statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

MAXSTORAGE Subcommand

Use the ENTRY MAXSTORAGE command to add or remove or replace the value for the maximum allowed virtual storage size. The value may occur on either the USER or IDENTITY statement, or on a STORAGE statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname MAXSTORage {"Parms"} [( "Options"]

Parms:

  nnnnnnnK
| nnnnM
| nnnnG
| nnnnT
| nnnnP
| nnnnE
| *
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the maximum storage size value from the entry.

?

Returns the value of the maximum storage limit from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the MAXSTORAGE statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

NAMESAVE Subcommand

Use the ENTRY NAMESAVE command to add or remove restricted segment names from a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname NAMESAVE {"Parms"} [( "Options"]

Parms:

  segment1 [segmentn] ...
| DELETE [segment1 [segment] ... | *]
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

useridn

Specifies the names of the restricted segments the entry is allowed to use.

DELETE

Removes the specified segment names from the entry. If '*' is specified, DELETE removes all segment names.

?

Returns all segment names from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the NAMESAVE statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

NICDEF Subcommand

Use the ENTRY NICDEF command to add or remove or replace a NICDEF statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname NICDEF vdev {"Parms"} [( "Options"]


Parms:

  TYPE {HIPERsockets | QDIO | IEDN | INMN} "Nicdef Options"
| DELETE
| ?

Nicdef Options:


[DEVices 3 | DEVices devs]
[LAN {{ownerid | *} lanname | SYSTEM vswitch}]
[CHPID hh]
[MACID macaddress]

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the matching statement from the entry.

?

Returns the matching statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a NICDEF statement. For details on the NICDEF statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

NOPDATA Subcommand

Use the ENTRY NOPDATA command to add or remove or replace a NOPDATA statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname NOPDATA ["Parms"] [( "Options"]

Parms:

  DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the NOPDATA statement from the entry.

?

Returns the NOPDATA statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the NOPDATA statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

OPTION Subcommand

Use the ENTRY OPTION command to add or remove or replace option values from a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname Option {"Parms"} [( "Options"]

Parms:

  option1 [value1] [optionn [valuen]] ...
| DELETE option1 [optionn] ...
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

optionn

Specifies the name of an option keyword to be added to or removed from the directory entry.

valuen

Specifies an option keyword value required for the following keywords: CHPIDVIRTUALIZATION, LANG, MAXCONN, or MAXVMCFI.

DELETE

Removes the specified options from the directory entry. At least one option keyword must be specified.

?

Returns all option values from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36). For details on the OPTION statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

POSIXGLIST Subcommand

Use the ENTRY POSIXGLIST command to add or remove or replace a POSIXGLIST statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname POSIXGLIST {"Parms"} [( "Options"]


Parms:
  GIDS   [DELETE] gid ...
| GNAMES [DELETE] gnames ...
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes all POSIXGLIST statements, or selected individual GNAME or GID values, from the entry.

?

Returns the values of all POSIXGLIST statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the POSIXGLIST statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

POSIXINFO Subcommand

Use the ENTRY POSIXINFO command to add or remove or replace a POSIXINFO statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname POSIXINFO {"Parms"} [( "Options"]

Parms:

  UID    [uid    | DELETE]
| GID    [gid    | DELETE]
| GNAMES [gname  | DELETE]
| FSROOT [fsroot | DELETE]
| IUPGM  [iupgm  | DELETE]
| IWDIR  [iwdir  | DELETE]
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes all POSIXINFO statements, or selected individual POSIXINFO values, from the entry.

?

Returns the values of all POSIXINFO statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the POSIXINFO statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

POSIXOPT Subcommand

Use the ENTRY POSIXOPT command to add or remove or replace a POSIXOPT statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname POSIXOPT {"Parms"} [( "Options"]

Parms:

  EXEC_SETIDS [ALLOW | DISALLOW | SYSDEFAULT | DELETE]
| QUERYDB     [ALLOW | DISALLOW | SYSDEFAULT | DELETE]
| SETIDS      [ALLOW | DISALLOW | DELETE]
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

When used without other parms, DELETE removes all POSIXOPT statements from the entry. When used with another parm, DELETE removes the POSIXOPT statements of the type that the other parm specifies.

?

Returns the values of all POSIXOPT statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the POSIXOPT statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

REVIEW Subcommand

Use the ENTRY REVIEW command to retrieve the contents of a directory entry. REVIEW processes any USER, IDENTITY, SUBCONFIG, or PROFILE directory entry. It passes results back to the caller in a message, on the program stack, or in a REXX stem.

```
ENTRY entryname REVIEW [ALL] [( "Options"]


Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

ALL

Returns all of the directory entries that make up a virtual machine definition. All of the contents of the entry PROFILE, USER, IDENTITY, and SUBCONFIG contents are returned in that order.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

SCREEN Subcommand

Use the ENTRY SCREEN command to add or remove or replace a SCREEN statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname SCReen {"Parms"} [( "Options"]

Parms:

  "Area Parms" ...
| DELETE ["Area"]
| ?      ["Area"]

Area Parms:

"Area" {"Hilight" ["Color"] | "Color" ["Hilight"]}

Area:

  ALL
| INArea
| STAtarea
| OUTarea
| CPOut
| VMOut
| INRedisp

Hilight:

  BLInk
| NONe
| REVvideo
| UNDerlin

Color:

  BLUe
| GREen
| PINk
| RED
| TURquois
| WHIte
| YELlow
| DEFault

Options:

[MSG | NOMSG]
[FILE [filename]]
```

```
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes all SCREEN statements from the entry or removes a selected SCREEN area.

?

Returns all SCREEN statements from the entry, or returns a selected SCREEN area. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a SCREEN statement. For details on the SCREEN statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

SHARE Subcommand

Use the ENTRY SHARE command to add or remove or replace the statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname SHARE {"Parms"} [( "Options"]


Parms:

  {ABSolute y% | RELative z} ["Share Parms"]
| DELETE
| ?

Share Parms:

  NOLimit
| LIMITSoft
| LIMITHard
| {ABSolute a% | RELative b} [LIMITSoft | LIMITHard]

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes all SHARE statements from the entry.

?

Returns the value of all the SHARE statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a SHARE statement. For details on the SHARE statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

SPECIAL Subcommand

Use the ENTRY SPECIAL command to add or remove or replace a SPECIAL statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname SPEcial vdev {"Parms"} [( "Options"]

Parms:

  3270
| {2701 | 2702 | 2703} [Ibm | Tele]
| {CTCA | 3088 | BCTC | SCTC | CNC | FCTC} [* | userid]
| HIPERs [devs [{ownerid | *} lanname]]
| QDIO [devs [{ownerid | *} lanname | SYSTEM switchname]]
| MSGPROC userid [n]
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

vdev

Uniquely identifies the virtual device address.

DELETE

Removes the matching statement from the entry.

?

Returns the matching statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a SPECIAL statement. For details on the SPECIAL statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

SPOOL Subcommand

Use the ENTRY SPOOL command to add or remove or replace a SPOOL statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname Spool vdev {"Parms"} [( "Options"]

Parms:

  devtype [class ["Spool Parms"]]
| DELETE
| ?

Spool Parms:

width length
[4WCGM | 2WCGM]
[CFS | BTS]
[NODATACK | DATACK]

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

vdev

Uniquely identifies the virtual device address.

DELETE

Removes the matching statement from the entry.

?

Returns the matching statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a SPOOL statement. For details on the SPOOL statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

SPOOLFILE Subcommand

Use the ENTRY SPOOLFILE command to add or remove or replace a SPOOLFILE statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname SPOOLFile {"Parms"} [( "Options"]

Parms:

  MAXSPOOL nnnn
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

vdev

Uniquely identifies the virtual device address.

DELETE

Removes the SPOOLFILE statement from the entry.

?

Returns the value of the SPOOLFILE statement from the entry. The value is passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on a SPOOLFILE statement. For details on the SPOOLFILE statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

STDEVOPT Subcommand

Use the ENTRY STDEVOPT command to add or remove or replace an STDEVOPT statement in a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname STDEvopt {"Parms"} [( "Options"]

Parms:

  {"Stdevopt Parms"}
| DELETE
| ?

Stdevopt Parms:

  "Dasdsys Parms"
| "Library Parms"
| "Dasdsys Parms" "Library Parms"
| "Library Parms" "Dasdsys Parms"

Dasdsys Parms:

{DASDSYS {NODATAMOVER | NOCONCOPY} | DASDSYS {DATAMOVER | CONCOPY}}

Library Parms:

{LIBRARY NOCTL | LIBRARY CTL}

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

vdev

Uniquely identifies the virtual device address.

DELETE

Removes all STDEVOPT matching statements from the entry.

?

Returns all STDEVOPT statements from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

The remaining operands are the same as those on an STDEVOPT statement. For details on the STDEVOPT statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

STORAGE Subcommand

Use the ENTRY STORAGE command to add or remove or replace the value for the default virtual storage size. The value may occur on either the USER or IDENTITY statement or on a STORAGE statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname STORage {"Parms"} [( "Options"]

Parms:

  nnnnnnnK
| nnnnM
| nnnnG
| nnnnT
| nnnnP
| nnnnE
| *
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the default storage size value from the entry.

?

Returns the value of the default storage size from the entry. The value is passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the STORAGE statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

VMRELOCATE Subcommand

Use the ENTRY VMRELOCATE command to add or remove or replace a VMRELOCATE statement in a USER or PROFILE directory entry.

```
ENTRY entryname VMRELOCATE {"Parms"} [( "Options"]

Parms:

  [ON] DOMAIN {domain_name | SSI}
|  ON  [DOMAIN SSI | DOMAIN domain_name]
| OFF  [DOMAIN {domain_name | SSI}
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the VMRELOCATE statement from the entry.

?

Returns the VMRELOCATE statement from the entry. The value is passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the VMRELOCATE statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

XAUTOLOG Subcommand

Use the ENTRY XAUTOLOG command to add or remove autolog user IDs from a USER, IDENTITY, or PROFILE directory entry.

```
ENTRY entryname XAUTOlog {"Parms"} [( "Options"]

Parms:

  userid1 [useridn] ...
| DELETE {* | userid1 [useridn] ...}
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

useridn

Specifies the names of the user IDs allowed to XAUTOLOG the entry user ID.

DELETE

Removes the specified autolog user IDs from the entry. If '*' is specified, DELETE removes all autolog user IDs.

?

Returns all autolog user IDs from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the XAUTOLOG statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

XCONFIG Subcommand

Use the ENTRY XCONFIG command to add or remove or replace an XCONFIG statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname XCONfig {"Parms"} [( "Options"]

Parms:

  ACCEsslist {ALSIZE alecount | DELETE | ?}
| ADDRspace {"Addrspace Parms" | DELETE | ?}
| ?

Addrspace Parms:

MAXNUMBER nnnn
TOTSIZE {nnnnnM | nnnnG}
[NOSHARE | SHARE]

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the matching statement from the entry.

?

Returns the value of all XCONFIG statements or of the matching XCONFIG statement from the entry. The value is passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the XCONFIG statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

XSTORE Subcommand

Use the ENTRY XSTORE command to add or remove or replace the XSTORE statement in a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

```
ENTRY entryname XSTORE ["Parms"] [( "Options"]

Parms:

  ALL
| size
| DELETE
| ?

Options:

[MSG | NOMSG]
[FILE [filename]]
["Stack Options"]
[STEM stemname]

Stack Options:

  FIFO
| LIFO
| STACK [FIFO | LIFO]
```

Definitions

entryname

Specifies the name of the directory entry to process.

DELETE

Removes the XSTORE statement from the entry.

?

Returns the XSTORE statement from the entry. The values are passed to the caller in a message, on the program stack, or in a REXX stem.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[STEM *stemname*]

These options are explained in Common API Command Options (see page 36).

For details on the XSTORE statement, refer to the definitions in the section on that statement in the IBM *CP Planning and Administration* publication for your release of VM.

EXPIRE Command

Use the EXPIRE command to expire logon passwords.

```
EXPIRE userid
```

Definition

userid

The user ID whose logon password is to be expired.

Description

The EXPIRE command expires a logon password for a specified user ID. The next time a user ID with an expired logon password logs on and CA VM:Director is active, CA VM:Director prompts the user ID to change the password. User IDs that are autologged are not affected until the machine is logged on in a connected state.

(CA VM:Secure only) If the Rules Facility is installed, the EXPIRE command updates the user ID's directory entry by changing the flag special comment (*FL=) to indicate that the logon password is expired. An automatic expiration feature is also available when the Rules Facility is implemented. This method does not mark the directory entry as expired.

If you are using CA VM:Director or running CA VM:Secure without the Rules Facility, the IPLDISK record is required in the PRODUCT CONFIG file in order to use the EXPIRE command. The EXPIRE command updates the user's directory entry by adding an expire special comment (*EX=) and a directory link to the IPLDISK minidisk, and modifying the user's IPL statement. The added special comment provides the date the EXPIRE command was issued and the user ID that issued the command.

If you have user IDs that you do not want to expire on a regular basis, remove the password history special comment (*PW=) from their directory entries. (User IDs whose directory entries do not contain the *PW= special comment are not listed in the CMS EXEC file created by a QUERY PASSWORD command.) If you need to run the VMXGNR utility, specify the NOPW option so *PW= special comments are not inserted in all directory entries.

Example

To expire a group of user IDs at one time, use the QUERY command to create a CMS EXEC that lists all user IDs whose logon passwords have not changed in the last 30 days and whose logon passwords have not expired:

```
vmdirect query password 30 (exec
```

To expire those user IDs, enter:

```
cms vmdirect expire
```

The system responds, listing the name of each user ID and then listing when that user ID is actually expired.

Note:

- For information about changing logon passwords to NOLOG, so that the user ID is inactive and inaccessible, see NOLOG Command (see page 286).
- For information about querying old passwords or other monitored functions, see QUERY Command (see page 303).
- For information about running VMXGNR, see VMXGNR Utility (see page 452).
- (CA VM:Secure only) For more information about the automatic expiration of logon passwords, see Automating Password Expiration in the *Rules Facility Guide*.

EXTRACT Command

Use the EXTRACT command to extract information from USER, IDENTITY, and SUBCONFIG directory entries.

```
EXTRACT ["Parms" [HOLD]]
```

```
Parms:
```

{*filename* | EXTRACT} [{*filetype* | DIRECT} [{*filemode* | A1}]]

Definitions

[*filename* [*filetype*[*filemode*]]]

Specifies the filename, filetype, and filemode of the file that will contain the extracted information. The default is EXTRACT DIRECT A1. Creates a file that contains those directory entries that are not on hold.

[HOLD]

Creates a file that contains only those directory entries on hold.

Description

The EXTRACT command creates a condensed source directory file, which includes information extracted from directory entries. This file can be used by utilities such as the CA VM:Account VMJSCD utility.

The output file created by the EXTRACT command contains the following statement types:

- USER
- IDENTITY
- SUBCONFIG
- ACCOUNT
- DEDICATE
- MDISK
- ACIGROUP (CA VM:Secure only)

Any USER, IDENTITY, or MDISK statements in the EXTRACT file appear with logon passwords masked and minidisk passwords eliminated.

To extract information for all user IDs (those on HOLD and those not on HOLD), you must issue the EXTRACT command twice.

If a file of the specified name (or the default file if no name is specified) already exists on your A-disk, it is replaced by the file created by this command.

You must have a read/write A-disk to run this command.

GENINCL Command

Use the GENINCL command to include or remove a directory profile from a USER or IDENTITY directory entry.

```
GENINCL userid [profile]
```

Definitions

userid

Specifies the user directory entry in which a directory profile is to be added or removed.

profile

Is the name of the directory profile to include in the directory entry.

The profile name is required if you are including a directory profile in a user directory entry. It is omitted if you are removing a directory profile from a user directory entry.

Description

The GENINCL command adds an INCLUDE statement to the directory entry of the specified user ID or removes the INCLUDE statement from the specified user directory entry. If you use this command to add a directory profile to a user directory entry, this statement includes the name of the directory profile you specify in the GENINCL command.

A user directory entry contains only one directory profile at a time. If you add a directory profile to a user directory entry and the directory entry already references one, the GENINCL command replaces the INCLUDE statement currently in that user directory entry with the one specified by this GENINCL command. If the user directory entry did not already reference a directory profile, the GENINCL command simply adds an INCLUDE statement for the profile defined by this GENINCL command to that entry.

If you remove a directory profile from a user directory entry, the GENINCL command removes the INCLUDE statement from that entry.

If you try to make a change with the GENINCL command that will result in an invalid directory entry, GENINCL terminates with an error message, and the change is not made.

Examples

- To add a reference to the SALES directory profile in user ID CLIFFC's directory entry, enter:

  vmdirect genincl cliffc sales

- NORMP's directory entry currently refers to the SALES directory profile. To remove this reference, enter:

  vmdirect genincl normp

Note:

- For more information about incorporating statements from a user directory profile directly in the user directory entry, see EDX Command (see page 148).
- For more information about the INCLUDE statement, see the IBM *CP Planning and Administration* guide for your release of VM.

GETENTRY Command

Use the GETENTRY command to retrieve a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

To retrieve a directory entry:

GETENTRY *entry* [("Options"]

Options:

[DISK | RDR]
[FILE [*filename*]]
[MSG | NOMSG]
[NOLock | Lock]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[STEM *stemname* [.]]

Note: Options can be entered in any order

To get a directory profile:

GETENTRY *entry* (PROFILE "Options"

Options:

[DISK | RDR]
[FILE [*filename*]]
[MSG | NOMSG]
[NOLock | Lock]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[STEM *stemname* [.]]

Note: Options can be entered in any order

Definitions

entry

Specifies the name of the directory entry to retrieve.

***entry* (PROFile**

Specifies name of the directory profile to retrieve.

[DISK | RDR]

Copies the directory entry to your (DISK) or sends it to your reader (RDR). The name of the directory entry file on your disk or in your reader is the user ID name; the file type is DIRECT.

[<u>NOLock</u> | Lock]

Locks (LOCK) the directory entry or directory profile using the LOCK command or does not lock (NOLOCK) the directory entry or directory profile.

Locking the entry prevents further updates by other users until you issue the REPENTRY command or another authorized user issues the UNLOCK command.

If you specify the NOLOCK option, you cannot later update the directory entry with the REPENTRY command. Do not use the GETENTRY command with the NOLOCK option and then lock it later with the LOCK command, because the user ID may have been updated in the meantime.

[STEM *stemname*[.]]

Specifies the name of REXX stem variable array containing the statements retrieved from the directory entry contents. Any filetype or filemode parameters are ignored. The array contains the directory entry definition, so no disk file is referenced. The number of statements in the directory entry is defined in element *stemname.0, stemname.1* contains the first statement in the directory entry, and so on. The stem will be defined in the same REXX program which calls the GETENTRY command, after the command is called.

[FILE [*filename*]]

[<u>MSG</u> | NOMSG]

[STACK [<u>FIFO</u> | LIFO]]

[FIFO]

[LIFO]

These options are explained in Common API Command Options (see page 36).

Description

The GETENTRY command retrieves the current directory entry for the specified user ID or the specified directory profile. It places this file in your reader queue or on your A-disk, as you specify.

CA VM:Director protects logon and link passwords from observation by replacing them with a string of eight asterisks. If there are no link passwords, three tokens of eight asterisks appear in their place.

Return Codes and Error Messages

The following table lists the return codes and error messages for the GETENTRY command.

GETENTRY Command: Return Codes and Error Messages

Return Code	Message Number	Text
1	0127E	USER *userid1* EXCEEDED USER *userid2 type* PASSWORD LIMIT
2	0038E	MISSING PARAMETER
3	0531E	PARAMETER *parameter* IS TOO LONG
4	0643E	EXTRA PARAMETER(S) *parameters* WERE ENTERED AND ARE NOT ALLOWED
11	0265E	NOT AUTHORIZED FOR: *authorization words*
12	0349E	PASSWORD IS INCORRECT
13	0018E	INVALID OPTION *option*
14	0080E	USER *userid* DOES NOT EXIST
15	0382E	USER *userid* IS ON HOLD
16	0270W	PROFILE *profid* DOES NOT EXIST
17	0680E	*filename filetype* IS ALREADY LOCKED BY *userid*
18	0364E	FILE *filename filetype filemode* IS BEING UPDATED
21	0099I	’GETENTRY’ COMMAND CANCELED
22	0380E	A { READ | WRITE } A-DISK IS REQUIRED

Note:

- For explanations of the message routing and disconnect options, see Common API Command Options (see page 36).
- For information about locks created by the LOCK command, LOCK Command (see page 256).
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide.*

GETPWEXP Command

Use the GETPWEXP command to obtain information the date when a user LOGON password is set to expire.

```
GETPWEXP [userid] [("Options"]

Options:

[FILE [filename]]
[MSG | NOMSG ]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

Authorizations

You must have GETPWEXP authorization through a GRANT GETPWEXP record in the AUTHORIZ CONFIG file to be able to obtain password expiration information.

To use the GETPWEXP command on user IDs other than your own, you must have a GRANT GETPWEXP OVER *userid* TO *userid* record in the AUTHORIZ CONFIG file.

Note: For more information, see the section "Granting Authorizations to Use Commands on Only Some User IDs" in the chapter "Authorizations" of the *Administration Guide.*

Definitions

[*userid*]

Specifies the name of a USER or IDENTITY directory entry whose password expiration you want to see. If a user ID is not specified, the password expiration information is collected for the requesting user ID.

[FILE [*filename*]]

Appends the results of the command to the specified file. The default filename is GETPWEXP; the filetype is always RESULTS. The file is created if it does not already exist, but is appended to if it does exist.

[MSG | NOMSG]

Controls the output messages.

[STACK [<u>FIFO</u> | LIFO]]

[FIFO]

[LIFO]

Stacks the results in your program stack.

Note: For more information about the FILE, MSG|NOMSG, and STACK options, see Common API Command Options (see page 36).

Examples

- To view password information for your own user ID (CLIFFC), enter:

  `vmdirect getpwexp`

- CA VM:Director displays:

```
VMXGET0534I USERID CLIFFC PASSWORD WILL EXPIRE IN 19 DAYS on 15 AUG 2003
```

- To view password information for NORMP, enter:

  `vmdirect getpwexp normp`

  CA VM:Director displays:

```
VMXGET0536I USERID NORMP PASSWORD EXPIRED 309 DAYS AGO ON 15 AUG 2002
```

GRANT AUTHORITY Command

Use the GRANT AUTHORITY command to allow users to grant access to SFS directories and files owned by you or other users.

```
GRANT AUThority ["Parms"] TO “Target” [("Options"]
```

```
Parms:
```

[*filename filetype*] *dirid*

```
Target:
```

{*userid* | PUBLIC | *nickname*}

```
Options:
```

```
[READ | WRITE | NEWRead | NEWWrite | DIRRead | DIRWrite]
[TYPE | NOType]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

The CA VM:Director GRANT AUTHORITY command syntax is identical to the syntax of the CMS GRANT AUTHORITY command.

Note: For an explanation of all parameters and options for the GRANT AUTHORITY command, see the IBM CMS command reference documentation.

Authorizations

Enroll the CA VM:Director service virtual machine as an SFS administrator in the DMSPARMS configuration file of every file pool for which the GRANT AUTHORITY command will be issued. CA VM:Director does not need to manage the file pool.

You must have GRANT AUTHORITY authorization through a GRANT record in the CA VM:Director AUTHORIZ CONFIG file.

Description

The CA VM:Director GRANT AUTHORITY command allows you to give certain users, such as directory managers, access to files and directories in file spaces owned by other user IDs.

After verifying that the issuing user ID is authorized to perform the GRANT AUTHORITY command for the target user IDs (including all user IDs in a nickname list), CA VM:Director passes the GRANT AUTHORITY command to CMS.

CA VM:Director generates an audit record (1070) for each user ID specified on the command. If a nickname is specified, CA VM:Director generates an audit record for every user ID defined in the nickname list.

Nickname Resolution

When you specify a nickname for *userid*, CA VM:Director first uses information in your NAMES file to determine the local user. If there are user IDs that CA VM:Director cannot resolve using the NAMES file, it then uses the DMSJNE routine to resolve the remaining unresolved user IDs. DMSJNE is an optional customer-written routine that returns a local user ID for the supplied user ID and nodeid. If your site uses DMSJNE and it is not available, or if CA VM:Director cannot resolve any of the user IDs, the GRANT AUTHORITY command terminates with a return code of 328.

Examples

- To allow anyone to read the FONE LIST file in the VMSYSU:EMPLOY.INFO directory, enter:

  vmdirect grant authority fone list vmsysu:employ.info to public (read

- To allow anyone to read any file currently in the VMSYSU:ENGINEER.TOOLS directory and any files created in the future, enter:

  vmdirect grant authority vmsysu:engineer.tools to public (read newread

  vmdirect grant authority * * vmsysu:engineer.tools to public (read

- To allow FRAISERC to add new files to the VMSYSU:ENGINEER.TOOLS directory, enter:

  vmdirect grant authority vmsysu:engineer.tools to fraiserc (write

Return Codes and Error Messages

The table, GRANT AUTHORITY Command: Return Codes and Error Messages, lists return codes and error messages for the GRANT AUTHORITY command.

The GRANT AUTHORITY command may also generate CMS messages.

Note: For information about these messages, use the CMS help or see the IBM *CMS Commands and Utilities Reference* for your release of VM.

GRANT AUTHORITY Command: Return Codes and Error Messages

Return Code	Message Number	Text
24	1117E	GRANT IS NOT A CA VM:Director COMMAND
38	0038E	MISSING PARAMETER
39	0039E	INVALID PARAMETER *parameter*
76	0076S	USER ERROR *code* COPYING FILE *nickname_file*
99	0099I	GRANT COMMAND CANCELED
221	0221E	MISSING OPTION OPERAND
265	0265E	NOT AUTHORIZED FOR: GRANT *filepool userid*
328	0328E	INVALID USERID *userid*
543	0543E	THE SERVANT FACILITY IS NOT CURRENTLY TURNED ON
586	0586E	ERROR *code* STARTING SERVANT
625	0625E	TOO MANY SERVANTS IN USE, COMMAND CANNOT BE EXECUTED
7063	7063E	EXTRANEOUS PARAMETER(S) *parameters*
7065	7065E	FILE POOL *filepoolid* NOT RESPONDING TO REQUESTS

Note:

- For more information about how to remove or downgrade a previously granted authority, see REVOKE AUTHORITY Command (see page 322).
- For more information about the audit record produced by the GRANT AUTHORITY command, see AUDITEXT Command (see page 71).

- For more information about the GRANT AUTHORITY authorization, see the chapter "Authorizations" in the *Administration Guide*.
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide*.
- For detailed information about the parameters and options on the GRANT AUTHORITY command, see the IBM *CMS Commands and Utilities Reference*.
- For more information about coding the DMSJNE routine, see the IBM *CMS Filepool Planning, Administration, and Operation* guide.
- For more information about other CMS messages generated by the GRANT AUTHORITY command, see the IBM *CMS Commands and Utilities Reference*.

IPLDISKX Command

Use the IPLDISKX command to change directory entries to support the EXPIRE command and the Rules Facility.

```
IPLDISKX entry
```

Definition

entry

Specifies the USER or IDENTITY directory entry to modify. The directory entry must not be in held status.

Description

The IPLDISKX command invokes a conversion utility that changes directory entries to allow expiration of logon passwords through the EXPIRE command and the Rules Facility.

The conversion utility scans a user directory entry and removes the link to the IPLDISK minidisk, modifies the IPL statement, and replaces the expire special comment (*EX=) with a flag special comment (*FL=).

A copy of the modified directory entry is not written to the CA VM:Director backup minidisk (BKUP).

This command does not affect user IDs that do not have the expire statement, the directory link to the IPLDISK, and the modified IPL statement.

The IPLDISK record must be in the PRODUCT CONFIG file for you to use the IPLDISKX command. Do not remove the IPLDISK record from the PRODUCT CONFIG file until all user IDs have been converted to the method of expiration implemented with the Rules Facility.

Example

Create a CMS EXEC of all user IDs at your site and convert them to a method of expiration implemented with the Rules Facility:

```
vmdirect query users (exec
cms vmdirect ipldiskx
```

JOURNAL Command

Use the JOURNAL command to display or clear information about invalid password conditions for a specific user ID.

To list current journal information:

```
JOURNAL {LIST | LISTPSWD} [userid1 [command [userid2 [vaddr]]]]
```

To reset journal entries:

```
JOURNAL RESET userid1 command [userid2 [vaddr]]
```

Definitions

LIST

Displays current journal information. If you do not specify any variables, the command lists all journal information. If you specify any variables, the command lists only journal entries that contain the specified values, formatted to match the order of the specified variables. The display includes the number of times a particular activity occurred.

LISTPSWD

Displays the same journal information that the LIST parameter displays and invalid passwords if CA VM:Director was initialized with the AUDINV startup parameter. Password phrases are not displayed with this command.

RESET

Removes any journal entries that exactly match the specified variables.

userid1

Specifies a user ID for which to review or reset invalid passwords.

command

Specifies a command for which to check for the invalid password attempts, as follows:

Values for *command*	Checks for
VMXLINK	Invalid link passwords were entered from the User Selection Screen option Define a Link to Another User's Minidisk
VMXLOGON	Invalid logon passwords were entered while issuing CA VM:Director commands

userid2

Specifies the target user ID for *command*.

vaddr

Specifies the target virtual address for *command*.

Description

CA VM:Director creates journal entries when it detects a user ID issuing an invalid password. If the number of consecutive invalid attempts exceeds the limit specified on the JOURNAL record in the SECURITY CONFIG file, CA VM:Director prevents further attempts until the JOURNAL RESET command is issued. CA VM:Director automatically resets journal counters when the password is correctly entered.

The JOURNAL command only monitors invalid logon passwords entered while issuing CA VM:Director commands (VMXLOGON) and invalid link passwords entered from the User Selection screen option, Define a Link to Another User's Minidisk (VMXLINK).

Listing Journal Entries

To display the current journal information, enter the JOURNAL command with the LIST parameter:

```
vmdirect journal list
```

CA VM:Director responds:

```
1 DIANEC  VMXLOGON
4 DIANEC  VMXLINK VMSYS1  199
```

The output reflects the order of the variables in the command syntax, prefixed by the number of occurrences of each journal entry.

Clearing the Journal Count

The JOURNAL command with the RESET parameter clears the number of invalid password attempts recorded for a specific user ID. You enter the variables for the JOURNAL RESET command in the order shown by the JOURNAL LIST command, except for the omission of the number of times the specific event occurred.

To clear the invalid password attempts (VMXLOGON and VMXLINK) for the above example, enter the following commands:

```
vmdirect journal reset dianec vmxlogon
vmdirect journal reset dianec vmxlink vmsys1 199
```

To use the RESET command as another method of clearing and resetting journal information, see RESET Command (see page 320).

LISTAUTH Command

Use the LISTAUTH command to display all AUTHORIZ CONFIG records about a particular authority or a user ID.

```
LISTAUTH [userid [authword ...]]  [(TYPE | FIFO | LIFO]
```

Definitions

userid

Specifies the user ID whose authorizations are to be listed. You can specify a user ID by name if you are authorized to query that user ID, or any of the following variable user IDs:

Userid variable	Userids that LISTAUTH command finds
I	User ID under which you are entering the LISTAUTH command
ANYUSR	Any user, regardless of name (the first one the command finds)
SOMEUSR	Any user, regardless of name (the first one the command finds)
EVERYUSR	All users, regardless of name

The default user ID is I.

authword ...

Specifies a list of authorities to query. You can specify any authority, or any of the following authority variables:

Authority variable	Authorities that LISTAUTH command finds
ANYCMD	Any command or authority for the specified user ID or group
SOMECMD	Any command or authority for the specified user ID or group
EVERYCMD	All commands or authorities for the specified user ID or group
ANYWRD	Any command or authority for the specified user ID or group
SOMEWRD	Any command or authority for the specified user ID or group

Authority variable	Authorities that LISTAUTH command finds
EVERYWRD	All commands or authorities for the specified user ID or group

The default for each authority is EVERYWRD.

Note: The default authorization word for the LISTAUTH command is EVERYWRD, while the default authorization word for the MAY command is ANYWRD. These different default values can produce different results for the LISTAUTH and MAY commands, even though their functions are similar.

[TYPE | FIFO | LIFO]

Determines whether to display or stack the GRANT or WITHHOLD configuration file records, if any, that determine whether the use of the command, utility, or screen selection is permitted. TYPE displays the record. FIFO stacks the records in the response in first in, first out order. LIFO stacks the records in the response in last in, first out order.

Description

The LISTAUTH command lets you query the GRANT, LIST, and WITHHOLD records in the AUTHORIZ CONFIG file to determine whether a user ID has a specific authorization and responds by displaying the GRANT and WITHHOLD records that relate to the specified authorization.

The LISTAUTH command is very similar to the MAY command. Both respond to your queries by displaying the GRANT and WITHHOLD records that relate to the specified authorization, but the MAY command also responds by return code.

If you specify fewer authorities than are possible for a particular authorization, the LISTAUTH command treats your request as if you supplied a trailing EVERYWRD. For example, the following commands mean the same: "Can Helen issue all forms of the ADMIN command?":

vmdirect listauth helen admin

or

vmdirect listauth helen admin everywrd

CA VM:Director searches for authorizations that allow use of all forms of the ADMIN command, and responds with a message like the following:

```
ACCEPTED BY: GRANT ADMIN MANAGERS TO SUSAN HELEN JOHN
```

Examples:

- To determine the CA VM:Director authorizations JUNIOR has, enter:

  `vmdirect listauth junior`

- To display all authorizations that control the use of the CHANGE command, enter:

  `vmdirect listauth anyusr change`

- To find out if you authorized anyone to change SMITH's user ID:

  `vmdirect listauth anyusr change smith`

Note: For information about how to find the final disposition of a user ID's authorization for a command or function, see the MAY Command (see page 270).

LOCAL Command Prefix

Use LOCAL as a prefix to another command to direct that the CA VM:Director product server on the local node execute the command.

```
LOCAL command
```

Definitions

command

Specifies the command string to execute on the product server running on the local member node in an SSI complex.

Description

The LOCAL command prefix is used in a Single System Image environment to direct that certain member specific commands run on the local product server. In an SSI environment, commands issued using the product command MODULE (usually VMDIRECT MODULE or VMDIRECT MODULE) execute on the Master product server.

Note: For more information on SSI command processing, see “Command Processing in an SSI Complex” in the *Administration Guide*.

Use the LOCAL command prefix to retrieve audit data from a specific member product server by logging on to that member with an authorized user ID, and issuing the following command:

```
VMDIRECT LOCAL AUDITEXT
```

Use the LOCAL command prefix to convert an Agent server into the Master server during an outage by issuing the following command:

```
VMDIRECT LOCAL TAKEOVER
```

The following commands are supported for execution on an Agent server:

- ABEND
- ACITRACE
- AUDITEXT
- CMS
- DEBUG
- END
- MAP
- QCPCFG
- QLOCK
- QPCB
- QSTART
- REBUILD
- SYSWORD
- TAKEOVER
- TRACE
- VERSION

LOCK Command

Use the LOCK command to lock a USER or IDENTITY directory entry, a directory profile, or a CMS file.

```
LOCK {"Parms"} [("Options"]

Parms:

  FILE filename filetype filemode
| PROFILE profilename
| USERID userid

Options:

[FILE [filename ]]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

Definitions

FILE *filename filetype filemode*

Specifies the file to lock. Pattern-matching is not allowed in the file specification.

PROFILE *profilename*

Specifies the directory profile to lock.

USER *userid*

Specifies the user ID of the directory entry to lock.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]

[FIFO]

[LIFO]

These options are explained in Common API Command Options (see page 36).

Description

The LOCK command prevents updates to a user ID, directory profile, or CMS file. No one can update a locked item until you or another authorized user ID unlocks it using the UNLOCK command.

The LOCK command creates two permanent locks for each user ID, directory profile, or CMS file that you lock and writes them to a CMS file called PERMLOCK FILE A on the CA VM:Director A-disk. For user IDs and directory profiles, the locks have the following form:

SHARED DISK *filename filetype filemode*
PRIVATE LOCKUSER *userid*
PRIVATE LOCKUSER *profilename*

For CMS files, the form is as follows:

SHARED DISK *filename filetype filemode*
PRIVATE LOCKFILE *filename filetype filemode*

The LOCK command creates permanent locks that are maintained across CA VM:Director initializations. Other commands that manipulate user ID or profile entries create temporary locks that are not written to any file and do not last beyond the duration of the command.

CA VM:Director removes entries from the PERMLOCK FILE A file when you clear the lock. If this file exists at initialization, CA VM:Director reads it to re-establish any previously existing permanent locks.

When processing a user or profile entry, the LOCK and UNLOCK commands use the same internal locks as the GETENTRY and REPENTRY commands. Thus, you cannot distinguish between the locks from the LOCK command and the locks from those commands. Do not use the GETENTRY command with the NOLOCK option and lock it later with the LOCK command. The user ID may have been updated in the meantime.

Return Codes and Error Messages

The following table lists the return codes and error messages for the LOCK command.

Return Code	Message Number	Text
1	0127E	USER *userid1* EXCEEDED USER *userid2 type* PASSWORD LIMIT
2	0038E	MISSING PARAMETER
3	0531E	PARAMETER *parameter* IS TOO LONG

Return Code	Message Number	Text
4	0643E	EXTRA PARAMETERS *parameters* WERE ENTERED AND ARE NOT ALLOWED
10	0039E	INVALID PARAMETER *parameter*
11	0265E	NOT AUTHORIZED FOR: *authorization words*
12	0349E	PASSWORD IS INCORRECT
13	0018E	INVALID OPTION *option*
14	0080E	USER *userid* DOES NOT EXIST
15	0680E	{ USERID | PROFILE } *entryid* IS ALREADY LOCKED BY *userid*
16	0364E	FILE *filename filetype filemode* IS BEING UPDATED
17	0687E	YOU MUST USE LOCK USER OR LOCK PROFILE TO LOCK A USERID OR PROFILE
18	0682E	FILE *filename filetype filemode* HAS ALREADY BEEN LOCKED BY *userid*
19	0270W	PROFILE *profid* DOES NOT EXIST
20	0038E	MISSING PARAMETER
28	0681E	FILE *filename filetype filemode* DOES NOT EXIST

Note:

- For explanations of the message routing and disconnect options, see Common API Command Options (see page 36).
- For information about removing these locks, see UNLOCK Command (see page 345).
- For complete descriptions of all messages listed in this section, see *Message Reference Guide.*

MACLOAD Command

Use the MACLOAD command to load a macro to the CA VM:Director service virtual machine.

```
MACLOAD filename [filetype | VMDIRECT] [(LOAD]
```

Definitions

***filename* [*filetype*]**

Specifies the filename and filetype of the macro to copy. The default for *filetype* is VMDIRECT.

LOAD

Loads the macro in the CA VM:Director service virtual machine's memory after copying it.

Description

The MACLOAD command copies the specified CA VM:Director macro from your A-disk to the CA VM:Director service virtual machine's A-disk. It can also load the macro in the service virtual machine's memory using the CMS EXECLOAD command.

The MACLOAD command is intended for use by systems personnel after applying local service to CA VM:Director macros. In order for new macros to be usable by CA VM:Director, they must be on a disk that is accessible to the CA VM:Director service virtual machine. Macros with local updates are typically kept on the CA VM:Director service virtual machine's A-disk.

The macro maintenance utility, VMXMACUP, automatically issues the MACLOAD command for you. If the macro already exists on the CA VM:Director service virtual machine's A-disk, the new copy replaces the existing one.

Example

You have applied a local update to the MANAGE macro to resolve a problem or make a local modification. To copy the MANAGE VMDIRECT macro from your A-disk to the CA VM:Director service virtual machine's A-disk and load it in memory, enter:

```
vmdirect macload manage vmdirect (load
```

MAINT Command

Use the MAINT command to allow users to make directory changes for their own directory entries, and to allow directory managers to define, modify, and manage directory entries.

```
MAINT {"Parms"}

Parms:

  Account newacct [NEXTLOG | TEMP | IMMED]
| DEFine oldvaddr newvaddr
| Delete vaddr
| DIstrib newdist
| Help | ?
| Ipl [sysname | vaddr ] [PARM parameters]
| Link  ownerid vaddr1 vaddr2 [ mode]
| MACHINE [370 | XA | ESA | XC]
| Manage entry subfunction
| MDpw [vaddr]
| MGrid
| Minidisk vaddr
| Option attribute {ON | OFF}
| Password
| Review
| RLink vaddr
| SCratch vaddr
| Storage {sizeK | sizeM}
| Terminal symbolname {ON | OFF | char | hex}
```

Authorizations

To perform user functions on your own user ID, you must have MAINT *subfunction* authorization through a GRANT MAINT *subfunction* record in the AUTHORIZ CONFIG file, where *subfunction* is the specific subfunction you want to use.

To perform user functions on another user ID, you must have MAINTMAN *userid* USER *subfunction* authorization through a GRANT MAINTMAN *userid* USER *subfunction* record in the AUTHORIZ CONFIG file, where *subfunction* is the specific subfunction you want to use.

To perform management functions, you must have MAINTMAN *userid subfunction* authorization through a GRANT MAINTMAN *userid subfunction* record in the AUTHORIZ CONFIG file.

Definitions

[Account *newacct* [NEXTLOG | TEMP | IMMED]]

Changes your primary account number. If CA VM:Director is interfaced with VM:Account, CA VM:Director allows only immediate (IMMED) changes to account numbers. Otherwise, NEXTLOG is the default.

When issuing the MAINT MANAGE command to change a user's account number, CA VM:Director must have OPTION D84NOPAs in its directory entry if the directory has any user IDs that have the special password of LBYONLY. Failing to have this authorization can result in errors and the command outcome can be unpredictable.

[DEFine oldvaddr newvaddr]

Changes the virtual address of one of your minidisks from *oldvaddr* to *newvaddr*. CA VM:Director updates directory links to *oldvaddr* to reflect the new virtual address. Minidisk passwords for the minidisk remain unchanged.

[Delete *vaddr*]

Deletes the directory link to the minidisk at *vaddr*.

[DIstrib *newdist*]

Changes the distribution code associated with your spooled output to *newdist*.

[<u>Help</u> | ?]

Displays a brief list of the MAINT command functions and their operands. HELP is the default if you do not enter a function.

[Ipl [*sysname* | *vaddr*] [PARM *parameters*]]

Changes the entire content of your IPL statement. The variable *sysname* is the system to IPL at logon. The variable *vaddr* is the virtual device address of the device to IPL at logon. If no values follow IPL, CA VM:Director removes any existing IPL statement from your directory entry.

[Link *ownerid vaddr1 vaddr2* [*mode*]]

Defines a directory link to another user's minidisk. Directory links are performed for you at logon. To define a directory link, you must specify the minidisk password for the ownerid's minidisk (*vaddr1*) and specify a virtual address (*vaddr2*) not used by your virtual machine. You can specify the link mode to use through *mode*, or use the default mode "R".

[MACHINE [XA | ESA | XC]]

Switches your MACHINE architecture between the available options. The default is determined by the setting in the VMDIRECT GLOBALS file.

[Manage *entry subfunction*]

For directory managers; defines USER or IDENTITY directory entries or manages existing USER, IDENTITY, or SUBCONFIG entries. You can manage only one *entry* per session and cannot manage your own directory entry. When you use this parameter you are placed in a VM READ state, from which you can enter one of the MAINT MANAGE command subfunctions listed below.

MAINT MANAGE subfunction	Purpose
?	Displays the MAINT MANAGE subfunctions
ACTIVATE	Activates a USER entry that is on hold and enables any file spaces in all file pools that CA VM:Director controls
COMMENTS	Invokes XEDIT to edit comment records in a user directory entry
DISKMOVE [*vaddr*]	Moves a minidisk in a USER, IDENTITY, or SUBCONFIG directory entry to a new location; the default virtual address is 191
END	Terminates the MAINT MANAGE function and returns control to CMS
HOLD	Places an existing USER directory entry on hold and disables any user file spaces in all file pools that CA VM:Director controls
NEWDISK [*vaddr*]	Defines and optionally formats a new minidisk in a USER, IDENTITY, or SUBCONFIG directory entry; the default virtual address is 191
NEWUSER	Creates a USER or IDENTITY entry and optionally enrolls the user in file pools that CA VM:Director controls. You can identify a sample PROFILE EXEC file to copy to the 191 minidisk or top-level directory by placing the *SP= special comment in the skeleton or input file. If a sample profile is available or specified on a *SP= special comment, that profile is copied to the top-level directory for the user as PROFILE EXEC. If there is already a PROFILE EXEC file in the user’s top-level directory, a sample profile specified on a *SP= special comment is not copied.

MAINT MANAGE subfunction	Purpose
REMOVE	Removes a USER or IDENTITY entry from the directory and deletes any user file spaces from all file pools that CA VM:Director controls. With this subfunction, when the command deletes a minidisk that is on a pack that is under CA VM:Director control, CA VM:Director deletes links to the deleted that minidisk from the directory. An IDENTITY entry can only be removed if it has no BUILD statements.
REVIEW	Invokes XEDIT to display a directory entry
SCRATCH [*vaddr*]	Removes and formats a minidisk from a USER, IDENTITY, or SUBCONFIG directory entry; the default virtual address is 191. With this subfunction, when the command deletes a minidisk that is on a pack that is under CA VM:Director control, CA VM:Director deletes links to that deleted minidisk from the directory.
USER *parameters*	Invokes the MAINT subfunctions for the directory entry being managed by the MAINT MANAGE function

[MDpw [*vaddr*]]

Displays the passwords for your minidisks. If *vaddr* is not specified, the command displays passwords for all your minidisks. Passwords may be masked at your site.

[MGrid]

Displays the user ID of your directory manager.

[MInidisk *vaddr*]

Prompts you for a new link mode and read, write, or multiple passwords for your minidisk at virtual address *vaddr*. If you enter a null at a prompt, CA VM:Director sets all values that were not previously set to nulls. The values can include link mode, read password, write password, and multiple write password.

[Option *attribute* ON | OFF]

Selects (ON) or removes (OFF) various attributes, specified by *attribute*, associated with your user ID.

Supported attributes are:

- CONCEAL
- MIH
- TODENABLE
- CPUID
- NOVF

[Password]

Changes your logon password. If you are using traditional passwords, you may change your password to anything that does not include spaces, a left or right parenthesis, or one of the IBM reserved passwords.

[Review]

Creates a read-only XEDIT session of all directory control statements in your directory entry. It provides information you need for the DEFINE, DELETE, LINK, MINIDISK, MDPW, RLINK, and SCRATCH functions, such as existing directory links and all virtual addresses associated with your user ID.

[RLink *vaddr*]

Reviews and optionally removes any directory links that other users may have defined to your minidisk at virtual address *vaddr*. This function is particularly useful when you determine that data on your minidisk should no longer be shared with other users. Review any existing directory links to determine if the user still has access to your minidisk.

[SCratch *vaddr*]

Deletes your minidisk at virtual address *vaddr*. CA VM:Director automatically removes directory links to the minidisk being scratched from other users' directory entries.

[Storage *size*K | *size*M]

Changes the amount of virtual storage your user ID is set up with at logon. Virtual storage can be modified after logon with the CP DEFINE STORAGE command.

[Terminal *symbolname* { ON | OFF | *char* | *hex*]}

Selects or removes values for various line editing symbols associated with your user ID. ON with a symbol name selects the default character to use for that line editing symbol; a character (char) or a hexadecimal value (hex) with a symbol name uses that character or hexadecimal value for that line editing symbol. OFF with a symbol name removes that line editing symbol. Changeable line editing symbols are:

- CHARDEL
- LINEDEL
- ESCAPE
- LINEND

Description

Some forms of the MAINT command depend on line-mode interaction with the issuing user ID.

Valid replies to prompts are documented in *Message Reference Guide*.

If you prefer a full-screen interface for management functions, use the MANAGE command. If you want to use commands that do not require interaction with the issuing user ID, use the API commands.

Note:

- For a list of API commands, see Application Programming Interface Commands (see page 35).
- For more information about special comment, see *SP= Special Comment in the chapter "Special Comment Reference" of the *Administration Guide*.

MANAGE Command

Use the MANAGE command to display the Manager Selection Menu or the entry screens of one of the selections available from that menu.

```
MANage entry [number [vaddr]]
```

Definitions

***entry* [*number* [*vaddr*]]**

Specifies the USER, IDENTITY, or SUBCONFIG directory entry to process, the number of the selection on the Manager Selection Menu to use, and, for certain selections, the virtual address of a minidisk on which to use the selection. The following table lists the menu selections:

Number	Selection	Accept *vaddr*?
1	Define/Reactivate a Userid	—
2	Define and Format a New Minidisk	3
3	Select User Menu	—
4	Scratch a Minidisk	3
5	Hold or Remove a Userid	—
6	Move a Minidisk	3
7	Edit Directory Comments	—
8	Review the Directory Entry	—
9	Redefine a Minidisk in Place	3
10	Work with Shared File System Menu	—

If you do not specify a menu selection or a virtual address when it is needed, the Manager Selection Menu displays.

The default virtual address is 191.

Note: With selection 4, Scratch a Minidisk, and selection 5, Hold or Remove a Userid, when the command deletes a minidisk that is on a pack that is under CA VM:Director control, CA VM:Director deletes links to that deleted minidisk from the directory.

Description

The MANAGE command displays screens that let you define new USER or IDENTITY directory entries and modify existing USER, IDENTITY, or SUBCONFIG directory entries.

When you create or move a minidisk with the MANAGE command and do not specify a device type, the size you enter is interpreted as the number of 512-byte blocks.

Directory managers are not allowed to manage their own user IDs. Changes to a directory manager user ID must be made by the CA VM:Director system administrator or another authorized directory manager.

Use the QUERY command to determine the skeleton files and subpools you are allowed to use before you create or move minidisks or create user IDs.

If CA VM:Director manages your SFS and you use this command to define a new user ID, then all file pools that CA VM:Director manages must be active and available so that it can check them all for file space that may be owned by the same user ID name as the one you are defining. If any file pool managed by CA VM:Director is not available for checking, or if CA VM:Director finds any file space in an available file pool owned by the same user ID name, the command will fail.

For menu selection 1, Define/Reactivate a User ID, and menu selection 10, Work with Shared File System Menu, from which you answer Yes in the Update IPL record? field, you can identify a sample PROFILE EXEC to copy to the minidisk or top-level directory. To identify the sample PROFILE EXEC, place the special comment record, *SP=, in the skeleton or input file. If a sample profile is available or specified on a *SP= special comment in the directory entry, that profile is copied to the top-level directory for the user as PROFILE EXEC. If there is already a PROFILE EXEC file in the user’s top-level directory, a sample profile specified on a *SP= special comment is not copied.

Note:

- For more information about determining the skeleton files and subpools you are allowed to use, see QUERY Command (see page 303).
- For more information about the *SP= special comment record, see *SP= (Sample Profile) in chapter "Special Comment Reference" of the *Administration Guide.*
- For a detailed description of the MANAGE command screens, see the *Directory Manager’s Guide*

MAP Command

Use the MAP command to produce a map of the disk space use.

```
MAP [volume [ALL | ALLOC | EXTENT | FREE]] [("Options"]
```

```
Options:
```

```
[FILE filename ]
[POOLS]
[PRINT]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[TERM]
```

Note: Options can be entered in any order.

Definitions

volume

Specifies the volume serial number of the DASD to map. You can specify certain characters and a trailing asterisk to map all DASD that starts with the indicated characters. The default for *volume* is an asterisk, which tells the command to map all DASD identified to CA VM:Director.

[ALL | ALLOC | EXTENT | FREE]

Indicates the areas to map. ALL maps all minidisks, extents, and free disk space; ALLOC maps only minidisks; EXTENT maps only extents; FREE maps only free disk space.

[FILE *filename*]

Places the map in the specified file on your A-disk. The filetype of this file is MAP.

[POOLS]

Includes the subpools associated with each extent.

[PRINT]

Prints the map to your virtual printer.

[STACK [FIFO | LIFO]]

Stacks the map in your program stack in first-in, first-out (FIFO) or last in, first out (LIFO) order.

[TERM]

Displays the map on your terminal.

Description

This command maps only volumes that are defined in the DASD CONFIG file; it does not include ignored minidisks in the output. If you want to map all DASD volumes, including volumes that are outside of CA VM:Director control, do the following on a CA VM:Director administrator user ID, such as VMANAGER:

1. Run the following command:

   vmxbkp01 hold

2. Use the IBM DISKMAP command against the USER DIRECT file you created above.

Examples

- To map all DASD volumes the DASD CONFIG file defines, enter:

  vmdirect map *

- To map only the free space on volume VMPK02, enter:

  vmdirect map vmpk02 free

MAY Command

Use the MAY command to query the AUTHORIZ CONFIG file.

```
MAY [userid [authword ...]]
```

Definitions

userid

Specifies the user ID whose authorizations you want CA VM:Director to query. You can specify a user ID by name if you are authorized to query that user ID, or any of the variable user IDs described under the LISTAUTH command.

The default user ID is I.

authword ...

Specifies a list of authority words that describe the authorization to query. You can specify any authority, or any of the authority variables described under the LISTAUTH command.

The default for each authority word is ANYWRD.

Note: The default authority word for the MAY command is ANYWRD, while the default authority word for the LISTAUTH command is EVERYWRD. These different default values can produce different results for the LISTAUTH and MAY commands, even though their functions are similar.

Description

The MAY command lets you query the AUTHORIZ CONFIG file to determine whether a user ID has a specific authorization. This command always returns a code of 0 or 8. It does not stack any data. You can use this command in programs that check CA VM:Director authorizations.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	User ID is authorized to perform the function

Return Code	Meaning
8	User ID is not authorized to perform the function

Examples

- To determine whether JUNIOR is allowed to issue the CHGMDISK command for any user IDs, enter:

  `vmdirect may junior chgmdisk anyusr`

- To determine whether any user has been given authority to manage every other user ID, enter:

  `vmdirect may anyusr manage everyusr`

- To check whether you can use the EXPIRE command on user ID SMITH, enter:

  `vmdirect may i expire smith`

Note: For information about how to find all authorizations a user ID has for a command or function, see LISTAUTH Command (see page 252).

MDSKSCAN Command

Use the MDSKSCAN command to report on the minidisks allocated within specified directory entries.

```
MDSKSCAN [entry| *] [("Options"]

Options:

[FILE [filename ]]
[MANAGER {mgrid | *}]
[NOWAIT]
```

Note: Options can be entered in any order.

Authorizations

If the MDSKSCAN command is to be issued from an EXEC, the issuing user ID must have NOPASS MDSKSCAN authorization in the AUTHORIZ CONFIG file.

Definitions

[*entry* | *]

Specifies the USER, IDENTITY, or SUBCONFIG directory entry containing minidisks to scan. You can specify entry names with a trailing asterisk to scan minidisks belonging to all entries that start with the indicated characters. If you specify *, which is the default, the scan includes minidisks of all directory entries that you manage.

[FILE [*filename*]]

Places the report in a file with the specified filename and a filetype of SCAN on your A-disk. You must have a read/write A-disk if you use the FILE option. The default filename is MDSKSCAN.

If you use the NOWAIT option with the FILE option, the command sends the report to your reader.

[MANAGER {*mgrid* | *}]

Specifies the directory manager whose userids' minidisks to scan. An asterisk for *mgrid* scans minidisks of all directory managers and displays their userids.

[NOWAIT]

Does not wait while the minidisk scan is done. A servant user ID sends you a reader file when the scan completes. This option is valid only if the Servant Facility is enabled.

Description

The MDSKSCAN command reports on the minidisks allocated to specified user IDs. It reports only on minidisks allocated on volumes that are defined in the DASD CONFIG file.

This command provides information such as the virtual address of the minidisk, the minidisk type, block size, number of files, total size, percent of utilization, and date and time of last update. It does this by calling the VMXAQD MODULE to access and query each minidisk in a read-only mode.

The minidisk type can be any of the following:

Minidisk type	Description
CMS	Regular CMS minidisk
OS	OS-formatted minidisk
RES	Reserve-formatted minidisk
SFD	Minidisk that has had the SAVEFD command run against it
???	Unrecognized

If you are not using the Servant Facility, the VMXAQD MODULE must be accessible, preferably on a public minidisk, for MDSKSCAN to execute properly.

Example

To get a report of minidisks for user IDs that begin with TEST that are under all directory managers, enter the following:

```
vmdirect mdskscan test* (manager *
```

CA VM:Director produces a report similar to the one below:

```
USERID    VADR  TYPE BLKSZ  FILES  SIZE   %   DATE      TIME      MANAGER
TEST1     191   CMS  4096   1      300K   2   99/10/17  13:07:22  CARLAT
TEST2     191   CMS  4096   25     300K   80  99/10/17  13:07:26  CARLAT
TEST2     192   OS
TEST3     191   CMS  4096   8      1024K  4   99/10/17  13:07:30  WOODYB
TEST4     191   CMS  4096   15     300K   52  99/10/17  13:07:33  WOODYB
```

MODIFY Command

Use the MODIFY command to change user space allocation in any SFS file pool in which the user ID is already enrolled.

```
MODIFY userid [blocks [filepool]] [("Options"]

Options:

[FILE [filename]]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

Authorizations

You must be the SFS manager for the user ID and you must be authorized to allocate space in the affected file pools, or you must be an SFS administrator.

Definitions

userid

Specifies the user ID whose allocation limits to modify.

[*blocks* [*filepool*]]

Specifies the number of 4K blocks to allocate to the user ID in the indicated file pool. (A user ID can be allocated space from only one user storage group in a file pool. CA VM:Director allocates the space from the correct user storage group.)

The number of blocks can be either an absolute value (no leading sign) or a value that is relative to the current allocation (a leading + to increase the current allocation; a leading - to decrease the current allocation). An absolute value (for example, 800) changes the user allocation to that number of blocks. A relative value (for example, +500 or -500) increases or decreases the current allocation by that number of blocks.

The default for *filepool* is your default file pool as defined in CA VM:Director.

[FILE [*filename*]]

[<u>MSG</u> | NOMSG]

[STACK [<u>FIFO</u> | LIFO]]

[FIFO]

[LIFO]

These options are explained in Common API Command Options (see page 36).

Description

If you increase a user allocation, CA VM:Director checks that the allocation does not exceed your allocation limit in the affected user storage group, and that it does not exceed the user storage group's own allocation limit. If it does, the request fails. For instance, if a particular SFS manager can only allocate 200 more blocks in a certain user storage group and you try to increase an allocation of 500 blocks to 800 blocks, your request will fail. Similarly, if there are only 150 blocks remaining to be allocated in a user storage group and you try to increase an allocation of 500 blocks to 800 blocks, your request will fail.

If you decrease a user allocation, CA VM:Director checks that the user still has enough space for the current data. If not, the request fails. For instance, suppose a user allocation is 800 blocks and 650 of those blocks are in use. If you try to decrease the allocation to 600 blocks, the request will fail.

You cannot enroll a user ID in a file pool through this command. If you want to enroll a user ID in a file pool, use the ENROLL command.

Return Codes and Error Messages

The following table lists return codes and error messages for the MODIFY command.

MODIFY Command: Return Codes and Error Messages

The following table describes the return code:

Return Code	Message Number	Text
18	0018E	INVALID OPTION *option*
99	0099I	*command* COMMAND CANCELED
7003	7003E	SFS ERROR REASON CODE *code* DURING *operation* FOR *userid filepoolid*

Return Code	Message Number	Text
7012	7012E	*serverid* NOT AUTHORIZED IN FILE POOL *filepoolid*
7014	7014E	FILEPOOL NAME *filepoolid* is INVALID
7016	7016E	INVALID NUMBER OF BLOCKS *blocksreq*
7017	7017E	INVALID STORAGE GROUP NUMBER *sgreq*
7022	7022E	YOU ARE NOT AUTHORIZED TO MANAGE *userid* IN *filepoolid*
7023	7023E	INCOMPATIBLE ENTRIES *field1 field2*
7024	7024E	YOU ARE NOT AUTHORIZED TO MANAGE THIS USERID
7025	7025E	*userid* NOT YET ENROLLED in *filepoolid*
7026	7026E	NO CHANGE SPECIFIED FOR *userid*
7027	7027E	ERROR *nn* DURING MODIFY PROCESSING FOR *userid*
7065	7065E	FILE POOL *filepoolid* NOT RESPONDING TO REQUESTS
7066	7066E	*userid* NOT ENROLLED, LIMITS EXCEEDED FOR *filepoolid* STORAGE GROUP *nn*
7082	7082E	REQUIRED FIELD *fieldname* NOT PROVIDED
7083	7083E	FILESPACE ALLOCATION FOR *userid* NOW *xx* BLOCKS, *yy(zz*%) IN USE
7121	7121E	STORAGE GROUP MARKED ** IS THE DEFAULT FOR ENROLLMENT
7122	7122E	REMOVE AUTHORITY REQUEST CANCELLED
7131	7131E	YOU ARE TRYING TO DELETE TOO MUCH STORAGE FOR *userid*

Examples

- User ID PMASON has only 30 blocks of space available in file pool VMXA. To increase PMASON’s allocation by 500 blocks, enter:

  vmdirect modify pmason +500 vmxa

- To decrease user ID PDRAKE’s allocation in file pool PRODB by 350 blocks, enter:

  vmdirect modify pdrake -350 prodb

- To change user ID MARYR's allocation in file pool FINA to 2,000 blocks, regardless of her current allocation, enter:

  vmdirect modify maryr 2000 fina

Note:

- For explanations of the message routing and disconnect options, see Common API Command Options (see page 36).
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide*

MOVE2SFS Command

The MOVE2SFS command copies data from a minidisk to an SFS file space.

```
MOVE2SFS userid [vaddr] [("Options"]

Options:

[DELete]
[IPLstmt]
[POOL ["Poolparms"]]

Poolparms:

{filepool | *} [{storgroup | *} [blocks | ADDSPACE | USESPACE ]]
```

Note: Options can be entered in any order.

Authorizations

You must be the SFS manager for the user ID and you must be authorized to allocate space in the affected file pools, or you must be an SFS administrator.

Definitions

userid

Specifies the name of a user ID owning data you want to copy to SFS.

[*vaddr*]

Specifies the virtual address of a single minidisk containing data you want to copy to SFS. If you do not specify *vaddr*, CA VM:Director copies the data from all the user minidisks to SFS.

[DELete]

Deletes the user minidisks after copying their data. For minidisks on packs that are under CA VM:Director control, this command also deletes links to the deleted minidisks from the directory.

[IPLstmt]

Updates the user directory entry so that the target file pool is active at user ID logon. If a sample profile is available or specified on a *SP= special comment in the directory entry, that profile is copied to the top-level directory for the user as PROFILE EXEC. If there is already a PROFILE EXEC file in the user's top-level directory, a sample profile specified on a *SP= special comment is not copied.

[POOL [*filepool* [*storgroup* [*blocks* | ADDSPACE | USESPACE]]]]

Specifies the file pool and user storage group to which to copy the data, and indicates the space to use for the copied data.

The file pool and user storage group must already be defined to CA VM:Director. The defaults for *filepool*, *storgroup*, and *blocks* are the default enrollment values for the user's SFS manager as specified in the SFS configuration.

blocks specifies the number of 4K blocks to allocate to the user ID and in which to copy the data. If the user ID already has a file space, the number of *blocks* is added to the existing file space.

ADDSPACE tells CA VM:Director to add as much space as needed, in 4K blocks, to the user ID's file space to accommodate the data. If the user ID is not yet enrolled in SFS, ADDSPACE gives the user ID just enough space for the copied data.

USESPACE tells CA VM:Director to copy the data to a file space the user already has. If the data needs more space than is available or the user is not yet enrolled in SFS, the command fails.

If you specify POOL as the last option on the MOVE2SFS command, you can specify it without parameters and get the default enrollment values.

Description

The MOVE2SFS command lets you copy user data from minidisks to directories in an SFS file space.

The MOVE2SFS command creates one directory for the contents of each minidisk it copies. Default directory names are MDSK*vaddr*, where *vaddr* is the virtual address of the minidisk from which the data is copied. For example, data on a minidisk at virtual address 0536 is copied to SFS directory MDSK0536.

The MOVE2SFS command uses the PREMOVE and the POSTMOVE user exits in its processing. You can use the PREMOVE user exit to name these destination directories yourself if you have your own directory naming scheme. The POSTMOVE user exit manipulates data that has been moved from minidisks to SFS.

The MOVE2SFS command does not fully copy certain types of minidisks to SFS. The following table explains how MOVE2SFS proceeds when it detects each type of minidisk.

For this type of minidisk	MOVE2SFS does this
Reserved or SAVEFD	Sends you warning messages and does not copy the data from the minidisk to SFS
Recomputed	Sends you warning messages and copies only data from the CMS-formatted portion of the minidisk to SFS

Example

User NORMP has data on 191 and 192 minidisks. To move NORMP's data from the existing minidisks to subdirectories under the USERS:NORMP file space, and to enroll NORMP in the USERS file pool using storage group 2, enter:

```
vmdirect move2sfs normp ( ipl pool users 2 addspace
```

The data is moved to default directory names MDSK0191 and MDSK0192.

If the PREMOVE user exit is implemented, the directory name is provided by the user exit. You can code the PREMOVE user exit to recognize the 191 virtual address and to stack a null line for its directory name. This causes the top-level directory to be used instead of a subdirectory when copying the data from the 191 minidisk.

Note:

- For more information about POSTMOVE, see POSTMOVE User Exit (see page 435)
- For more information about PREMOVE, see PREMOVE User Exit (see page 437)
- For more information about special comment, see *SP= Special Comment in the chapter "Special Comment Reference" of the *Administration Guide.*

MULTIPLE Command

Use the MULTIPLE command to change the status of all directory entries defined in a control file.

```
MULTIPLE {"Parms"}

Parms:

  ACTIVATE filename filetype
| HOLD filename filetype
| NEWUSER filename filetype
| REMOVE filename filetype
```

Definitions

ACTIVATE

Moves the USER directory entries listed in the specified control file from the CA VM:Director HOLD minidisk to the CA VM:Director DRCT minidisk and the CP object directory. This move activates the user IDs.

HOLD

Moves the USER directory entries listed in the specified control file from the CA VM:Director DRCT minidisk to the CA VM:Director HOLD minidisk and removes them from the CP object directory. This move deactivates the user IDs, putting them on hold.

NEWUSER

Creates USER or IDENTITY directory entries using disk space allocation and format information in the specified control file. Through console prompts, you can specify an alternative skeleton file or subpool name. If in the specified skeleton file, there is a *ENROLL special comment with the IPLstmt option, and if a sample profile is available or specified on a *SP= special comment in the skeleton file, CA VM:Director copies that sample profile to the top-level directory for the user ID as PROFILE EXEC. If there is already a PROFILE EXEC file in the user's top-level directory, a sample profile specified on a *SP= special comment is not copied.

The new user's directory entry is not written to the BKUP minidisk when using the MULTIPLE command with the NEWUSER parameter.

REMOVE

Removes the USER or IDENTITY directory entries listed in the specified control file from the CA VM:Director DRCT minidisk and the CP object directory, and formats any minidisks in those entries. An IDENTITY entry can only be removed if it contains no BUILD statements.

Note: When CA VM:Director deletes a minidisk that is on a pack that is under its control, it deletes links to that deleted minidisk from the directory.

filename filetype

Is the CMS filename and filetype of the control file to use. The file can reside on any of your accessed minidisks.

Description

The MULTIPLE command uses a control file, which resides on a minidisk accessed by the directory manager, to add, delete, hold, or activate a group of virtual machines.

If CA VM:Director manages your SFS and you use this command to add new users, then all file pools that CA VM:Director manages must be active and available so that it can check them all for file space that may be owned by the same user ID names as the ones you are adding. If any file pool managed by CA VM:Director is not available for checking, or if CA VM:Director finds any file space in an available file pool owned by one of the same user ID names, the command will fail.

Control File Records

A control file is a regular CMS file that contains records that describe the specifications for each virtual machine. The records in the control file used with the MULTIPLE command have positional parameters. Tokens exceeding the valid number of characters are truncated to the maximum acceptable length. The general format of these records is:

`userid [{password|*} [{acct|*} [{distcode|*} [disksize [blocksize [label ]]]]]]`

userid

Specifies the user ID to activate, place on hold, create, or remove.

The remaining parameters are valid only with the NEWUSER parameter. The command ignores these parameters if you specify them with ACTIVATE, HOLD, or REMOVE.

[*password*]

Specifies the password for the new user ID. The default password is the same as the user ID. To use the default password, specify an asterisk. You cannot enter a password phrase in the control file. Only passwords of up to eight characters are supported.

[*acct*]

Specifies the account number for the new user ID. The default account number is the one specified in the skeleton file. If there is no account number in the skeleton file, the default account number is the same as the user ID name. To use the default account number, specify an asterisk.

[*distcode*]

Specifies the distribution code for the user ID to create. The default distribution code is the one specified in the skeleton file. If there is no distribution code in the skeleton file, the default distribution code is the same as the user ID name. To use the default distribution code, specify an asterisk.

[*disksize*]

Specifies the size of the 191 minidisk to allocate, specified either as the number of 512-byte blocks or in units of K (1024 bytes) or M (1024K bytes). By default, no 191 minidisk is created. Disk space is allocated from the default subpool or the subpool specified in response to a console prompt.

[*blocksize*]

Specifies the blocksize for formatting the new 191 minidisk. The default is 1K.

[*label*]

Specifies the minidisk label for the 191 minidisk. The default is DSK191.

Example

To place all the user IDs in the control file USER FILE on hold, enter:

```
vmdirect multiple hold user file
```

NEWIPL Command

Use the NEWIPL command to change an IPL system name or device.

```
NEWIPL
```

Description

The NEWIPL command changes the IPL system name or device in all directory entries to a new IPL system name or device.

The command leads you through the name change with prompts for information it requires (logon password, old IPL system name, new IPL system name) then changes the IPL statement of each directory entry that has the old IPL system name or device. To terminate the command before completion, enter QUIT or a null line in response to a name or device prompt.

User IDs that are on hold or have expired passwords are not affected by the command.

Example

You want to change the IPL system from CMS14 to CMS15. Begin by entering:

vmdirect newipl

At the password prompt, enter your logon password. When your password is validated, you are prompted for the old and new IPL system name. Enter the current name:

cms14

At the next prompt, enter the new name:

cms15

CA VM:Director then changes all directory entries that have the old name, giving you these status messages:

```
THERE WERE 176 USERS CHANGED
DIRECTORY UPDATED ONLINE
```

NOLOG Command

Use the NOLOG command to prevent everyone from logging on to a user ID.

```
NOLOG userid
```

Authorizations

The user ID issuing the NOLOG command through an EXEC or through CP SMSG must have NOPASS NOLOG authorization.

Definition

userid

Specifies the user ID of the virtual machine whose logon password to change to NOLOG.

Description

The NOLOG command changes a user ID logon password to NOLOG, preventing anyone from logging on to that user ID.

The directory manager must change a user ID's password after it has been changed to NOLOG before that user ID can log on again. This can be done with the PASSWORD command.

Note: For more information about expiring logon passwords that users can change when they try to log on, see EXPIRE Command (see page 233).

OVERRIDE Command

Use the OVERRIDE command to alter the privilege class structure on your system without shutting down CA VM:Director.

```
OVERRIDE filename filetype filemode [("Options"]
```

```
Options:
```

```
[CLEAR]
[DEFER]
[IMMEDIATE]
[VALIDATE]
```

Note: Options can be entered in any order.

Authorizations

You must have the CP privilege class to use the CMS OVERRIDE command before you can use this command.

Description

The OVERRIDE command locks the VOL1 label on the SYSRES volume and executes the CMS OVERRIDE command. After the command has run, CA VM:Director releases the lock and continues processing.

Definitions

filename filetype filemode

Identifies the filename, filetype, and filemode of the class override file.

CLEAR

Alters the privilege class structure to the IBM-defined user classes.

DEFER

Defers the changes in the class structure until the next time the system is IPLed.

IMMEDIATE

Changes the class structure immediately.

VALIDATE

Checks the class override file for syntax errors.

Note: For more information about the CMS OVERRIDE command, see the IBM *CP Planning and Administration* guide for your release of VM.

PAINT Command

The PAINT command creates or modifies CA VM:Director screen definitions.

```
PAINT filename [skeleton]
```

Definitions

filename

Specifies the filename of a screen to create or modify. This name is usually the same as the optimized macro that displays the screen. The filetype is always SCREEN.

skeleton

Specifies the filename of a screen definition to use as the beginning definition for this screen if you are creating a screen.

Description

Painting involves entering and manipulating background text, defining and naming fields, and assigning field attributes. It supports up to 84 fields for each panel.

To modify an existing screen file, issue the PAINT command with the existing screen's filename. Keep a backup copy of the original screen, because using the Enter key writes a copy of the current screen to the CA VM:Director A-disk.

To create a screen file:

1. Issue the PAINT command with the filename of the screen to paint. A blank, unformatted screen displays.
2. Enter background text anywhere on the screen except in the last byte of the last line. Background text includes screen title, field name, graphic display, and lines of text.
3. After you position the background text on the screen, identify the data fields**.** Enter a field mark character at the positions at which each field begins and ends. Leave enough blanks between the field mark characters for the field length.

   For example, this positions an 8-character data field after the background text Enter Userid:

   ```
   Enter Userid:;........;
   ```

   (On 3270 display terminals the field mark character displays as a semicolon with a bar over it).

4. Check the screen in formatted mode by pressing PF10. Formatted mode converts the field mark characters to start-field characters that appear as blanks. You can use the tab keys to move the cursor from one field mark character to the next. (Because all field mark characters have been translated to a start-field character, the cursor will move to areas of the screen that may not be data fields.)

5. Protect portions of the screen, such as background text, so that the cursor cannot move to them. Use the tab keys to position the cursor in an area to be protected and press PF4. This protects the area bounded by the closest field mark characters on each side of the cursor.

   When you finish protecting the necessary fields on the screen, the tab key positions the cursor only at the unprotected fields.

6. To assign or remove other field attributes, position the cursor and press the PF keys, or special keys, for those attributes. Attributes can be set when the screen is in formatted or unformatted mode (toggle between them with PF10). Set the field protection attribute in formatted mode.

   Special keys and their functions are listed in the table, PAINT Command Function Keys.

7. To highlight an area, position the cursor at the area and press PF6. Shift to formatted mode (PF10) to check it. The area bounded by the closest field mark characters on each side of the cursor is now highlighted. To highlight a smaller portion of the area, add field-mark characters where the highlighting begins and ends. Remember that the field mark characters are translated to blanks in formatted mode.

   For example, to highlight only the word **Userid:**, add a field mark character as follows:

   ```
   Enter;Userid:;........;
   ```

   Then place the cursor over **Userid** and press PF6. Return to formatted mode (press PF10) to check the highlighting. Because you also created a new field by adding the field mark character, recheck the protected fields.

8. To review the attributes set up for any field, position the cursor over the field and press PF12. All field attributes are displayed on the field identification menu. Fields that are accessible by programs must be assigned names. The name can be set or changed by typing the new field name in the unprotected field tagged **Name:**. Attributes controlled by PF keys (that is, protected, highlighted, modified, numeric, or displayed) can also be changed from the field identification menu screen by pressing the appropriate PF keys. Press Enter to return to the screen being painted.

9. If you add or remove field mark characters at any time, the associated field definition is cleared. Recheck all fields by doing the following:

   a. View the screen in formatted mode (PF10).

   b. Tab through fields to make sure appropriate fields are protected.

c. Review the field identification menu for each field. (Return to unformatted mode (PF10), position the cursor at each field mark character, and press PF12.)

In general, it is a good idea to define fields on the screen (that is, insert field-mark characters) only after the background text has been completed.

10. When the screen is complete, press Enter to write the screen file on the CA VM:Director A-disk.

PAINT Command Function Keys

The following table describes the function keys:

Function Key	Purpose
CLEAR	Clears the screen. Any data typed on the screen but not saved on the CA VM:Director A-disk is lost. The original screen is displayed again.
ENTER	Ends the PAINT command after saving the screen.
PF1 (HELP)	Displays a help screen that shows the functions performed by the twelve PF keys.
PF2 (ADD)	Adds a blank line to the screen below the cursor position. All subsequent lines are shifted down one line and the bottom line is lost. All field names below the inserted line are cleared and must be redefined.
PF3 (QUIT)	Ends the PAINT command without saving the screen.
PF4 (PROTECT)	Toggles the protection attribute associated with the current field. Protected fields cannot have data entered in them by terminal operators and are skipped over when the forward or backward tab key is pressed. However, if you assign a name to a protected field, a macro can move data to it by means of the SCREEN FIELD function. Fields are initially unprotected (an operator can add data to it).
PF5 (DELETE)	Deletes the line on which the cursor is positioned. All subsequent lines are shifted up one line and a new blank line is inserted as the bottom line. All field definitions following the deleted line are cleared and must be redefined.
PF6 (HIGHLIGHT)	Toggles the highlight attribute associated with the field on which the cursor is positioned. Fields are initially at normal intensity.

Function Key	Purpose
PF7 (MODIFY)	Toggles the Modified Data Tag (MDT) attribute associated with the current field. This tag causes the system to read data from the field even if the terminal operator does not enter anything in it, and is normally used for default parameters in unprotected fields. Fields are initially unmodified.
PF8 (SCALE)	Displays or hides a horizontal scale on the line at which the cursor is positioned. When the scale is removed, the line's previous contents are restored.
PF9 (NUMERIC)	If the terminal has the numeric lock feature, toggles the numeric attribute associated with the current field. Numeric locks restrict a terminal operator to enter only numeric digits 0 through 9, the period, minus sign, and dup key in numeric fields. Fields are initially alphanumeric (an operator can enter any character in a field).
PF10 (FORMAT)	Toggles the format mode of the entire screen. In format mode, fields are delimited by the invisible start-field character, and any attributes that have been assigned are in effect. In unformatted mode, all start-field characters appear as visible field-mark characters except the last character of the last line (this character remains a start-field character to prevent the entire screen from being cleared if you press ERASE EOF); attributes are not in effect, and field definitions can be altered. Screens are initially displayed in unformatted mode, where control characters are displayed and field definitions can be altered.
PF11 (DISPLAY)	Toggles the display attribute associated with the current field. This attribute determines whether data in the field is visible on the screen; it is generally used to hide passwords. Fields are initially displayable.
PF12 (FIELD)	Displays the field identification menu. This menu shows the name of the current field, its length, line number and column position of the start field character, and the settings of the various attributes. You can set or change the name or any attributes of the current field when this menu is displayed.

PASSWORD Command

Use the PASSWORD command to replace passwords in a user's directory entry.

```
PASSWORD userid [newpass] [(LOGON]
```

Definitions

userid

Is the user ID whose passwords you want CA VM:Director to change. If you specify *userid* without any other parameters, the command generates separate random read, write, and multiple passwords for all minidisks that have link passwords owned by that user ID, and a new random logon password.

The randomly generated passwords are of the form *cvccvc*, where *c* is a consonant and *v* is a vowel.

[*newpass*]

Is the new password to replace the existing password in the directory entry for *userid*. The password can be one to eight characters long and cannot include spaces.

If you do not specify a new password, the command generates a random password.

[?]

Reads a password string from the stack. If the stack is empty, the issuer is prompted for the password value with the console input being masked.

[LOGON]

Replaces only the logon password in the directory entry for *userid*.

[TERM]

Prompts for a password directly from a terminal, bypassing the program stack, with the console input being masked.

Description

The PASSWORD command changes the logon password or all passwords for a specific user ID. It can do so with specified or randomly generated passwords.

Minidisk passwords must contain printing characters only; they cannot contain blanks or left or right parentheses.

Password changes made with this command are not processed by the PASSWORD or MINIDISK user exit, if defined.

Minidisk passwords of ALL are unaffected by the PASSWORD command.

Whenever a password is changed, a record indicating the user ID whose password was changed, the type of password change, and the new password is appended to the file PASSWORD CHANGES A0. This file is written to your A-disk. A read/write A-disk is required.

Examples

Changing Passwords for Many User IDs

To change all the passwords for each user ID that begins with FIN, issue the following commands. In this example, the user ID issuing the commands has NOPASS PASSWORD authorization.

vmdirect query user fin* (manager * exec
copyfile cms exec a (spec

CMS responds:

```
ENTER SPECIFICATION LIST:
```

Enter:

1-15 1

At the Ready prompt, enter:

cms vmdirect password

CA VM:Director responds:

```
VMDIRECT PASSWORD FIN1
PASSWORD CHANGES FOR 'FIN1' COMPLETE
VMDIRECT PASSWORD FIN2
PASSWORD CHANGES FOR 'FIN2' COMPLETE
```

Specifying a Password for a Single User ID

To specify a new password, COVERT, for user ID SLEUTH, enter this command:

```
vmdirect password sleuth covert
```

The logon password and all minidisk link passwords are changed to COVERT.

Creating a Random Password

To create a new logon password for user ID JAN, enter:

```
vmdirect password jan (logon
```

CA VM:Director creates a random logon password for user ID JAN. All minidisk link passwords remain unchanged.

To find out what the new password is, look in your PASSWORD CHANGES file.

QLOCK Command

Use the QLOCK command to display all locks held by all processes in the server.

```
QLock
```

Description

You can use the output from QLOCK along with the output from the QPCB command to determine which user ID is blocking a particular request.

If you need to use this command when the CA VM:Director process on your virtual machine has not completed, issue it through the CP SMSG command. For example:

```
#cp smsg vmdirect qlock
```

You can issue the QLOCK command from the CA VM:Director service virtual machine to determine which user ID is preventing a non-forced shutdown of CA VM:Director.

Lock Names

Lock names are built hierarchically and describe the lock level. Elements of a lock name describe the sets over which the lock is placed. The leftmost name is the outermost set with an empty lock name indicating the universal set.

Directory Entry Locks

A directory entry lock name begins with DISK and has the following format:

DISK-*filemode*-*filetype*-*filename*

filemode

Specifies the filemode of the DRCT minidisk.

filetype

Specifies the directory manager user ID.

filename

Specifies the user ID of the directory entry being locked.

For example, a private lock with a lock name of DISK-U-MAINT-ABC is a private lock on the directory entry for user ID ABC and implies a shared lock on all directory entries managed by MAINT (DISK-U-MAINT).

Macro Locks

A macro lock name begins with VMNUC and has the following format:

`VMNUC-MACRO-VMDIRECT-`*`macroname`*

macroname

Specifies the name of the macro being locked.

Each time a user invokes a command that is implemented by a macro, a shared lock is placed on the corresponding macro. The MACLOAD command must have a private lock on the macro being updated by VMXMACUP. Therefore, macro updates automatically wait while a shared lock exists on a macro.

Permanent Locks on User IDs or Profiles

A permanent lock on a user ID or profile has the following format:

`LOCKUSER-`*`entryname`*

entryname

Specifies the name of the user ID or profile. A private permanent lock is obtained when the GETENTRY command with the LOCK option is used to copy a directory entry to a user's machine or when the LOCK USER command is used.

Permanent Locks on CMS Files

A permanent lock on one of the CA VM:Director CMS files has the following format:

`LOCKFILE-`*`filemode-filetype-filename`*

A private permanent lock is obtained when a user issues the LOCK FILE command.

Permanent Locks

Permanent locks are used to prevent updating of an entry or file by one user while another user has a copy of the same entry or file. They are retained during a shutdown and restart of CA VM:Director.

All permanent locks are written to a CMS file called PERMLOCK FILE, located on the CA VM:Director A-disk. The locks persist until a command clears them or you issue the UNLOCK command on them. If CA VM:Director terminates before a lock is cleared, CA VM:Director reads the PERMLOCK file at initialization and re-establishes all of the previously existing permanent locks.

Example

Your operator has issued the END command, but the CA VM:Director shutdown process has not completed. To determine which virtual machine is preventing the shutdown process from completing, enter the QLOCK command from the CA VM:Director service virtual machine console:

qlock

CA VM:Director responds with a one-line message for each outstanding lock:

```
THE MANAGE PROCESS HAS A PRIVATE DISK-U-MAINT-ABC LOCK.
THE MANAGE PROCESS HAS A SHARED VMNUC-MACRO-VMDIRECT-MANAGE LOCK.
```

Then issue the QPCB command to determine which user ID has issued the MANAGE command.

Note: For more information, see QPCB Command (see page 299).

QPCB Command

Use the QPCB command to list currently active processes.

```
QPcb
```

Description

The QPCB command lists all active processes and the commands they are executing, generating a single line of output for each process. You can use this command to determine who is using the system before shutting down CA VM:Director with the END command.

Output from the QPCB command takes the following format:

```
PCB      Userid   Name     PCB# Parent Root VirtCPU RPCWSED Semaphore
036BD000 BOB      QPCB     0023   0000 0023   0.000 rp.....
036F8BE0 BOB      DRONE    0022   0000 0022   0.000 rpc....
036BD260 VMX$1001 SERVANT  001B   0000 001B   0.000 rp.w... VMXUSE+04B0
036F8980 VMDIRECT $APPCMON 0015   0000 0015   0.000 rpc.s.. NUCAIE+1564
03763BE0 VMDIRECT AGENT    0012   0000 0012   0.001 rpcw... NUCAIE+0500
036F8720 VMDIRECT SYSTEM   000E   0000 000E   1.041 rpc.s.. NUCWAI+00D2
03763000 VMDIRECT $EDITOR  000D   0000 000D   0.067 r.c.s.. VMXBED+08E2
036F84C0 VMX$0002 SERVANT  000C   0000 000C   0.000 rp.w... VMXUSE+04B0
036F8260 VMX$0001 SERVANT  000A   0000 000A   0.000 rp.w... VMXUSE+04B0
036F8000 VMDIRECT $APPCVM  0007   0000 0007   0.000 rpc.s.. NUCAIC+0480
03763980 VMDIRECT SYSTEM1  0005   0000 0005   0.003 r.c.s.. NUCWAI+00D2
03763720 VMDIRECT CONSOLE  0004   0000 0004   0.000 rpcw... NUCCON+00DA
037634C0 VMDIRECT INTPCB2  0003   0000 0003   0.062 rpc....
03763260 VMDIRECT INTPCB1  0002   0000 0002   0.255 rpc....
```

The headings of the previously mentioned output is explained as the following:

PCB

Specifies address of the process control block for the process

Userid

Specifies the Userid that initiated the process

Name

Specifies the name of the process

PCB#

Specifies a hexadecimal identifier for the process. This same identifier appears in any message issued by the process.

Parent

Specifies the address of the PCB for the process's parent process (if there is no parent process, this values is 0000)

Root

Specifies address of the PCB for the processe's root process

VirtCPU

Specifies the virtual CPU time consumed by this process in seconds

R

Specifies root process

P

Specifies high-priority process

C

Specifies initiated from or associated with CA VM:Director service virtual machine console

W

Specifies asleep in NUCWAIT

S

Specifies waiting for an available semphore or resource

E

Specifies exited

D

Specifies been detached

.

Specifies process information is not true

Semaphore

Specifies an internal address of the location in memory where the process is waiting for a semaphore to be posted

QSTART Command

Use the QSTART command to display the date and time at which CA VM:Director was most recently started.

```
QSTART
```

Example

To determine when CA VM:Director was last started, enter:

`vmdirect qstart`

CA VM:Director responds with the last date and time at which it was started:

```
THE SYSTEM WAS LAST STARTED AT 19:28:30 ON mm/dd/yy.
```

QUERY Command

Use the QUERY command to provide information about a number of monitored functions.

```
QUERY {"Parms"}

Parms:

  {"ParmA"} [(MANAGER mgrid]
| {"ParmB"} [("OptionB"]
| {"ParmC"} [("OptionC"]
| {"ParmD"} [("OptionD"]
| {"ParmE"}

ParmA:

  Alloc
| Skeleton
| Subpool
| PROfiles

ParmB:

  ACcount [userid]
| Class [userid]
| FILEPOOL
| LOGMSG | DIALFAIL | LOGFAIL | NORULE | SYSTEM]
| Servant
| SFS
| SSI
| Version

OptionB:

[STACK [FIFO | LIFO]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

```
ParmC:

  Hold [userid]
| Users [userid]

OptionC:

[EXEC]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[MANAGER mgrid]
[NOSFS]
```

Note: Options can be entered in any order.

```
ParmD:

  Password num
| PRfusers profname

OptionD:

[EXEC]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[MANAGER mgrid]
```

Note: Options can be entered in any order.

```
ParmE:

Managers [( [STACK [FIFO | LIFO]] [FIFO] [LIFO] [NOSFS]]
| WORKUNIT workunitnum
```

Definitions

ACcount [*userid*]

Displays account information for your user ID. If you specify a user ID, the account number for that user ID as specified in the directory entry is displayed.

Alloc

If you are a directory manager, displays how much disk space you can allocate and have allocated. Space allocations are shown in kilobytes (K), megabytes (M), gigabytes (G), or terabytes (T).

Class [*userid*]

Displays privilege class information for your user ID. If you specify a user ID, the CP privilege classes for that user ID as specified in the directory entry displays.

FILEPOOL

If you are an SFS administrator, lists all file pools that CA VM:Director manages, their status (available or unavailable), and their location (local or remote, or unknown if status is unavailable).

Hold [*userid*]

If you are a directory manager, for all user IDs on hold that you manage, displays the date each user directory entry was last updated and the amount of disk and SFS space that has been allocated to each user ID. Space allocations are shown in kilobytes (K), megabytes (M), gigabytes (G), terabytes (T), petabytes (P), or exabytes (E).

You can specify one or more user IDs for which to display this information. You can use pattern matching for *userid*.

Managers

Lists directory manager user IDs and the amount of disk and SFS space that has been allocated to each user ID. Space allocations are shown in kilobytes (K), megabytes (M), gigabytes (G), terabytes (T), petabytes (P), or exabytes (E).

Password *num*

If you are a directory manager, for all user IDs you manage, displays information about user IDs whose passwords have not been changed for the specified number of days and whose logon passwords are not expired. (A new user ID that has not been logged onto is considered to have an expired password because the user must change the password at first logon.)

This parameter provides information about only those user IDs that have a password history statement in their directory entries. This is useful for creating a CMS EXEC file that contains user IDs whose logon passwords are to be expired. You can prevent a logon password from being expired by removing the password history special comment (*PW=) from the directory entry; however, the user ID’s logon password can still be expired if you issue the EXPIRE command for that user ID.

PROfiles

Lists the filenames of all directory PROFILE definitions.

PRfusers *profname*

If you are a directory manager, lists the directory entries that use the directory profile *profname*. User IDs on hold are included in the list and are not distinguished from active users.

SErvant

Displays status information about the Servant Facility.

SFS

If you are an SFS manager, lists the file pools and the user storage groups within those file pools from which you can allocate space, and, for each storage group, the number of blocks you allocate by default, the number of blocks you have already allocated, and the maximum number of blocks you can allocate.

SSI

Displays the names of the member systems in a Single system Image complex, and the status of the product server virtual machine on each member.

Skeleton

If you are a directory manager, lists the skeleton files you can use to define new user IDs.

SUbpool

If you are a directory manager, lists the subpools you are authorized to use when allocating disk space, total disk space available to a subpool, unallocated disk space in a subpool, and the largest amount of contiguous free space available for allocation in a subpool. Space allocations are shown in kilobytes (K), megabytes (M), gigabytes (G), or terabytes (T).

Users [*userid*]

If you are a directory manager, for all active user IDs you manage, displays the date each user directory entry was last updated and the amount of disk and SFS space that has been allocated to each user ID. The date is in *yy/mm/dd* format. Space allocations are shown in kilobytes (K), megabytes (M), gigabytes (G), terabytes (T), petabytes (P), or exabytes (E).

You can specify one or more user IDs for which to display this information. You can use pattern matching for *userid*.

Version

Displays the version of CA VM:Director that is currently running on your system.

WORKUNIT *workunitnum*

Displays the workunit associated with a currently running asynchronous CA VM:Director command.

An asynchronous command is one that partially runs on a Servant machine. These commands are typically those that involve formatting a minidisk, which is a time consuming operation if the disk is large. Some common asynchronous commands are ADDMDISK, CHGMDISK, DELMDISK, DELENTRY, DUPMDISK, MDISKSCAN, RECLAIM, and various menu entries for the MANAGE and USER menus that add or delete minidisks.

If the Servant Facility is configured, the format commands run in a Servant machine. The original user who issued a command to create, delete, duplicate, or relocate a minidisk receives message 558I, telling the user that the formatting is in progress and specifies the workunit number. When the format is complete, the user receives a spool file describing the success or failure of the format operation.

CA VM:Director associates a number with each asynchronous activity. This number is the workunit. Using the workunit number from message 558I, the user can then issue the QUERY WORKUNIT *workunitnum* command to see if the formatting is complete or has failed.

[EXEC]

Writes query output to a CMS EXEC file on your A-disk. You cannot use this option with the STACK option.

[MANAGER *mgrid*]

Specifies a directory manager to use in the query. An asterisk for mgrid, when this option is used with the HOLD, PASSWORD, PRFUSERS, and USERS parameters, indicates all directory managers.

[NOSFS]

Excludes SFS information from the results of the QUERY USERS command, displaying instead N/A for any SFS-related information. This option speeds up execution of the QUERY command because gathering SFS data can be time consuming.

[STACK [<u>FIFO</u> | LIFO]]

[LIFO]

[FIFO]

Stacks the query output in your program stack in first-in, first-out (FIFO) or last-in, first-out (LIFO) order. You cannot use this option with the EXEC option.

Examples

- To list the skeleton directory entries available for you to use when creating user IDs, enter:

  `vmdirect query skeleton`

- To query the log message file of your own default security group, as specified on the ACIGROUP statement in your directory, enter:

  `vmdirect query logmsg`

REBUILD Command

Use the REBUILD command to compress the contents of the active half of the CP object directory by writing it to the inactive half of the CP object directory, condensing and defragmenting the directory. Typically, you use this command only under the direction of Technical Support.

```
REBUILD
```

Description

Rebuilding the active portion of the CP directory speeds up directory changes by rearranging the files and free space scattered throughout the directory so that they are stored in contiguous units. Free space is consolidated in one contiguous block. When the directory is full or when there is a high percentage of fragmented disk space, you may notice that writing to the directory takes longer.

When CA VM:Director initializes, it splits the CP DRCT (the CP object directory) in two halves: one half is the active portion that CA VM:Director is writing to. The other second half of the DRCT area is inactive. As users use CA VM:Director to make directory changes, the changes are made to the active portion of the directory. As changes continue to be made, the disk space in the active portion fragments and fills. When the active portion completely fills, or if a user issues certain commands such as ADMIN PROFILE, CA VM:Director dynamically issues the REBUILD command. During the rebuild, CA VM:Director writes the contents of the active directory to the inactive directory, compressing the data as it goes. The newly compressed directory contents become the active portion, and the previous active portion becomes inactive.

RECLAIM Command

Use the RECLAIM command to reclaim DASD from minidisks that were moved while users were linked to them.

```
RECLAIM    [vaddr| *] [("Options"]

Options:

[ACCess | NOAccess]
[FILE [filename]]
[Format | NOFormat]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[Wait | NOWait]
```

Note: Options can be specified in any order.

Definitions

***[vaddr* | *]**

Specifies a moved read-only disk in the virtual address range 600 to 6FF. The default is asterisk (*), which tells CA VM:Director to check all addresses in the range for minidisks to reclaim.

[ACCess | NOAccess]

Checks all accessed minidisks (ACCESS) of the CA VM:Director service virtual machine, or skips checks of the accessed minidisks (NOACCESS) of the CA VM:Director service virtual machine. Skipping checks performs the RECLAIM command faster.

[Format | NOFormat]

Indicates whether to format the minidisk after reclaiming and scratching it.

[FILE [*filename*]]

[<u>MSG</u> | NOMSG]

[STACK [<u>FIFO</u> | LIFO]

[FIFO]

[LIFO]

[<u>Wait</u> | NOWait]

These options are explained in the Common API Command Options (see page 36).

Description

The RECLAIM command allows you to reclaim DASD after the links are removed without initializing CA VM:Director again. This command is issued automatically when CA VM:Director is initialized if the Servant Facility is turned on.

If you are authorized (you have MOVERO authorization), CA VM:Director lets you move minidisks even if users are linked to them read-only. To accommodate the read-only links, CA VM:Director removes the minidisks from the directory entry but does not format them. Instead, it holds the space the minidisks occupied until you use the RECLAIM command against them, and maintains them by redefining them to the CA VM:Director directory entry in the virtual address range 600 to 6FF. You cannot reclaim this DASD until the read-only links are removed, which can be done with either the CP DETACH or the CP LOGOFF command.

During dynamic reconfiguration using the CONFIG PRODUCT command, CA VM:Director allocates a new COMMON disk if you have an ACCESS COMMON record. This new disk is allocated with an address in the 600-6FF range. When the RECLAIM command runs it will reclaim this COMMON disk if it is no longer current or in use by any servant process.

Important! Do not assign minidisks to the CA VM:Director server in the virtual address range from 600 to 6FF. CA VM:Director will attempt to reclaim minidisks in this range, and will eventually format and scratch them.

REPENTRY Command

Use the REPENTRY command to replace a USER, IDENTITY, SUBCONFIG, or PROFILE directory entry.

To replace a user directory entry:

```
REPEntry "Parms" [("Options"]
```

```
Parms:
```

```
entryname [{filetype | * | DIRECT} [filemode| * | A]]
```

```
Options:
```

```
[FILE [filename]]
[Format | NOFormat]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[Wait | NOWait]
```

Note: Options can be specified in any order.

To replace a directory profile:

```
REPEntry "Parms" (PROFILE ["Options"]
```

```
Parms:
```

```
filename [{filetype | * | DIRECT} [filemode | * | A]]
```

```
Options:
```

```
[FILE [filename ]]
[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be specified in any order.

Definitions

***name* [*filetype* [*filemode*]]**

Specifies the user ID whose directory entry to replace, and name of the copy of the directory entry. The user ID must be an active user ID.

The default file specification for the new directory entry is "*name* DIRECT A". An asterisk means the same as specifying DIRECT.

If you specify a filemode, it must be one of your currently accessed minidisks. (If you include this REPENTRY command in a program, the filemode must be that of a accessed minidisk in the submitting user ID. If you specify an asterisk for filemode, the command uses the normal CMS search order to find the file.

***name* [*filetype* [*filemode*]] (PROFile**

Specifies the directory profile whose directory entry CA VM:Director is to replace and the filetype and filemode of the new directory entry.

The default file specification for the new directory entry is "name DIRECT A."

If you specify a filemode, it must be one of your currently accessed minidisks. (If you include this REPENTRY command in a program, the filemode must be of a minidisk that the user ID that submits the program for execution currently has accessed.)

If you specify an asterisk for filemode, the command uses the normal CMS search order to find the file.

[Format | NOFormat]

Formats (FORMAT) or does not format (NOFORMAT) new minidisks after allocating them, original minidisks after moving them, and deleted minidisks after deleting them. You must have the appropriate NOFORMAT authorization to specify NOFORMAT. (If you include this REPENTRY command in a program, the user ID that submits the program for execution must have the appropriate NOFORMAT authority.)

Deleted or moved minidisks are formatted using a blocksize of 4K, and their labels are changed to ERASED.

CA VM:Director applies the option selected here to all minidisks specified on this command. The FORMAT or NOFORMAT options on an MDOPT statement can override that status for specific new or original minidisks. (MDOPT statements cannot control the action to take on deleted minidisks.)

CA VM:Director ignores these options for minidisks defined on T-DISK or V-DISK.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[Wait | NOWait]

These options are explained in the Common API Command Options (see page 36).

Description

The REPENTRY command can replace directory entries for active user IDs only. If you want to replace the directory entry for a user ID on hold, you must first reactivate the user ID.

Any directory statement can appear in the new directory entry for a user ID, including MDISK statements. The REPENTRY command lets the issuing user ID create, change, move, and delete minidisks for a user ID. The issuing user ID must be separately authorized to use the ADDMDISK, CHGMDISK, and DELMDISK commands before CA VM:Director will work with minidisks through the REPENTRY command.

If the new directory entry deletes minidisks that were in the old directory entry**,** then when you issue the REPENTRY command, CA VM:Director deletes the minidisks. If the minidisks are on packs that are under CA VM:Director control, CA VM:Director deletes links to the deleted minidisks from the directory.

The REPENTRY command cannot change logon passwords; the original logon password is maintained in the new directory entry. Minidisk link passwords can be changed by replacing the masking strings with the new passwords. The string $NONE$, which indicates that the specified password is to be deleted, is reserved, and cannot be used as a link password on any MDISK statement.

To use the REPENTRY command, the issuing user ID must have previously executed a GETENTRY command that specified the LOCK option. Locking a directory entry or profile prevents anyone else from modifying it while the issuing user ID is working with it. If the issuing user ID decides not to update the entry or profile after all, running the UNLOCK command removes the lock.

The process of replacing the directory entry for a user ID occurs in two steps:

1. CA VM:Director compares all statements from the new and original entry and calls the appropriate user exit routines for changed values. It compares the MDISK statements and saves all necessary information from any changed statements. CA VM:Director then replaces the entire directory entry, retaining the original MDISK statements. If it detects any errors, it does not perform the second step.

2. CA VM:Director tries to make all of the minidisk updates by moving, adding, or deleting all the minidisks identified during step 1. If it detects any errors, CA VM:Director continues, updating as many minidisks as possible.

When the issuing user ID replaces the directory entry for a directory profile, CA VM:Director does not call user exits or allow minidisks. CA VM:Director updates the directory entry of every user ID that is linked to the original directory profile to use the new directory profile.

A BUILD statement may neither be added or removed when an IDENTITY entry is updated by REPENTRY.

MDISK and MDOPT Statements

CA VM:Director creates, changes, or moves minidisks for a user ID based on MDISK and MDOPT statements in the new directory entry that is replacing the original directory entry.

MDISK statements define minidisks. MDOPT statements define options for minidisks. MDOPT statements precede MDISK statements; CA VM:Director applies the options specified on an MDOPT statement to all MDISK statements that follow that statement until CA VM:Director encounters an MDOPT statement with different options. In this way, the issuing user ID can specify unique options for each minidisk. If CA VM:Director does not find an MDOPT statement in the directory entry, it uses options specified on the REPENTRY command and default values described in MDOPT Statements.

When replacing the directory entry, the REPENTRY command evaluates the MDISK statements in the original entry against the MDISK statements in the new entry and proceeds as indicated:

MDISK statement contents	REPENTRY command function
In the new entry that specifies all of the same values as an MDISK statement in the original entry	Leaves the minidisk unchanged
In the new entry that specifies a virtual address not specified on any MDISK statement in the original entry	Creates a new minidisk
In the new entry that specifies a virtual address specified on an MDISK statement in the original entry, but the values defining the minidisk are different	Redefines the minidisk in new space and moves the data
In the original entry that specifies a virtual address not specified by an MDISK statement in the new entry	Deletes the minidisk

When REPENTRY executes, CA VM:Director passes the values specified on REPENTRY, the MDISK statements, and the MDOPT statements to the ADDMDISK and CHGMDISK commands, which create, change, or move the minidisks. REPENTRY calls the DELMDISK command to delete minidisks.

CA VM:Director cannot move a minidisk defined in system T-DISK or V-DISK space to permanent space, or conversely. CA VM:Director can only redefine T-DISK or V-DISK minidisks with a different size or on a different device type.

MDISK Statements

Because CA VM:Director can dynamically add or move a minidisk, CA VM:Director supports a special format of the MDISK statement that allows it to automatically determine where to place the new minidisk. The parameters appear in the same order as in a true MDISK record and their values are very similar to those on the ADDMDISK and CHGMDISK commands.

To create a permanent minidisk, a temporary minidisk, or a virtual minidisk, you use the following formats:

```
Mdisk vaddr [{devtype | *} [{start | *} [{size | *} [{volid | *} ["Parms"]]]]]
Mdisk vaddr {devtype | *} TDSK [size | *]
Mdisk vaddr FB-512 VDSK [{size | *} ["Parms"]]
Mdisk vaddr {devtype | *} DEVNO [{rdev | *} ["Parms"]]
```

```
Parms:
```

```
{mode | *} [rpw [wpw [mpw]]]
```

To change or move a permanent minidisk, a temporary minidisk, or a virtual minidisk, you use the following formats:

```
Mdisk vaddr [{devtype|=|*} [{start|=|*} [{size|=|*} [{volid|=|*} ["Parms"]]]]]
Mdisk vaddr [{devtype|=|*} [{TDSK | =} [{size | = | *}]]]
Mdisk vaddr FB-512 VDSK [{size|=|*} ["Parms"]]
```

```
Parms:
```

```
{mode | = | *} [{rpw | =} [{wpw | =} [mpw | =]]]
```

To change the minidisk label or passwords but not move the minidisk, the issuing user enters an equal sign or the original value for each of the devtype, start, size, and volid fields.

Note: For parameter definitions, see the ADDMDISK Command (see page 53).

MDOPT Statements

CA VM:Director uses MDOPT statement options when it creates, changes, or moves minidisks. This section shows the format of and option definitions for MDOPT statements. CA VM:Director ignores the FORMAT | NOFORMAT, SUBPOOL, and LABEL options when creating or changing minidisks on T-DISK or V-DISK. For more information about MDOPT statement options, see the definitions for equivalent options in ADDMDISK Command in this chapter.

CA VM:Director removes all MDOPT statements from the new directory entry before it places the entry in the source directory.

```
MDOPT "Parms"


Parms:

[Format | NOFormat]
[SUBpool {poolname | *}]
[BLKsize {blksize | *}]
[Label {label | *}]
[AUnit {unitsize | *}]
[STEM stemname[.]]
```

[Format | NOFormat]

Specifies whether CA VM:Director is to format the new and original minidisks. For these minidisks, this option overrides the FORMAT option on the REPENTRY command.

[SUBpool poolname]

Specifies the subpool from which CA VM:Director is to allocate new minidisks. The default for poolname is *MNGR, which searches the list of subpools in which the directory manager for this user ID can allocate space for the first subpool with enough space.

If you specify the subpool by name, that subpool must be defined in the DASD CONFIG file and included in your or the owning manager's list of subpools in the VMDIRECT MANAGERS file. ANYPOOL authorization overrides this restriction. To specify a subpool list, use one of the variable subpool categories listed in the table, Variable Subpool Categories, in the section Subpool Categories earlier in this chapter.

[BLKsize [blksize]]

Specifies the CMS block size CA VM:Director is to use when formatting minidisks. Possible values for blksize are 512, 1024, 2048, 4096, 1K, 2K, and 4K. The default for blksize is the value specified for AUNIT; if no value is specified for AUNIT, the default for blksize is the default block size defined for the device type.

If you specify NOFORMAT on the REPENTRY command and the MDOPT statement, CA VM:Director ignores this option.

[Label [label]]

Specifies the minidisk label that CA VM:Director is to give minidisks when it formats them. The default is DSKvaddr. CA VM:Director replaces vaddr with the virtual device address.

If you specify NOFORMAT on the REPENTRY command and the MDOPT statement, CA VM:Director ignores this option.

[AUnit [unitsize]]

Specifies an alternate allocation unit size for CA VM:Director to use when it calculates the size of new minidisks. When this option is used, CA VM:Director interprets the value for size on MDISK statements as the number of blocks of the size unitsize. It then uses this number to calculate the actual allocation size for the type of device used.

Possible values for unitsize are 512, 1024, 2048, 4096, 1K, 2K, and 4K. The default for unitsize is the specified or default value for *blksize*.

[STEM *stemname*[.]]

Specifies the name of REXX stem variable array containing the statements which define the directory entry contents. Any filetype or filemode parameters are ignored. The array contains the directory entry definition, so no disk file is referenced. The number of statements in the directory entry is defined in element *stemname.0, stemname.1* contains the first statement in the directory entry, and so on. The stem must be defined in the same REXX program which calls the REPENTRY command, before the command is called.

Return Codes and Error Messages

The table that follows shows errors that do not cause immediate termination, but do prevent CA VM:Director from updating minidisks. The command exits with return code 54 and message 633E.

REPENTRY Command: Error Messages (No Immediate Termination)

The following table describes the messages:

Message Number	Text
0383I	FROM EXIT: (*user coded messages from userexits*)
0451I	ACCOUNT NUMBER *number* IS INVALID FOR USER *userid*
0562E	MINIDISK *vaddr* NOT *action*, CURRENTLY IN USE
0662E	MINIDISK *userid vaddr* IS A T-DISK AND MAY NOT BE MOVED TO OR FROM PERMANENT SPACE

The table, REPENTRY Command: Return Codes and Error Messages (Immediate Termination), lists errors that cause immediate termination and shows the return code for each.

REPENTRY Command: Return Codes and Error Messages (Immediate Termination)

The following table describes the return codes and the associated error messages:

Return Code	Message Number	Text
2	0018E	INVALID OPTION *option*
4	0644E	NOWAIT OPTION SPECIFIED BUT THE SERVANT FACILITY IS NOT ACTIVE
6	0648E	THE NOWAIT OPTION MAY NOT BE SPECIFIED WITH THE STACK OR FILE OPTIONS
12	0038E	MISSING PARAMETER
14	0531E	PARAMETER *userid* IS TOO LONG
16	0265E	NOT AUTHORIZED FOR *authorization words*
18	0127E	USER *userid1* EXCEEDED USER *userid2 type* PASSWORD LIMIT {FOR *vaddr*}
20	0349E	PASSWORD IS INCORRECT
22	0356E	USER *userid* DOES NOT EXIST
25	0260E	PROFILE *profname* IS UNDEFINED
26	0382E	USER *userid* IS ON HOLD
28	0039E	INVALID PARAMETER *parameter*
30	0643E	EXTRA PARAMETERS *parameters* WERE ENTERED AND ARE NOT ALLOWED
32	0681E	FILE *filename filetype filemode* DOES NOT EXIST
34	0659E	ENTRY *entry* MUST BE RETRIEVED USING THE LOCK OPTION OF GETENTRY BEFORE USING REPENTRY TO REPLACE IT
36	0660E	ENTRY *entry* WAS PREVIOUSLY RETRIEVED FOR UPDATE BY *userid*
38	0621E	UNEXPECTED RETURN CODE *code* FROM OPEN
40	0076S	USER ERROR *code* COPYING FILE *filename filetype*
42	0621E	UNEXPECTED RETURN CODE *code* FROM OPEN

Return Code	Message Number	Text
44	0277E	FIRST RECORD *word1 word2* CONFLICTS WITH FILEID *filename filetype*
46	0277E	FIRST RECORD *word1 word2* CONFLICTS WITH FILEID *filename filetype*
48	0631E	THE NAME *name* ON THE { PROFILE | USER } RECORD MUST MATCH THE FILENAME OF THE DIRECTORY ENTRY
52	0018E	INVALID OPTION *option*
54	0633E	ONE OR MORE ERRORS WERE DETECTED WHILE PROCESSING THE UPDATE OF USERID *userid*

Note:

- For explanations of the message routing and disconnect options, see Common API Command Options (see page 36).
- For MDISK statement parameter definitions, see ADDMDISK Command (see page 53). (They are equivalent to the ADDMDISK command parameters)
- For NOFORMAT authorizations, see Authorizations for Command Processing table in the chapter "Authorizations" of the *Administration Guide.*
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide.*

RESET Command

Use the RESET command to reset to zero the number of invalid password attempts recorded for a specific user ID.

```
RESET {"Parms"}

Parms:

USERPASS userid
| VMXLINK requester [targetid [vaddr]]
```

Definitions

USERPASS *userid*

Clears the journal count for password violations committed by user ID while CA VM:Director was verifying that user ID was indeed the user.

VMXLINK *requester* [*targetid* [*vaddr*]]

Clears the journal count for a link password violation committed while the requester tried to define a directory link to the targetid vaddr minidisk. If you specify an asterisk for vaddr, the command resets journal counts for all minidisks owned by targetid. This is the default.

If you do not specify targetid, the command clears all invalid link password violations by requester while attempting to define a directory link. When you do not specify targetid and the RESET command resets the journal count for more than one user ID, CA VM:Director creates an audit record for each user ID.

Description

The RESET command clears the number of invalid password attempts recorded for a specific user ID. The JOURNAL record in the SECURITY CONFIG file sets the maximum number of invalid passwords that a user ID can enter before CA VM:Director prevents that user ID from doing either of the following:

- Using CA VM:Director commands if the password violations were in response to the verification password prompt

- Defining a minidisk link for any minidisks belonging to the user ID for which link passwords were violated

Invalid password counts are set to zero when the CA VM:Director service virtual machine is initialized.

The RESET command is similar in function to the JOURNAL command with the RESET parameter.

Note: If a directory manager exceeds the maximum number of invalid password entries while attempting to define a minidisk link on behalf of a user, the password violations are recorded against the directory manager. To clear the password violations recorded, issue the RESET command with the directory manager's user ID as *requester* and the user ID of the minidisk's owner as the *targetid*.

Examples

- To clear the invalid password record for HANK's invalid attempts to use an CA VM:Director command, enter:

  ```
  vmdirect reset userpass hank
  ```

- To clear the JOURNAL count for EDDIE's invalid attempts to define a directory link to the EARL 191 minidisk, enter:

  ```
  vmdirect reset vmxlink eddie earl 191
  ```

Note: To use the JOURNAL command as another method of clearing and resetting journal information, see the JOURNAL Command (see page 249).

REVOKE AUTHORITY Command

Use the REVOKE AUTHORITY command to revoke access to SFS directories or files from users.

```
REVOKE AUTHORITY "Resource" FROM {"Who"} [("Options"]
```

```
Resource:
```

[*filename filetype*] *dirid*

```
Who:
```

```
  userid
| ALL
| PUBLIC
| nickname
```

```
Options:
```

```
[KEEPDirread | KEEPNewread | KEEpread | NEWauth | RWAuth]
[TYPe | NOType]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

The CA VM:Director REVOKE AUTHORITY command syntax is identical to the syntax of the CMS REVOKE AUTHORITY command.

Note: For more information about parameters and options for the REVOKE AUTHORITY command, see the IBM *CMS Commands and Utilities Reference* for your release of VM.

Authorizations

Enroll the CA VM:Director service virtual machine as an SFS administrator in the DMSPARMS configuration file of every file pool for which the REVOKE AUTHORITY command will be issued. CA VM:Director does not need to manage the file pool.

You must have REVOKE AUTHORITY authorization through a GRANT record in the CA VM:Director AUTHORIZ CONFIG file.

Description

The CA VM:Director REVOKE AUTHORITY command allows you to remove authorization from a user ID to access certain files and directories. You can also use this command to downgrade a user's authority. The REVOKE AUTHORITY command can only revoke authorities previously established with either the CA VM:Director GRANT AUTHORITY command or the CMS GRANT AUTHORITY command.

After verifying that the issuing user ID is authorized to perform the REVOKE AUTHORITY command for the target user IDs (including all user IDs in a nickname list), CA VM:Director passes the REVOKE AUTHORITY command to CMS.

CA VM:Director generates an audit record (1080) for each user ID specified on the command. If a nickname is specified, CA VM:Director generates an audit record for every user ID defined in the nickname list.

Nickname Resolution

When you specify a nickname for *userid*, CA VM:Director first uses information in your NAMES file to determine the local user. If there are user IDs that CA VM:Director cannot resolve using the NAMES file, it then uses the DMSJNE routine to resolve the remaining unresolved user IDs. DMSJNE is an optional customer-written routine that returns a local user ID for the supplied user ID and nodeid. If your site uses DMSJNE and it is not available, or if CA VM:Director cannot resolve any of the user IDs, the REVOKE AUTHORITY command terminates with a return code of 328.

Examples

- You previously granted all users the authority to read the TEST RESULTS file in the VMSYSU:ENG.TOOLS directory. To prevent all users from reading the TEST RESULTS file, enter:

  ```
  vmdirect revoke authority test file vmsysu:eng.tools from all
  ```

- You want to prevent FRAN from reading or writing any files created in the future in the VMSYSU:ENG.TOOLS directory, and allow him to only look at names of files in that directory. You also want to remove read and write authorizations to the files in that directory. Enter:

  ```
  vmdirect revoke authority vmsysu:eng.tools from fran (newauth keepread
  vmdirect revoke authority * * vmsysu:eng.tools from fran
  ```

- You want to change WOODYB's authorization from WRITE to READ for all files in the VM:EMP.INFO directory. You also want to let him read any new files created in the future. Enter:

  ```
  vmdirect revoke authority vm:emp.info from woodyb (keepread keepnewread
  vmdirect revoke authority * * vm:emp.info from woodyb (keepread
  ```

Return Codes and Error Messages

The table, REVOKE AUTHORITY Command: Return Codes and Error Messages, lists return codes and error messages for the REVOKE AUTHORITY command.

The REVOKE AUTHORITY command may also generate CMS messages.

Note: For more information about these messages, use the CMS help or see the IBM messages and codes documentation.

REVOKE AUTHORITY Command: Return Codes and Error Messages

The following table describes the return code and the associated error message:

Return Code	Message Number	Text
24	1117E	REVOKE IS NOT A CA VM:Director COMMAND
38	0038E	MISSING PARAMETER
39	0039E	INVALID PARAMETER *parameter*
76	0076S	USER ERROR *code* COPYING FILE *nickname_file*
99	0099I	REVOKE COMMAND CANCELED
221	0221E	MISSING OPTION OPERAND
265	0265E	NOT AUTHORIZED FOR: REVOKE *filepool userid*
328	0328E	INVALID USERID *userid*
543	0543E	THE SERVANT FACILITY IS NOT CURRENTLY TURNED ON
586	0586E	ERROR *code* STARTING SERVANT
625	0625E	TOO MANY SERVANTS IN USE, COMMAND CANNOT BE EXECUTED
7063	7063E	EXTRANEOUS PARAMETER(S) *parameters*
7065	7065E	FILE POOL *filepoolid* NOT RESPONDING TO REQUESTS

Note:

- To authorize a user ID to read from or write to files and directories in SFS, see GRANT AUTHORITY Command (see page 244).
- For more information about the audit record produced by the REVOKE AUTHORITY command, see AUDITEXT Command (see page 71).

- For information about the REVOKE AUTHORITY authorization, see the chapter "Authorizations" in the *Administration Guide.*
- For complete descriptions of all messages listed in this section, see the *Message Reference* Guide
- For detailed information about the parameters and options on the REVOKE AUTHORITY command, see the IBM CMS command reference documentation
- For information about coding the DMSJNE routine, see the IBM SFS and CRR planning, administration, and operation documentation.
- For information about other CMS messages generated by the REVOKE AUTHORITY command, see the IBM messages and codes documentation.

SCAN command

Returns information about records in the source directory. Information is returned to the issuer's console. Each line of output consists of user ID and matching directory record.

```
SCAN token ... [("Options"]

Options:

[MSG | NOMSG]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
[FILE [filename]]
```

Note: Options can be specified in any order.

Authorizations

You must have SCAN authorization in the AUTHORIZ CONFIG file.

Definitions

Token

Each record from the CP source directory is matched against the token(s) provided.

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

[FILE [*filename*]]

These options are explained in the Common API Command Options (see page 36).

Description

The following points help to understand the way SCAN command obtains the record information from the source directory:

- Each record from the CP source directory is matched against a pattern string. The pattern string consists of up to 13 blank delimited tokens. The pattern string tokens are matched up against blank delimited tokens from each record of the CP source directory, and matching records are returned to the caller.
- If all tokens in the pattern match the corresponding tokens in the directory record, then the directory record is considered a match.
- An asterisk (*) may be used as a wildcard character in the pattern. Any number of asterisks may appear in a token of the pattern. An asterisk is considered to match any number of characters (including zero characters) in the corresponding token of the directory record. An asterisk can be used alone as a wild card indicating that all values found in that position in the record are matches.
- Any pattern token consisting only of wild card characters, for example, '*****', is treated the same as a token consisting of a single asterisk.
- A pattern consisting of a single asterisk returns all comment records (not all records).
- A scan pattern consisting of nothing but wild card designators in the form SCAN * * * * * * * * * * * * * is equivalent to the command SCAN *. Using this scan pattern would result in only comment records being returned (not all records). Special processing occurs when evaluating the first token of a directory record with the first token of the pattern. The first token is the directory statement type. The first token of a directory record may be an abbreviation of the directory statement type. If so, it is expanded out to the full, non-abbreviated statement type. For example, I is expanded to IPL, or IN is expanded to INCLUDE. Similarly, if the first token of the pattern does not contain asterisks, and is an abbreviation of a valid directory statement type, it is also expanded out to the full un-abbreviated statement type.
- Scan results may cause sensitive information (logon and minidisk passwords) to be sent to the requestor. Use caution in deciding who is authorize to use this command so that this sensitive information is not inadvertently given to users who should not have it.
- The search parameter list provided to SCAN is uppercased and compared to uppercased directory records. You must consider this when attempting to scan for directory statements that allow mixed case arguments such as POSIXGLIST, POSIXGROUP, and POSIXINFO statements.

Return Codes and Error Messages

The table, SCAN Command: Return Codes and Error Messages, lists return codes and error messages for the SCAN command.

The SCAN command may also generate CMS messages.

Note: For information about these messages, use the CMS help or see the IBM messages and codes documentation.

SCAN Command: Return Codes and Error Messages

The following table list the return code and the associated message:

Return Code	Message Number	Text
76	0018E	INVALID OPTION
16	0038E	MISSING PARAMETER
28	0826W	NO DIRECTORY RECORD FOUND MATCHING THE SEARCH CRITERIA

SUBCONFIG Command

Use the SUBCONFIG command from an IDENTITY user to bring up a screen presenting the names of all SUBCONFIG entries which are part of the user virtual machine definition.

```
SUBCONFIg
```

Description

The SUBCONFIG command displays a panel that allows you to select a specific SUBCONFIG entry for processing. The selected SUBCONFIG is then displayed in the same manner as if the USER command had been issued for that SUBCONFIG directory entry, to let users modify SUBCONFIG directory entries that are part of their own virtual machine definition.

Note: For more information on the facilities available with the USER command, see USER Command (see page 348).

This command requires an IBM 3270 display or compatible terminal.

You can authorize use of each of the selections individually so that users have access to only some of them.

TAG command

Use the TAG command to add or manipulate the TAG comment record. The form of the tag comment record is defined as an '*' followed by the name of the tag followed by a colon. This distinguishes the tag record from a comment record or a special record.

```
TAG { "Parms" }

Parms:

  DEFine      tagname nnn
| REVoke      tagname
| Query       {tagname | ALL}  [ ("Options"]
| SET         {userid |*}  tagname data
| DELete      {userid |*}  tagname
| SHOW        {userid |*}  {tagname | ALL} [("Options"]

Options:

[MSG | NOMSG ]
[STACK [FIFO|LIFO]]
[FIFO]
[LIFO]
[FILE [filename]]
```

Note: Options can be specified in any order.

The default for *filename* is TAG. The value for filetype is always RESULTS. These options are explained in the Common API Command Options (see page 36).

Authorizations

You must be authorized for the TAG command in the AUTHORIZ CONFIG.

Definitions

DEFine

Define the name of the TAG and the ordinal to the system.

REVoke

Delete an existing tag definition.

Query

Display the tag name and the associated sort ordinal.

SET

Set the value of a user's tag.

DELete

Delete the value of a tag from a user's directory entry.

SHOW

Display the tag name and the value of the tag.

tagname

Specifies the name of the tag. The tagname can be up to 8 alphanumeric characters long.

ALL

Specifies all tag name definitions in the system.

nnn

A variable specifying the value of the tag sort ordinal; this value is relative to the other defined local tags. Value can be 0-999. The ordinal is used when adding or updating TAG values. The TAG comment records are placed into a user directory entry in ordinal number order.

userid

The user ID whose TAG records you are manipulating.

*****

Means the issuer of the command.

data

Specifies the value of the TAG. This value consists of tokens of data separated by blanks. The total length of all tokens plus the blanks separating them may not exceed 240.

Important! The TAG command creates a file on VMDIRECT 191 called TAG CONFIG. Do not modify this file in any way.

Return Codes and Error Messages

The table, TAG Command: Return Codes and Error Messages, lists return codes and error messages for the TAG command.

The TAG command may also generate CMS messages.

Note: For information about these messages, use the CMS help or see the IBM messages and codes documentation.

TAG Command: Return Codes and Error Messages

The following table lists the return code and the associated message:

Return Code	Message Number	Text
76	0018E	INVALID OPTION
4	0038E	MISSING PARAMETER
28	0039E	INVALID PARAMETER *parameter*
76	0080E	USER *userid* DOES NOT EXIST
32	0356E	TAG *tagname* DOES NOT EXIST
16	0358E	TAG *tagname* ALREADY EXISTS IN FILE
12	0358E	ORDINAL *ordinal* ALREADY EXISTS IN FILE
24	0819E	TAG CANNOT BE REMOVED. IT IS IN USE BY *userid*
8	7079E	ITEM NOT FOUND

Example:

- To define several tag records to hold a user's name and address, enter the following commands:

  ```
  vmdirect tag define name 100
  vmdirect tag define addr 200
  vmdirect tag define state 500
  vmdirect tag define city 300
  ```

- To display all tags, enter the following command:

  ```
  vmdirect tag query all
  ```

  The result is displayed as follows:

  ```
  *NAME:     100
  *ADDR:     200
  *CITY:     300
  *STATE:    500
  ```

- To assign values to the tags, enter the following command:

  ```
  vmdirect tag set usera name Zack Bradford
  vmdirect tag set usera addr Maple Avenue
  vmdirect tag set usera state va
  vmdirect tag set usera city Anytown
  ```

- To show all tags for USERA, enter the following command:

  `vmdirect tag show usera all`

  The result is displayed as follows:

```
*NAME:     Zack Bradford
*ADDR:     Maple Avenue
*CITY:     ANYTOWN
*STATE:    VA
```

TAKEOVER Command

Use the TAKEOVER command in a Single System Image environment to recover from an outage of the member node running the Master product server virtual machine.

```
TAKEOVER
```

Description

The TAKEOVER command is issued to an Agent product server to request that it be converted to become the new Master product server, when the old Master is lost because of an SSI member system outage.

Note: For more information on the procedure for recovering from a member outage, see the section “Master Server Member Outage” in the *Administration Guide*.

TRACE Command

Use the TRACE command to trace macro execution.

```
TRACE {"Parms"}

Parms:

  MACRO "Macroparms"
| LIST [userid [macname]]
| STOP {tracenumber | * | [USER] userid}
| CLOSE {tracenumber | * | [USER] userid}

Macroparms:

[macname | *]
[LINES first last]
[OPT option]
[USER userid]
[TO destuser]
[TYPE | MSG | PRINT]
```

Definitions

LIST [*userid* [*macname*]]

Lists all active traces and their trace number for the specified user IDs and macros. Both *userid* and *macname* can contain the pattern matching characters % and *. The default value for *userid* is the issuing user ID and the default for *macname* is all macros.

MACRO

Indicates that the command is to trace the specified macros. A trace number is generated from this parameter which you can use later to stop tracing.

[*macname*]

Specifies the filename of the executable macros to trace. The macro name can contain the pattern matching characters % and *. The default is all macros. You must specify the *macname* parameter if you use any of the subsequent parameters.

[LINES *first last*]

Indicate the numbers of the first line and the last line of the range of macro lines to trace. The default for *first* is 0 and the default for *last* is the last line of the macro.

[OPT *option*]

Specifies the amount of information the command is to include in the trace messages. If the macros are written in EXEC2, you can use one of the following options:

- ALL
- ERR
- OFF
- ON

If the macros are written in REXX, you can use one of the following options:

- A
- C
- ERR
- I
- N
- OFF
- R
- <u>ALL</u>
- E
- ERROR
- L
- O
- ON

[USER *userid*]

Specifies that the command is to trace the indicated macros when run by this *userid*; *userid* can contain the pattern matching characters * and %. The default is your user ID.

[TO *destuser*]

Specifies the user ID to receive trace messages. The default is the user ID specified on the USER parameter. Specifying a *destuser* value of '*' sends the messages or print file to the CA VM:Director product server.

[TYPE | MSG | PRINT]

Indicates where to send the trace messages:

- TYPE sends the messages to the terminal of the TO user ID; when the TO user ID and the USER user ID are the same, TYPE is the default. When the user ID specified on the TO parameter is different than the user ID specified on the USER parameter, CA VM:Director changes the TYPE parameter to MSG
- MSG sends the messages using CP MSGNOH or MSG to the terminal of the TO user ID; when the TO user ID and the USER user ID are different, MSG is the default
- PRINT sends the messages in a printer spool file to the virtual reader of the TO user ID

STOP {*tracenumber* | * | [USER] *userid*}

Cancels the specified active trace. The *tracenumber* is the number CA VM:Director assigns to each trace; you can use the LIST parameter to determine this number. The * cancels all traces currently defined on the system. The USER *userid* parameter cancels all traces that apply to the specified user ID. The USER keyword is required only if the user ID is all numeric.

CLOSE {*tracenumber* | * | [USER] *userid*}

Closes a printer spool file for the specified active trace, but tracing continues. Use the CLOSE parameter only to close traces that were started with the PRINT parameter. The *tracenumber* is the number CA VM:Director assigns to each trace; use the LIST parameter to determine this number. The * cancels all traces currently defined on the system. The USER *userid* parameter cancels all traces that apply to the specified user ID. The USER keyword is required only if the user ID is all numeric.

Description

Traces remain active until stopped using the TRACE STOP command. If messages are going to a printer spool file, you can close the file and start a new spool file by using the TRACE CLOSE command. If these is more than one trace defined for a *userid* / *macroname* combination, CA VM:Director uses only the first trace definition against the macro.

Each TRACE command that specifies the PRINT parameter uses a new virtual printer device on the service virtual machine. CA VM:Director imposes a maximum of 240 virtual printers in the address range of 0010 - 00FF for use with trace output going to spooled printer files. If this limit is reached and a virtual printer cannot be defined, the TRACE command waits until a virtual printer is made available.

If a macro to be traced is run from the CA VM:Director service virtual machine, the TRACE command ignores the LINES, USER, and TO parameters, and types the trace messages to the service virtual machine console.

If a macro you are tracing calls a user exit written as an EXEC, the TRACE command will not trace the user exit. You can trace user exits only by inserting the appropriate trace function in the user exit itself. User exit trace output always goes to the service virtual machine console.

Note: (CA VM:Secure only) Tracing a macro's execution reveals passwords being checked or set. However, if the CA VM:Secure Password Encryption Facility is being used, only the encrypted passwords are revealed.

Example

User ID MAINT wants to trace the CHGMDISK command every time it is used by user ID MARY. Enter:

```
vmdirect trace macro chgmdisk user mary to maint print
```

Each time MARY issues the CHGMDISK command, CA VM:Director sends the trace messages to a new spool file in MAINT's reader queue.

TRANSFER Command

Use the TRANSFER command to transfer a minidisk from one directory entry to another. Either directory entry may have type USER, IDENTITY, or SUBCONFIG.

```
TRANSFER  oldentry oldvaddr newentry newvaddr
```

Definitions

oldentry

Identifies the directory entry containing the minidisk to transfer.

oldvaddr

Identifies the virtual address of the minidisk to transfer.

To change only the virtual device address of the minidisk, specify the same user ID for *oldentry* and *newentry*.

newentry

Identifies the directory entry receiving the transferred minidisk.

newvaddr

Identifies the new virtual address to assign to the minidisk under the new owner.

Description

The TRANSFER command transfers ownership of a minidisk from one user ID to another or changes just the virtual device address of a minidisk.

CA VM:Director automatically updates all directory links to the old minidisk to point to the new minidisk. It removes any pre-existing directory links to the new minidisk. CA VM:Director transfers minidisk passwords with the minidisk.

You cannot use the TRANSFER command to transfer a user ID and minidisk defined on an IGNORE record in the DASD CONFIG file.

Return Codes and Error Messages

The following table lists return codes and error messages for the TRANSFER command.

Return Code	Message Number	Text
0	none	TRANSFER command completed normally
24	0038E	MISSING PARAMETER
24	0304E	ADDRESS *vaddr* IS AN INVALID DEVICE ADDRESS
24	0309E	DEVICE *newvaddr* IS A [T-DISK | V-DISK]
24	0311E	DEVICE *vaddr* IS NOT A MINIDISK
24	0355E	USER *newuserid* ALREADY HAS A *newvaddr* DEVICE
28	0356E	USER *userid* DOES NOT EXIST
28	0334E	MINIDISK *vaddr* DOES NOT EXIST
40	0310E	DEVICE *vaddr* IS AN IGNORED MINIDISK
80	0466E	SCRATCH EXIT INDICATES MINIDISK *vaddr* MAY NOT BE REMOVED
80	0506E	*olduserid* NOT ALTERED. REFERENCED IN PROFILES: *profname1 profname2* ...

Examples

- To transfer the CLIFFC 193 minidisk to user ID TESTID as the TESTID 191 minidisk, enter:

  vmdirect transfer cliffc 193 testid 191

- To change the virtual address of the FINANCE 191 minidisk to 194, enter:

  vmdirect transfer finance 191 finance 194

- To change the virtual device address of the FRED 391 minidisk to 222:

  vmdirect transfer fred 391 fred 222

Note: For information about defining minidisks that cannot be transferred, see the IGNORE Record (see page 375).

ULIST Command

Use the ULIST command to list directory entries available for management functions.

```
ULIST [entryname [mgrid]] [( [FILE | NOSFS]]
```

Definitions

[*entryname* [*mgrid*]]

Specifies directory entry names to list, and directory managers whose entries to list. Pattern-matching syntax used by the CMS LISTFILE command is valid for specifying the entry names. An asterisk for *entryname* lists all entries managed by the specified directory manager. An asterisk for *mgrid* lists the specified directory entry names for all directory managers.

If you do not specify *entryname* or *mgrid*, the command displays all entries that you manage.

[FILE]

Writes the results of the ULIST command to a file on the issuing user ID's A-disk. The filename is the user ID of the issuing user; the filetype is ULIST.

[NOSFS]

Excludes SFS information from the results of the ULIST command, displaying instead **N/A** for any SFS-related information. This option speeds up execution of the ULIST command because gathering SFS data can be time consuming.

Description

The ULIST command displays the User List Menu, which looks like the one shown in the following User List Menu section.

It allows directory managers to list directory entries they manage and perform full-screen directory management functions on those entries. It also provides information about the status of logon passwords and the number of days since logon passwords have been changed.

When an asterisk is displayed in the PwAge column, it indicates that the associated user ID does not have a *PW= special comment in its associated directory entry. Therefore, there is no associated password age information to report.

If you specify the user IDs to include in this display, but CA VM:Director authorization checking indicates that you do not have ULIST authorization over some user IDs that fit this specification, the last line of this screen displays the following message:

```
ADDITIONAL USERS EXCLUDED BY AUTHORIZATION CHECK
```

For example, you request the list of F* user IDs that CARLAT manages, but you are authorized to use the ULIST command on only one of the three user IDs she manages that begin with the letter F. CA VM:Director displays this one name on the list, and also displays this message at the bottom of the screen.

CA VM:Director User List Menu

```
ULIST                          User Menu List                   VM:Director
----------------------------------------------------------------------------

Userid    Manager  Group  PwStat PwAge Disks    Alloc  SFS Pools  Usage

FRASIERC  CARLAT          Exprd    315     1     300K    1         42.0K
FREDERIC  CARLAT                     1     1    1000K    1         50.0K
LILITHS   CARLAT                    24     1    1150K    1          4.0K
NICKT     CARLAT          Nolog     25     1     900K
REBECCAH  CARLAT                    55     1    2500K    1        500.0K




Copyright (c) 2011 CA. All rights reserved.
----------------------------------------------------------------------------
PF:  1=Cancel      2=Review    3=Quit       4=Edit     5=Manage   6=User
PF:  7=Backward    8=Forward   9=Mdskscan  10=Print   11=Sort    12=End
```

PWStat

Status of each user password.

PWAge

Age in days for each user password.

Disks

Number of minidisks allocated to each user.

Alloc

Amount of disk space allocated to each user.

SFS Pools

Number of SFS file pools where each user is enrolled.

Usage

Amount of SFS storage each user has in use.

With PF2 (Review), PF4(Edit), PF5 (Manage), PF6 (User), and PF9 (Mdskscan) you can display directory management screens. Place the cursor under the user ID you want to manage and press the PF key for the desired function. You may or may not be authorized to perform all of the functions.

You can sort the data in a column by placing the cursor in the field you want to sort on and pressing PF11 (Sort).

Password Status Values

The following table describes the Password status values:

Password Status (PwStat Value)	**Meaning**
Autlg	The logon password is AUTOONLY. The user ID can only be autologged and cannot be logged on directly.
Exprd†	The logon password has been expired with the EXPIRE command.
Logby	The logon password is not expired and is LBYONLY. This password forces users to log on to this user ID using the IBM-provided LOGONBY feature. It cannot be logged on directly.
Nolog	The logon password is NOLOG. This user ID is not allowed to log on.
Nopas	The logon password is NOPASS. The user can log on without specifying a password.

† When this value is listed as a user ID's password status, the user ID is prompted for a new password at next logon.

Examples

- To display all user IDs that begin with the characters TEST, enter:

  `vmdirect ulist test* *`

- To list the user IDs for manager FARFEL, enter:

  `vmdirect ulist * farfel`

UNLOCK Command

Use the UNLOCK command to unlock a user ID, profile, or CMS file that was previously locked by the LOCK command.

```
UNLOCK {"Parms"} [("Options"]

Parms:

  FILE filename filetype filemode
| PROFILE profile
| USER userid

Options:

[FILE [filename]]
[MSG | NOMSG ]
[STACK [FIFO | LIFO]]
[FIFO]
[LIFO]
```

Note: Options can be entered in any order.

Definitions

FILE *filename filetype filemode*

Specifies the CA VM:Director file to unlock. Pattern matching in the file specification is not allowed.

PROFILE *profile*

Specifies the profile to unlock.

USER *userid*

Specifies the user ID whose directory entry to unlock.

[FILE [*filename*]]

[MSG | NOMSG]

[STACK [FIFO | LIFO]]

[FIFO]

[LIFO]

These options are explained in Common API Command Options (see page 36).

Description

CA VM:Director writes permanent locks to a CMS file called PERMLOCK FILE A on the CA VM:Director A-disk when it creates the locks. The UNLOCK command looks for two permanent locks to clear for each user ID, profile, or CMS file being unlocked. For user IDs and profiles, the locks have the form:

```
SHARED DISK filename filetype filemode
PRIVATE LOCKUSER userid | profid
```

For CMS files, the form is:

```
SHARED DISK filename filetype filemode
PRIVATE LOCKFILE filename filetype filemode
```

When processing a user or profile entry, the LOCK and UNLOCK commands use the same internal locks as the GETENTRY and REPENTRY commands. Thus, UNLOCK can remove the locks that a GETENTRY command with the LOCK option creates. You may use the UNLOCK command if you decide not to update a user ID or profile after retrieving it with GETENTRY and the LOCK option.

Return Codes and Error Messages

The following table lists the return codes and error messages for the UNLOCK command.

Return Code	Message Number	Text
1	0127E	USER *userid1* EXCEEDED USER *userid2 type* PASSWORD LIMIT {FOR *vaddr*}
2	0038E	MISSING PARAMETER
3	0531E	PARAMETER *userid* IS TOO LONG
4	0643E	EXTRA PARAMETERS *parameters* WERE ENTERED AND ARE NOT ALLOWED
10	0039E	INVALID PARAMETER *parameter*
11	0265E	NOT AUTHORIZED FOR: *authorization words*
12	0349E	PASSWORD IS INCORRECT
13	0018E	INVALID OPTION *option*
14	0080E	USER *userid* DOES NOT EXIST
15	0684E	{ USERID | PROFILE } *name* IS NOT LOCKED
16	0685E	FILE *filename filetype filemode* IS NOT LOCKED

Return Code	Message Number	Text
19	0270W	PROFILE *profid* DOES NOT EXIST
20	0038E	MISSING PARAMETER
28	0681E	FILE *filename filetype filemode* DOES NOT EXIST

Note:

- For explanations of the message routing and disconnect options, see Common API Command Options (see page 36).
- For complete descriptions of all messages listed in this section, see the *Message Reference Guide*.

USER Command

Use the USER command to display the User Selection Menu or the entry screens of one of the selections available from that menu.

```
User [number [vaddr]]
```

Definitions

[*number* [*vaddr*]]

Is the number of the selection on the User Selection Menu to use, and, for certain selections, the virtual address of a minidisk on which to use the selection. Menu selections are:

Number	Selection	Accept *vaddr*?
1	Logon Password and Storage Size	—
2	Minidisk Link Mode and Passwords	3
3	Virtual Machine Options	—
4	Logical Line Editing Symbols	—
5	IPL System Name and Parameters	—
6	Screen Colors and Highlighting	—
7	Define a Link to Another User's Minidisk	3
8	Review/Remove Links By Other Users	3
9	Account Number and Distribution Code	—
10	Delete a Link to Another User's Minidisk	3
11	Review the Directory Entry	—

If you do not specify a menu selection or a virtual address, the User Selection Menu is displayed.

The default virtual address is 191.

Description

The USER command displays screens that let users modify their own directory entries.

This command requires an IBM 3270 display or compatible terminal.

You can authorize use of each of the selections individually so that users have access to only some of them.

Chapter 3: Configuration File Reference

This chapter provides reference information for the individual records in the configuration files used by CA VM:Director.

About Configuration File Records

You can create comments in the configuration file by placing an asterisk (*) in the first non-blank character in a record.

Configuration file records can be up to 71 characters long. To continue a record past the 71st character, specify a comma as the last non-blank character in the record. This character continues the logical line to the next record.

A comma at the end of a comment line does not serve as a continuation character. For example:

```
* THIS IS A COMMENT LINE,
```

However, if a continuation character ends the line preceding a comment line, the asterisk is not a comment character. For example:

```
GRANT RULES USER,
*SELF TO KAREN
```

Is the same as:

```
GRANT RULES USER *SELF TO KAREN
```

The following table lists each record along with its configuration file:

Record	Configuration File
ACCESS	PRODUCT
ALTERNAT	PRODUCT
AUDRECFM	PRODUCT
AUTOPASS	SECURITY
DEVTYPE	DASD
DIRECT	PRODUCT
DUMP	PRODUCT

Record	Configuration File
ESM	PRODUCT
EXTENT	DASD
GRANT	AUTHORIZ
IGNORE	DASD
IPLDISK	PRODUCT
JOURNAL	SECURITY
LIST	AUTHORIZ
MACLOAD	PRODUCT
MAXOLDPW	PRODUCT
MESSAGE	PRODUCT
MSGCASE	PRODUCT
NOLINK	DASD
PRODUCT	PRODUCT
PWSUPRES	SECURITY
RESID	PRODUCT
SERVANT	PRODUCT
SSINODE	PRODUCT
SUBPOOL	DASD
SYSOPER	PRODUCT
USEREXIT	PRODUCT
VOLUME	DASD
WITHHOLD	AUTHORIZ

ACCESS Record

Use the ACCESS record to identify the minidisks used by the CA VM:Director product server.

`ACCESS` *`name vaddr filemode`*

Configuration File

ACCESS is defined in the PRODUCT CONFIG file.

Definitions

name

Specifies the name of one of the CA VM:Director server minidisks. This name must also match the disk label found at the specified minidisk address. You specify one of the following values for minidisk:

- AUDT to specify the audit file minidisk
- BKUP to specify the source directory backup minidisk
- COMMON to specify the Servant Facility common minidisk.

  The product server dynamically allocates and initializes a new COMMON minidisk every time you save changes to the PRODUCT CONFIG file.

  The COMMON disk contains all modules and EXECs needed by the Servant Facility to manage minidisks. The COMMON disk is re-initialized automatically when the product initialization occurs and when the PRODUCT CONFIG file changes.
- DRCT to specify the source directory minidisk
- HOLD to specify the minidisk containing directory entries which are in held status.
- LOG to specify the Single System Image transaction logging minidisk

Note: For more information about the minidisks used by CA VM:Director, see Backing up CA VM:Director, in the chapter "Managing CA VM:Director" in the *Administration Guide*.

vaddr

Specifies the virtual address of the minidisk to be accessed.

filemode

Specifies the access mode letter to be used when accessing the minidisk.

Description

An ACCESS record must exist for each of the three required database minidisks, DRCT, BKUP, and HOLD. These database minidisks must contain only files placed there by CA VM:Director. Do not allocate the DRCT and BKUP minidisks on the same physical DASD. The BKUP minidisk, in combination with backups you run on a regular basis, is used to restore the current CA VM:Director directory database if a hardware failure results in the loss of the DRCT minidisk.

Note: Do not directly modify files on the BKUP, DRCT, or HOLD minidisks while CA VM:Director is running. Use the appropriate CA VM:Director command to update any files on these minidisks; otherwise, unpredictable results may occur.

(CA VM:Secure only) You must have an ACCESS RULE record in the PRODUCT CONFIG file to activate the Rules Facility.

Examples

- These ACCESS records identify the three required directory database minidisks with the recommended virtual addresses and access modes:

  ```
  ACCESS DRCT 1B0 U
  ACCESS BKUP 1B1 V
  ACCESS HOLD 1B2 H
  ```

- These ACCESS records identify the two optional minidisks with the recommended virtual addresses and access modes:

  ```
  ACCESS AUDT 1D0 T
  ACCESS COMMON 194 C
  ```

- This ACCESS record (CA VM:Secure only) identifies the optional Rules Facility minidisk, with the recommended virtual address and access mode:

  ```
  ACCESS RULE 1B4 R
  ```

ALTERNAT Record

Use the ALTERNAT record to specify commands to be used by various CA VM:Director macros in place of the CMS DDR, COPYFILE and FORMAT commands.

```
ALTERNAT command altcmd
```

Configuration File

ALTERNAT is defined in the PRODUCT CONFIG file.

Definitions

command

Specifies one of the CMS commands, COPYFILE, DDR, or FORMAT, for which an alternate command is to be used.

altcmd

Specifies alternate command module to be used by CA VM:Director instead of COPYFILE, DDR, or FORMAT.

Note: The altcmd specified must have the exact command syntax as the COPYFILE, DDR, or FORMAT command being replaced; if not, you must also implement the DISKCOPY or FORMAT user exit.

Description

Some of the directory manager functions use the CMS COPYFILE, DDR and FORMAT commands. Therefore, any alternate commands should reside on minidisks common to directory managers and the CA VM:Director service virtual machine, for example, the S- or Y-disk.

If you are using the Servant Facility, please note the following:

- To add a FORMAT command to the ALTERNAT record, you must first END CA VM:Director and XEDIT the PRODUCT CONFIG file. You cannot use the CONFIG command.
- The alternate command modules are automatically copied to the CA VM:Director COMMON minidisk.

- The COMMON DISK is re-initialized during product initialization and when the PRODUCT CONFIG file is updated.
- New copies of the programs defined on the ALTERNAT record are copied to the COMMON disk when the product initialization occurs and when the PRODUCT CONFIG file changes.

Example

To specify that you want to use the CA VM:Backup HiDRO fast minidisk formatting capabilities, add the following ALTERNAT records to the PRODUCT CONFIG file:

```
ALTERNAT FORMAT HIDRO
LTERNAT COPYFILE HIDRO
ALTERNAT DDR HIDRO
```

Implement the DISKCOPY and FORMAT user exits, then copy the HIDRO MODULE to a minidisk accessible by CA VM:Director. The prototype exits provided on the CA VM:Director system administrator minidisk (which is typically the VMDIRECT 176 minidisk) illustrate the use of CA VM:Backup HiDRO.

AUDRECFM Record

Use the AUDRECFM record to specify the record format of the CA VM:Director AUDIT file.

```
AUDRECFM {F | V}
```

Configuration File

AUDRECFM is defined in the PRODUCT CONFIG file.

Definitions

F

Specifies that the CA VM:Director AUDIT file will have a fixed length record format with 80 character records. If no AUDRECFM record is present the default file format is fixed length records, for compatibility with previous releases.

V

Specifies that the CA VM:Director AUDIT file will have a variable length record format with a minimum record length of 80 characters and a maximum record length of 256 characters per record.

Description

The VMDIRECT AUDIT file exists in one of two administrator selected formats. The traditional AUDIT file consists of fixed-format, 80-byte records that contain common and variable information. The enhanced AUDIT file uses variable length record format to contain up to 256 bytes of information per record. New information for some AUDIT codes will be added in new fields which start in column 81. Such new information will not be available when fixed length AUDIT records are configured.

During startup, any existing file on the AUDIT disk must have a record format which matches the AUDRECFM specification, otherwise the CA VM:Director server will not come up. However, the administrator may change the record format, while CA VM:Director is down by issuing a simple COPYFILE command, after accessing the AUDIT file as the A disk in the VMDIRECT server machine. To convert the file from FIXED format to VARIABLE, issue:

```
COPYFILE VMDIRECT AUDIT A ( RECFM V LRECL 256
```

To convert the file from VARIABLE to FIXED format, issue:

```
COPYFILE VMDIRECT AUDIT A ( RECFM F LRECL 80 FILL 40
```

Converting from VARIABLE format back to FIXED format will cause the loss of any record fields after column 80.

Note: For more information about the record fields, see the AUDITEXT command (see page 71).

AUTOPASS Record

Use the AUTOPASS record to force minidisk passwords to automatically be the same as logon passwords.

```
AUTOPASS password
```

Configuration File

AUTOPASS is defined in the SECURITY CONFIG file.

Definitions

password

Specifies a one- to eight-character site-defined keyword to appear in the place of a minidisk password. This tells CA VM:Director to require the logon password of the minidisk owner to be used as the minidisk password.

Note: If a user must enter a password to link to a minidisk, and the minidisk password is the specified keyword, the requesting user must use the logon password of the owning user ID.

Description

By including the AUTOPASS record in the SECURITY CONFIG file, the minidisk password changes every time the logon password changes. This means fewer passwords to remember and ensures that all passwords are changed regularly with no additional effort.

(CA VM:Secure only) If CA VM:Secure is configured for password phrase support, the AUTOPASS record cannot be used. A procedure to remove it is described in the password phrase support appendix in the *Administration Guide*.

You can use the MDISK user exit to force the use of this feature.

If you implement this AUTOPASS feature at your site, do not remove it without warning users who are using this feature; they will need to establish new minidisk passwords.

CA VM:Director places the logon password in the MDISK password field of the CP object directory.

The source directory contains the keyword. If you back up the source to a CP source directory file and use the CMS DIRECTXA command against it, the CP object directory will contain the literal AUTOPASS password for the minidisk passwords instead of the user's logon password.

Note: For more information, see MDISK User Exit in the chapter User Exit Reference (see page 405).

DEVTYPE Record

Use the DEVTYPE record to customize the name, attributes, or space characteristics of a specific device type.

```
DEVTYPE name {"ParmA" | "ParmB"} [("Options"]
```

```
ParmA:
```

```
prototype  [blksize [units]]
```

```
ParmB:
```

```
X'hhhh' blksize units tracks bytes num512blks  num1Kblks
num1Kblks num2Kblks num4Kblks [ sector ...]
```

```
Options:
```

```
[CACHE | NOCACHE]
[CYL0RES pages]
[FBA]
[MDC | NOMDC]
```

Configuration File

DEVTYPE is defined in the DASD CONFIG file.

Definitions

name

Specifies the customized device type name. You can make the name anything that is meaningful to your installation. See the Name column in the table for the device type names supplied by CA VM:Director.

prototype

Identifies a device type that CA VM:Director supplies and that you want to use as a model for the new device type. To create the new device type, CA VM:Director uses all values defined for the prototype device, except those values that the *blksize* and *units* parameters or the options override.

If you specify the prototype form, as opposed to the X'*hhhh*' form of the record, then the *blksize* and *units* parameters are optional. You can use them to override the values found in the prototype. However, you cannot specify any of the other parameters. You can still include any options that apply.

X'hhhh'

Identifies the device type as a new one with no prototype definition. Replace *hhhh* with the new device's two-byte device class and type, specified in hexadecimal. (IBM defines these values.)

The type codes for existing devices differ between different operating systems. Be sure to use the correct codes for your running system.

For the *name* parameter, specify the device name as the true device type that the MDISK directory record supports. If you do not, some commands using this device name may fail.

If you specify the X'*hhhh*' form, as opposed to the prototype form of the record, you must also specify all remaining parameters except for the *sector*1 ... *sector*n parameters. Those parameters are only required if the defined device type will be used to contain the online directory.

blksize

Specifies a default CMS blocksize that CA VM:Director uses whenever you format a minidisk. You can override this default on input screens and on some commands. Valid values are 512, 1024, 2048, 4096, 1K, 2K, and 4K.

units

Specifies the maximum number of allocatable units (cylinders or blocks) described by EXTENT records in the configuration file.

bytes

Specifies the number of bytes on each track of CKD devices. The number must be greater than 0 and less than 1000000. For current FBA devices, the number is always 512.

Note: For the following parameters, CA VM:Director validates the space characteristic by verifying that the specified number for each size block multiplied by the blocksize does not exceed the number of bytes per allocatable unit.

num512blks

Specifies the number of 512-byte blocks for each allocatable unit. This number must be less than 10000.

num1Kblks

Specifies the number of 1K-byte blocks for each allocatable unit. This number must be less than 1000.

num2Kblks

Specifies the number of 2K-byte blocks for each allocatable unit. This number must be less than 1000.

num4Kblks

Specifies the number of 4K-byte blocks for each allocatable unit. This number must be less than 1000.

sector ...

Specifies a list of the optional, one-byte Rotational Position Sensing (RPS) values required for CKD devices to contain the online directory.

The RPS values in the sector list represent the physical locations on the disk surface where CA VM:Director can read or write directory pages. CA VM:Director requires these values when a disk of this device type will contain the online directory.

Note: CA VM:Director does *not* verify these codes. Because you cannot calculate the values, contact Technical Support for the correct values to enter for any new device type.

FBA devices do not support RPS, and you cannot use this parameter with them. However, you can still use FBA devices to contain the online directory on systems that support them.

CACHE | NOCACHE

Indicates that the device type supports caching (CACHE) or does not support caching (NOCACHE).

CYL0RES *pages*

Specifies the number of pages to reserve for allocation information when the object directory begins in cylinder 0. Specify the CYL0RES option only if you are creating a new device type that will contain the object directory, and if the object directory will start in cylinder 0 of the new device.

You do not need to specify this option if the new device is an FBA device, or if the object directory will not start in cylinder 0 of the new device.

FBA

Indicates that the device type is a fixed block architecture (FBA) device. This option is required when you are defining an FBA device.

MDC | NOMDC

Indicates that caching uses expanded memory to cache the minidisk, regardless of the DASD device the minidisk is on (MDC), or that no expanded memory caching is to be used for the minidisk (NOMDC).

Description

The DEVTYPE record provides support for a new device type between releases of CA VM:Director or makes device type names more meaningful or more usable. For example, if you change 3390-3 to 3390-9, CA VM:Director rechecks existing VOLUME and EXTENT records to ensure the update does not invalidate them.

The DEVTYPE record enables you to customize device types by specifying either a prototype device type or an explicit device type. Differences between the two methods are explained in the parameter definitions.

You can redefine entries that CA VM:Director supplies and change the default block size or limit the number of allocatable units for a device type. For example, if you use the name TEST90 for a 3390 with only 200 cylinders, you can also use the available options to override the defaults that CA VM:Director specifies.

The DEVTYPE record is optional as long as the device types that you use are included in those that CA VM:Director supplies. This is the case for most installations, except when a new device type is announced. When CA VM:Director includes support for new device types in a subsequent release, that DEVTYPE record will be unnecessary.

Limitations Using CACHE | NOCACHE and MDC | NOMDC

CA VM:Director automatically adds MINIOPT records when it updates an object directory. Each time a directory update is made, CA VM:Director checks each MDISK record in the directory entry against the caching attributes (CACHE|NOCACHE or MDC|NOMDC) on the DEVTYPE record. If no MINIOPT record is present and the specified attributes for the device type are not the CP default, CA VM:Director adds a MINIOPT record with the caching attributes specified on the DEVTYPE record.

For example, suppose you have a DEVTYPE record similar to this:

```
DEVTYPE 3390 3390 (NOCACHE NOMDC
```

The CP default for z/VM systems is CACHE MDC. If you issue the CA VM:Director EDIT command for a directory entry, CA VM:Director updates the directory. If there is currently a MINIOPT record following an MDISK record in the directory entry, no changes are made to this MINIOPT record. However, if there is no MINIOPT record following an MDISK that defines a minidisk on a 3390, CA VM:Director inserts the following MINIOPT record after the MDISK record:

```
MINIOPT NOCACHE NOMDC
```

It also updates the object directory and saves the source directory entry on the CA VM:Director DRCT (1B0) minidisk.

For CA VM:Director initialization, the scenario may be different. If a SOURCE start is invoked, CA VM:Director opens each directory entry. If a MINIOPT record is needed for caching, CA VM:Director sets the correct caching attributes in the object directory. However, the source directory entry is *not* normally changed and saved on the CA VM:Director DRCT (1B0) minidisk. The source directory does *not* contain the MINIOPT record even though the object directory entry is set with the correct caching attributes. You can use the SAVEDIR startup option to change the normal procedure and make CA VM:Director update all of the source directory entries on the CA VM:Director DRCT minidisk. For more information about SAVEDIR startup option, see Changing Startup Options in the chapter "Introduction" in the *Administration Guide*.

A directory update is not invoked during a quick start. Therefore, if the caching attributes on the DEVTYPE record are changed while CA VM:Director is down and a quick start is invoked at initialization, CA VM:Director does not make any changes to the online directory to reflect the new caching attributes of the device type. However, subsequent directory updates will recognize the new DEVTYPE record.

The following table lists all device types that CAVM:Secure support:

Name	Default Block Size	Units (Cylinders or Blocks)	Tracks per unit	Blocks per unit	CMS 512 byte units	Per 1K units	Unit 2K units	4K units	ATTR
3390	4096	65520	15	57326	735	495	315	180	C
9336	1024	2147483640	1	512	1	0	0	0	F
FB-512	1024	2147483640	1	512	1	0	0	0	F

DIRECT Record

Use the DIRECT record to identify the CP-owned volume that contains the page-formatted directory area for the CP object directory.

```
DIRECT vaddr volser [volser2 [volser3 [volser4]]]
```

Configuration File

DIRECT is defined in the PRODUCT CONFIG file.

Definitions

vaddr

Specifies the virtual address of the full-pack minidisk (or a read/write link that the CA VM:Director virtual machine has to the full pack minidisk) containing the CP object directory. Be sure that CA VM:Director has at least an MR link to the minidisk specified by this virtual address. If another user has a Read-Write LINK to this minidisk, and that user might logon before the product server is started), then it may be necessary to grant the CA VM:Director product server an MW LINK to the Object Directory volume.

***volser* [*volser2* [*volser3* [*volser4*]]]**

Specifies a list of up to four volume serial numbers of the real DASD that contains the CP object directory. Usually a single volume serial number is defined, but in a Single System Image complex, a unique volume serial is specified for each member in the complex.

Description

The volume(s) specified by the DIRECT record are usually a system residence volume. The minidisk specified by the virtual address is normally a full-pack minidisk; this minidisk must begin at real cylinder or block zero and extend through the entire directory area. The minidisk should cover the entire DASD.

Important! If other users are to share access to this minidisk, take extreme care to prevent multiple users from updating the CP object directory.

Do not use DIRECTXA, DIRMAINT, or other programs to modify the CP object directory while CA VM:Director is running. Using these programs may cause the CP object directory to become corrupted. If CA VM:Director detects that another program has modified the CP object directory, CA VM:Director immediately ends.

DUMP Record

Use the DUMP record to indicate which user ID will receive a VMDUMP if the CA VM:Director virtual machine abnormally terminates.

```
DUMP userid [CLEAR | NOCLEAR]
```

Configuration File

DUMP is defined in the PRODUCT CONFIG file.

Definitions

userid

Specifies the user ID to whose virtual reader the VMDUMP should be spooled.

CLEAR | NOCLEAR

Clears and releases (CLEAR) or does not clear and release (NOCLEAR) the buffers used for updating the CP directory. These buffers should be cleared before a dump is made to prevent passwords from appearing in the dump; however, they should not be cleared if this information is required for Technical Support to determine the problem.

Description

If the DUMP record is omitted, the dump is spooled to the user ID specified on the SYSDUMP parameter of the SYSOPR macro in HCPSYS.

ESM Record

Use the ESM record to activate external security manager command authorization.

```
ESM resourceclass resourceprefix
```

Configuration File

ESM is defined in the PRODUCT CONFIG file.

Definitions

resourceclass

Specifies the one- to eight-character name of the of resource class that will be specified on the CLASS parameter of a RACROUTE REQUEST=AUTH macro call.

resourceprefix

Specifies the one-to eight-character value for the first-level qualifier in the constructed resource name.

Note: For more information about using the ESM record with CA VM:Director, see the chapter "ESM Support for CA VM:Director" in the *Administration Guide*.

Description

The ESM record identifies the resource class name and associated tokens as specified by the system administrator.

Note: Only one ESM record is allowed per configuration file. If more than one ESM record is detected, message 495E is issued.

EXTENT Record

Use the EXTENT record to assign a contiguous block of disk space to one or more subpools.

```
[extentname] EXTENT beginning ending [subpool ... | * comment]
```

Configuration File

EXTENT is defined in the DASD CONFIG file.

Definitions

[*extentname*]

Specifies a one- to eight-character name associated with this extent description. The extentname field is optional and is only used by the IBM System Management API Server; for more information, see the appendix "Systems Management API Support" in the *Administration Guide*. Each extentname statement name must be unique and cannot be used by any other statement in any configuration file.

beginning

Specifies the first cylinder or block number in this extent.

ending

Specifies the last cylinder or block number in this extent.

***subpool* ...**

Specifies the names of up to eight subpools, separated by blanks that can allocate space in this extent.

*** *comment***

Identifies the fixed CP extent and provides documentation. The *asterisk* identifies this extent as a fixed CP extent, and the *comment* provides documentation about its use. These comments appear in volume maps produced by the MAP command. No subpool names can be assigned to a fixed CP extent; therefore, no space can be allocated from a fixed CP extent.

You do not need to identify disk space used by guest SCPs.

Description

The disk space in an extent is allocated through the subpools specified on an EXTENT record. (For more information about subpools, see SUBPOOL Record in the chapter "Configuration File Reference.")

CA VM:Director assumes that an extent resides on the DASD identified by the VOLUME record that immediately precedes an EXTENT record. This means that, in the DASD CONFIG file, each EXTENT record must appear immediately after its corresponding VOLUME record.

Note: For more information about volumes, see VOLUME Record (see page 401).

Additionally, SUBPOOL records must appear before they are referenced by an EXTENT record. You cannot continue an EXTENT record beyond one line.

Multiple extents can be defined for a single volume, but they cannot overlap. Minidisk definitions cannot cross over extent boundaries.

Minidisks defined outside extents are not controlled by CA VM:Director; overlapping minidisk definitions are not detected for these minidisks. All disk space used for CMS minidisks should be identified by EXTENT records.

Note: Minidisk extents can start in cylinder 0 on CKD DASD or in blocks 0 through 15 on FBA DASD. Use caution when using these cylinders or blocks because these areas are used by CP to hold the device's label and, on CP OWNED volumes, the allocation information. If the CMS FORMAT command is used against a minidisk that starts in these areas, it will re-label the entire device. The ability to define minidisks as cylinder or block 0 is intended for situations in which the device is dedicated to a second-level guest as a full pack minidisk.

Note: Additionally, minidisk extents should not cover alternate areas on a DASD. See the appropriate IBM planning and administration documentation for the location and size of alternate areas for each DASD.

Example

The following EXTENT records, immediately preceded by their VOLUME record, define volume VMPK11:

```
VOLUME VMPK11 3380
EXTENT  0       001   * CP ALLOC MAP AREA
EXTENT  2       401   SYSTEMS DEV1 TEST1
EXTENT  402     552   * CP PAGING AREA #1
EXTENT  553     884   MAIN SYS1 TEMP
```

GRANT Record

Use the GRANT record to authorize users to use CA VM:Director commands, utilities, and screen selections.

```
GRANT authority [OVER target-users] TO users [( "Options"]
Options:

GROUP
```

Configuration File

GRANT is defined in the AUTHORIZ CONFIG file.

Definitions

authority

Specifies the authorization to use an entire command, a command and some of its parameters, or a list of commands; *authority* can be any of the authorizations in the following tables:

- Any command or utility authorization, User Selection Menu (USE command) authorization, or Manager Selection Menu (MANAGE command) authorization from the Authorizations to Use CA VM:Director Commands and Utilities table in the *Administration Guide.*
- Any predefined variable list from the Predefined Variable Lists table in the *Administration Guide.*
- Any command processing authorization from the Authorizations for Command Processing table in the *Administration Guide.*
- Any lists you create using LIST records.
- For more information about how to use lists, see User ID Lists and Authority Lists in the chapter "Authorizations" in the *Administration Guide*. For more information about the LIST record, see LIST Record (see page 379).

  You can also use an authorization with a trailing asterisk (for example, MAN*) to indicate all authorizations that begin with the specified characters. Such authorizations will be flagged by the configuration file editor as unrecognized, but they are still functional.

OVER

Limits the scope of the *authority*; you can use the word OVER, to make it clear that the authority is valid only for user ID(s) that follow the word OVER.

target users

Specifies the user ID or a list of user IDs, over which the authority is granted to the *users.*

users

Specifies a list of user IDs, separated by blanks that receive these authorizations. A trailing asterisk (for example, M*) indicates that all user IDs beginning with the specified characters are authorized.

Description

Depending on your site's requirements, authorizations can be simple complex. For more information about setting up authorizations, see the chapter "Authorizations" in the *Administration Guide*. A user ID must be defined in the VMDIRECT MANAGERS file to use manage subfunctions that manipulate DASD. See the chapter on directory managers in the *Administration Guide* for more information.

Example

- Give user ID VMANAGER every possible authorization to make him a system administrator:

  ```
  GRANT * TO VMANAGER
  ```

- Permit BATMAN to use selection 3 (Select User Menu) from the Manager Selection Menu and all selections from the User Selection Menu to manage user IDs assigned to manager ROBIN. BATMAN is not allowed to use any other menu selection from the Manager Selection Menu over these users:

  ```
  GRANT MANSEL03 OVER *DIRUSRS OF ROBIN TO BATMAN
  GRANT USER OVER *DIRUSRS OF ROBIN TO BATMAN
  ```

- Authorize JETHRO, ELLIMAE, and JED to use any selection from the Manager Selection Menu for any user ID other than their own (users cannot use the Manager Selection Menu to manage their own user IDs):

  ```
  GRANT MANAGE *ALL TO JETHRO ELLIMAE JED
  ```

- Authorize every user ID to use selections 3, 4, and 6 from the User Selection Menu, but no others. First, add a LIST record to create a list of authorizations to grant, then grant the list to all users:

  ```
  LIST *USRSTUF USESEL03 USESEL04 USESEL06
  GRANT *USRSTUF OVER *SELF TO *ALL
  ```

- Permit only users TED and MAINT to assign class A privilege through the CLASS command:

  ```
  GRANT CLASS A TO TED MAINT
  ```

- Grant manage authority to a list of directory managers. First, add a LIST record to create the list, then grant manage authority to the list:

  ```
  LIST *MGRTEAM TLOOIS BABS JOEP KRISTEN KAYCEEP
  GRANT MANAGE OVER *DIRUSRS OF *SELF TO *MGRTEAM
  ```

- Authorize SCOTTY to use the site-written macro FINDMGR (site-written macros require special authorization):

  ```
  GRANT $FINDMGR TO SCOTTY
  ```

  Note that some keyboards ascribe a different hexadecimal representation to the dollar sign, which, in the U.S., is usually X'24.' In the United Kingdom, X'24' is the pound sign, so this GRANT record is written:

  ```
  GRANT #FINDMGR TO SCOTTY
  ```

- Grant a single MAINT MANAGE subfunction, DISKMOVE, over user ID SPOCK to manager KIRK:

  ```
  GRANT MAINTMAN SPOCK DISKMOVE TO KIRK
  ```

- Permit MAINT to use CPFMTXA command:

  ```
  GRANT CPFMTXA TO MAINT
  ```

  This allows MAINT to execute the CPFMTXA command:

  VMDIRECT CPFMTXA *parameters*

IGNORE Record

Use the IGNORE record to indicate which user IDs or minidisks that CA VM:Director is to ignore when it checks for overlapping minidisks.

```
IGNORE userid [vaddr]
```

Configuration File

IGNORE is defined in the DASD CONFIG file.

Definitions

userid

Specifies the user ID whose minidisks are to be ignored.

vaddr

Specifies the virtual device address of a specific ignored minidisk, owned by *userid*. If you do not specify a virtual device address, all minidisks owned by the specified user ID are ignored.

Note: To prevent accidental overlays, specify a virtual address to be ignored rather than defaulting to all minidisks for a user ID.

Note: If an IGNORE record identifies only a user ID and that user ID owns a minidisk that is not a full-pack minidisk that resides within a non-CP-owned extent defined to CA VM:Director, a new minidisk can be allocated on top of the ignored minidisk. All the files on the ignored minidisk will be destroyed.

Description

CA VM:Director does not detect a minidisk overlay involving ignored minidisks. Ignored user IDs and minidisks are ignored for a number of other CA VM:Director functions as well. For instance, CA VM:Director does not perform any of these functions on ignored minidisks or minidisks belonging to ignored user IDs:

- Moving or formatting
- Changing with the CHANGE command
- Transferring with the TRANSFER command
- Mapping with the MAP command
- Updating directory links if that minidisk is deleted or its owner or virtual address is changed

However, you can use CA VM:Director to remove an ignored user ID or an ignored minidisk from the directory. When an ignored user ID is removed from the directory, associated minidisks are not formatted.

Specify overlay definitions on an IGNORE record. Some sites define dummy minidisks that deliberately overlay CP extents or other minidisks. These overlay definitions are typically assigned to NOLOG user IDs and are not used in normal system operation.

Full-pack minidisks and minidisks defined within a fixed CP extent must be ignored. To protect your CP extents properly, you must define an EXTENT record with an asterisk.

Note: For more information, see EXTENT Record (see page 370).

You cannot continue IGNORE records beyond one line.

IPLDISK Record

Use the IPLDISK record to identify an IPL capable minidisk owned by the CA VM:Director service virtual machine.

```
IPLDISK vaddr
```

Configuration File

IPLDISK is defined in the PRODUCT CONFIG file.

Definitions

vaddr

Specifies the virtual address of the IPL capable minidisk owned by CA VM:Director.

Description

(CA VM:Secure only) If you have the Rules Facility installed, the IPLDISK record is not necessary. However, if you use the EXPIRE command to expire user IDs and you do not have the Rules Facility installed, the IPLDISK record is required because the password prompt issued to new user IDs or to user IDs that have been expired is produced by a program IPLed from this minidisk.

If you use the EXPIRE command to expire user IDs, the IPLDISK record is required because the password prompt issued to new user IDs or to user IDs that have been expired is produced by a program IPLed from this minidisk.

Note: For information about setting up the IPL capable program, see the VMXIPL Utility (see page 456).

JOURNAL Record

Use the JOURNAL record to control the monitoring of invalid password conditions.

`JOURNAL` *`warning`*

Configuration File

JOURNAL is defined in the SECURITY CONFIG file.

Definitions

warning

Specifies the number of consecutive invalid password attempts recorded for a user ID or device address after which a warning message appears on the system operator's console. The minimum is 0, the maximum is 99999999.

Description

The JOURNAL record monitors the following invalid password conditions:

- Invalid logon passwords entered while using CA VM:Director commands
- Invalid link password entered from the Define a Link to Another User's Minidisk selection on the User Selection Menu

Separate journal counters are maintained for each monitored condition. The counters are reset to zero whenever a password is correctly entered or when CA VM:Director is shut down; the appropriate counter is incremented by one whenever its password is incorrectly entered. If consecutive invalid attempts exceed the limit specified on this JOURNAL record, CA VM:Director prevents further attempts until someone issues the JOURNAL command with the RESET parameter against that password, or issues the REST command.

Example

The following record appears in the SECURITY CONFIG file:

```
JOURNAL 3
```

If JACK enters three consecutive invalid logon passwords, a message is sent to the system operator.

LIST Record

Use the LIST record to create a list of user IDs or authorizations to use on GRANT and WITHHOLD records.

```
LIST *listname {userids | authwords}
```

Configuration File

LIST is defined in the AUTHORIZ CONFIG file.

Definitions

listname

Defines a name for the list. The first character must be an asterisk; you can use up to seven alphanumeric characters for the rest of the name.

userids

Consists of a list of user IDs, separated by blanks that are to belong to *listname*. These user IDs will have the same authorizations when you use this list name in a GRANT or WITHHOLD record.

authwords

Consists of a list of single-word authorizations, separated by blanks that are to belong to *listname*. These authorizations will all be granted or withheld from one or more users when you use this list name in a GRANT or WITHHOLD record.

Description

The LIST record is optional. Using LIST records you can:

- Build lists that supplement the special authorizations provided with CA VM:Director.
- Nest lists created with one LIST record in lists created by another LIST record.
- Create user ID lists and authority lists in separate LIST records, then use these lists in a GRANT or WITHHOLD record.

Note: Verify any items associated with the LIST record; CA VM:Director does not validate LIST record items.

For more information about using lists in authorizations, see User ID Lists and Authorization Lists in the chapter "Authorizations" in the *Administration Guide*.

Examples

- When you put a command on a LIST record and grant that list of commands to a user ID, you are giving that user ID authority to use the command over all user IDs. For example, the MANAGE command is one of several commands specified for the *DIRCMDS authorization by the following LIST record, which you then use to grant *DIRCMDS authorization to ELROY, a manager at your site:

  ```
  LIST *DIRCMDS MANAGE QUERY PASSWORD
  GRANT *DIRCMDS TO ELROY
  ```

  The combination of this LIST record and this GRANT record is equivalent to the following record, and means that ELROY can manage anyone:

  ```
  GRANT MANAGE OVER *ALL TO ELROY
  ```

- In contrast to the previous example, assume the MANAGE command is not part of any authorization list, and, in fact, the authorization *DIRCMDS does not exist. On a GRANT record, give ELROY permission to use the MANAGE command:

  ```
  GRANT MANAGE TO ELROY
  ```

  This GRANT record is equivalent to the following record, and means that ELROY can manage only his own users:

  ```
  GRANT MANAGE OVER *DIRUSRS OF *SELF TO ELROY
  ```

- Define the list called *MYGRP that includes the user IDs USER01, USER02, USER03, and USER04:

  ```
  LIST *MYGRP USER01 USER02 USER03 USER04
  ```

- You want all users in the list *MYGRP, defined above, to be able to extract audit records, take CA VM:Director backups, and use the EXTRACT command. Define a list called *MYAUTHS that includes the CA VM:Director commands these user IDs should be allowed to use, then grant authority to *MYAUTHS to the user IDs in the list *MYGRP:

  ```
  LIST *MYAUTHS AUDITEXT BACKUP EXTRACT MAY
  GRANT *MYAUTHS TO *MYGRP
  ```

MACLOAD Record

Use the MACLOAD record to specify which command macros to load and keep in virtual memory.

```
MACLOAD {macroname ... | * }
```

Configuration File

MACLOAD is defined in the PRODUCT CONFIG file.

Definitions

***macroname* ...**

Specifies a blank delimited list of the filenames of the macros that are to be loaded into the virtual memory. You can use a trailing asterisk on a partial filename for pattern matching if your macro filenames are defined similarly. For example, ACCT* loads all macro files whose names begin with the letters ACCT.

To load all CA VM:Director macros, specify an asterisk instead of any macro name.

Description

System performance can be enhanced when you execute a macro that is loaded in virtual memory. For maximum performance, load all user and manager macros. The user and manager macros can be specified on the MACLOAD record as USE* and MAN*, respectively. If you are using SFS, specify SFS* on a MACLOAD record to load all of the SFS macros.

Although you can list the names of many command macros in a single MACLOAD record, you can have multiple MACLOAD records in the PRODUCT CONFIG file.

MAXOLDPW Record

Use the MAXOLDPW record to specify the number of previously used logon passwords to save for a user ID.

`MAXOLDPW` *`num`*

Configuration File

MAXOLDPW is defined in the PRODUCT CONFIG file.

Definitions

num

Num can be any number from 1 to 99. The default for *num* is 8. A *PW= record must be present in the directory entry before MAXOLDPW takes effect.

Description

A record is retained for the last *num* previous passwords or password phrases for a user ID. A PASSWORD User Exit may choose to prohibit any passwords or phrases in these records from being re-used.

MESSAGE Record

Use the MESSAGE record to control various aspects of CA VM:Director messages.

```
MESSAGE {"ParmA" | "ParmB" | "ParmC"}
```

```
ParmA:
```

```
{MSGNOH | MSG}
```

```
ParmB:
```

```
{MIXED | UPPER | LOWER}
```

```
ParmC:
```

`{ON | OFF | CODE | TEXT | ABEND}` *`msgnumber ...`*

Configuration File

MESSAGE is defined in the PRODUCT CONFIG file.

Definitions

MSGNOH | MSG

Indicates which CP message facility command should be used. It specifies that CA VM:Director should always use either the CP MSGNOH command (MSGNOH) or the CP MSG command (MSG) when required to send a message to another user with the CP message facility. MSGNOH requires that CA VM:Director have CP privilege class B.

Note: For information about the difference between the MSG and MSGNOH CP commands, see the appropriate IBM CP commands manual. If no value is specified, MSGNOH is used.

MIXED | UPPER | LOWER

Causes CA VM:Director to display messages in either mixed case (MIXED) or uppercase (UPPER). If no value is specified, mixed case is used.

ON | OFF | CODE | TEXT | ABEND

Controls the contents of the displayed message. The ON, OFF, CODE, TEXT, and ABEND levels are all mutually exclusive. The most recently requested level for a specific message number will override all previous settings for that message:

- **ON** indicates that the message should be displayed in its entirety, but still subject to the user's EMSG setting when appropriate. This is the default.
- **OFF** indicates that the message should be completely suppressed to all destinations.
- **CODE** indicates that only the message prefix (product prefix, CSECT name, message number) should be displayed.
- **TEXT** indicates that only the text of the message, and not the message prefix, should be displayed.
- **ABEND** indicates that CA VM:Director should abend as soon as the specified message has been formatted. This is typically only used during problem determination, and should not be specified unless requested by customer support.

Msgnumber ...

Specifies a list of the message identifiers controlled by this display specification. This option consists of 4 numeric digits, followed by one alphanumeric suffix.

Description

With the MESSAGE record you can set which CP message facility command should be used, the case of the text of the messages, and attributes of the messages themselves.

Messages sent by either command, MSG or MSGNOH, contain a standard 11-character message header.

Note: For an explanation of the message format, see CA VM:Director Messages (see page 28). For an explanation of each message, see the *Message Reference Guide*.

MSGCASE Record

Use the MSGCASE record to specify how CA VM:Director displays its messages

```
MSGCASE {MIXED | UPPER}
```

Configuration File

MSGCASE is defined in the PRODUCT CONFIG file.

Definitions

MIXED

Displays CA VM:Director messages in mixed case.

UPPER

Displays CA VM:Director messages in uppercase.

Description

The MSGCASE record specifies the case in which CA VM:Director displays its messages. This record has been superseded by the MESSAGE record and is retained for compatibility purposes only.

NOLINK Record

Use the NOLINK record to indicate which directory links are not to be mapped by CA VM:Director by identifying the target user ID and minidisk of a directory link to skip.

```
NOLINK userid [vaddr]
```

Configuration File

NOLINK is defined in the DASD CONFIG file.

Definitions

userid

Specifies a user ID that owns a frequently linked minidisk whose directory links you do not want mapped by CA VM:Director.

vaddr

Specifies the virtual address of a specific minidisk owned by *userid*. If the virtual address is not specified, then directory links to all minidisks owned by the specified user ID are not mapped.

Description

Not mapping directory links that are common to most user IDs (such as the CMS S- and Y-disk links) reduces storage and processing overhead.

During CA VM:Director initialization, a table is created in virtual memory that is used to map the LINK statements from all directory entries. Each entry in this table represents a single LINK statement and is currently 40 bytes in length. A site with 5,000 user IDs, where each user ID has links to the CMS S- and Y-disks, saves 400K of CA VM:Director virtual storage by not mapping these two disks.

Users who own minidisks that are specified on a NOLINK record cannot use Selection 8 on the User Selection Menu (Review or Remove Links By Other Users) to act on those directory links.

If you use the CA VM:Director ASSIGN, CHANGE, or TRANSFER command to change a user ID or minidisk that also appears on a NOLINK record, the directory links to that user’s minidisks are not automatically updated to refer to the new user ID.

Example

Add the following records to the DASD CONFIG file to prevent the links to MAINT's 190 and 19E minidisks from being mapped:

```
NOLINK MAINT 190
NOLINK MAINT 19E
```

PRODUCT Record

Use the PRODUCT record to identify other service virtual machines that are running CA products and activate the interface between CA VM:Director and these other service virtual machines.

PRODUCT *product userid* [MASTER]

Configuration File

PRODUCT is defined in the PRODUCT CONFIG file.

Definitions

product

Specifies the abbreviation of a CA product whose service virtual machine you want to identify to CA VM:Director. The CA products that you can specify and their abbreviations are:

Product	Abbreviation
VM:Account	VMACCOUN
VMSISERV	VMSISERV

If CA VM:Account is installed and you specify VMACCOUN, the CA VM:Director facility for changing account numbers is coordinated with CA VM:Account.

The PRODUCT VMSISERV record allows VMANAGER to perform component maintenance.

userid

Specifies the user ID of the service virtual machine, running the specified product.

[MASTER]

Identifies this service virtual machine as the master for the product; required if the product has more than one virtual machine defined to CA VM:Director.

CA VM:Director allows more than one virtual machine for each product to be defined in the PRODUCT CONFIG file and will supply answers to any virtual machine that makes a request. However, if CA VM:Director needs to make a request to a product with more than one virtual machine defined, it makes the request to the virtual machine that is defined as the master.

Description

To implement an interface between CA VM:Director and another CA product, the PRODUCT CONFIG file must have an appropriate PRODUCT record, and the other product's configuration file must contain a PRODUCT VMDIRECT record.

PRODUCT records are also used during processing of the SYSTEM macro. CA VM:Director routinely checks for links to any of its accessed minidisks, and sends warning messages to any users found to have a link. However, other CA products may need to have a link to them. Use of the PRODUCT record will prevent the warning message from being sent.

Note: For more information about performing product maintenance, see the chapter "Macros" in the *Administration Guide*. For more information about the CA product interfaces, see the *CA Mainframe VM Product Manager Product Interface Guide*.

Example

Implement the interface between CA VM:Account and VM:Director:

1. Edit the PRODUCT CONFIG file by entering:

   `vmdirect config product`

2. Add the following record to the PRODUCT CONFIG file, where *vmaccount* is the name of your CA VM:Account virtual machine:

   `PRODUCT VMACCOUN` *`vmaccount`*

3. File the changes.
4. Log on to the CA VM:Account server machine and end the server.
5. Edit the VMACCOUN CONFIG file and add the following record, where *vmdirect* is the name of your CA VM:Director service virtual machine:

   `PRODUCT VMDIRECT` *`vmdirect`*

6. Restart the CA VM:Account server.

PWSUPRES Record

Use the PWSUPRES record to suppress, or mask, CA VM:Director password display.

```
PWSUPRES
```

Configuration File

PWSUPRES is defined in the SECURITY CONFIG file.

Description

Using the PWSUPRES record to suppress the password disables the PA2 key from displaying a password with the USER or MANAGE commands, and the password is masked on the displays from the MAINT MANAGE and MAINT MDPW commands. However, you can still display passwords with the EDIT command.

(CA VM:Secure only) If you are using the Password Encryption Facility, password suppression is automatically in effect for the USER, MANAGE, MAINT MANAGE, and MAINT MDPW displays, so this record is not necessary.

RESID Record

Use the RESID record to switch CA VM:Director server communications from IUCV to APPC.

```
RESID resourcename [GLOBAL [SSIONLY]]
```

Configuration File

RESID is defined in the PRODUCT CONFIG file.

Definitions

resourcename

Specifies the name of an APPC resource. This resource name must also be specified on an IUCV*IDENT statement in the product server directory entry.

GLOBAL

Specifies that the named resource is known globally throughout the TSAF or ISFC collection. The GLOBAL parameter must also be specified on an IUCV *IDENT statement in the product server directory entry, if this operand is present.

SSIONLY

Specifies that APPC connections to the RESID are only accepted from VM systems residing in the SSI cluster as the VM system hosting the product server virtual machine.

Description

The RESID record is required if you are using the IBM Systems Management Application Programming Interface (API). The Systems Management API server must use APPC communications to the CA VM:Director server.

Note: For more information about using and setting up the Systems Management API, see the appendix "Systems Management API Support" in the *Administration Guide*.

The RESID record, with the GLOBAL operand, is required if you are operating in a Single System Image complex. The product server virtual machines on every member system communicate with each other using APPC.

SERVANT Record

Use the SERVANT record to enable or disable the Servant Facility.

```
SERVANT {ON | OFF} [prefix [time [numfree [numactive]]]]
```

Configuration File

SERVANT is defined in the PRODUCT CONFIG file.

Definitions

ON

Enables the Servant Facility. You must create the COMMON 194 minidisk and access it using the ACCESS record in the PRODUCT CONFIG file when you are enabling the Servant Facility.

OFF

Disables the Servant Facility. All servant machines that are not attached to a process are deleted. Those that are attached to a process will be deleted when the process ends.

prefix

Specifies the four-character prefix for the names of the servant user IDs. The default is VMX$. This prefix must be unique; no other user IDs on the system can start with this prefix.

time

Specifies the number of seconds a process will wait for a servant to connect before timing out. If a servant is started and does not connect to CA VM:Director within this amount of time, the process requesting the servant terminates. The minimum number of seconds is 0, the maximum is 99999999. The default is 50 seconds.

numfree

Specifies the number of servants to keep free. A free servant is one that has been started and has connected to CA VM:Director but is not being used by a process. The minimum is 0, the maximum is 25. The default is 2. If you reduce the number of free servants, the servants will not be logged off until the next time they are detached from a process, whether currently active or not.

numactive

Specifies the maximum number of servants that can be active at any time. This number can be any positive integer larger than *numfree*. The default is 25. If you reduce the maximum number of servants, the servants will not be logged off until the next time they are detached from a process, whether currently active or not.

Description

When the Servant Facility is enabled, CA VM:Director uses worker machines to aid in minidisk and SFS management. You must enable the servant facility if you are managing SFS with CA VM:Director. You can enable or disable the Servant Facility dynamically by using the CONFIG PRODUCT command to edit the SERVANT record. All changes take effect immediately.

To display the status of the Servant Facility, use the QUERY SERVANT command.

Important! Do not use the ASSIGN command with servant user IDs. The directory manager for the CA VM:Director service virtual machine and the servant virtual machines must be the same.

Note: For more information about using and setting up the Servant Facility, see the ACCESS Record (see page 353) in this document and the *Administration Guide*.

SSINODE Record

Use the SSINODE record to identify the names of members of a Single System Image complex, and to specify that the Object Directory is to be compiled in an SSI-Ready format.

```
SSINODE nodename
```

Configuration File

SSINODE is defined in the PRODUCT CONFIG file.

Definitions

nodename

Specifies name of a Single System Image member node.

Description

Add an SSINODE record naming each member of an SSI complex. Names specified on SSINODE records are allowed as operands for the ON keyword in a BUILD directory statement, or as the operand of an ON option keyword in an ADDENTRY command.

You must remove all BUILD statements referencing a specific node name value before you remove the SSINODE record which defines that member name.

SUBPOOL Record

Use the SUBPOOL record to define a CA VM:Director subpool and specify how CA VM:Director allocates disk space from it.

```
SUBPOOL name {"Method"} {HIGHEND | LOWEND} maxspace [EXPLICIT]
```

```
Method:

  ROTATING
| LINEAR
| LARGEGAP
| SMALLGAP [nK | nM]
```

Configuration File

SUBPOOL is defined in the DASD CONFIG file.

Definitions

name

Specifies a unique name that identifies the subpool you are defining.

ROTATING

Allocates disk space to minidisks in a first fit fashion across all volumes having extents in the subpool. CA VM:Director allocates the first minidisk on the first volume that has an extent in the subpool with sufficient space for the minidisk. It allocates the second minidisk on the next volume that has an extent in the subpool with sufficient space for the minidisk, and so on. The HIGHEND | LOWEND parameters determine allocations within an extent on a volume.

LINEAR

Allocates disk space to minidisks in a first fit fashion, from the first volume that has an extent in the subpool with sufficient space for the minidisk. CA VM:Director searches the volumes in reverse order of their definition in the DASD CONFIG file. It allocates subsequent minidisks to the same volume until its extents that are in the subpool are full or the required space is not available. Then CA VM:Director uses the next volume that has extents in the subpool. The HIGHEND | LOWEND parameters determine allocations within an extent on a volume.

SMALLGAP [*n*K | *n*M]

Allocates disk space to minidisks in a best fit fashion from the smallest gap of available space on any volume that has an extent in a subpool. CA VM:Director searches volumes in reverse order of their definition in the DASD CONFIG file. It repeats the search for a best fit for each subsequent minidisk allocation. If you specify the *n*K parameter, then the gap must be large enough for the minidisk you request and the amount of additional disk space you specify with this parameter. CA VM:Director only tests for this minimum amount of remaining space if it cannot locate a gap in the subpool that exactly matches the size of the requested minidisk. If you do not specify this parameter, CA VM:Director does not check for remaining space. Replace *n* with the number of kilobytes (K) or megabytes (M) you want remaining in the gap after CA VM:Director allocates a minidisk you request. Replacing *n* with zero is the same as not specifying the parameter.

The best fit is always an exact match in size between the requested disk space and an available gap. If CA VM:Director cannot find an exact match, it looks for the next best fit which is the smallest gap that has sufficient available space. If two or more locations in a single extent satisfy the best fit criteria, then the HIGHEND | LOWEND options determine which one CA VM:Director uses. If no gaps are large enough to allocate the requested space and leave the required minimum amount of space remaining, the allocation reverts to LARGEGAP rules. Thus, the largest amount of disk space possible remains after the allocation.

Note: SMALLGAP and LARGEGAP can sometimes help reduce the amount of fragmentation and the number and frequency of COMPRESS runs.

LARGEGAP

Allocates disk space to minidisks in a best fit fashion from the largest gap of available space on any volume that has an extent in a subpool. CA VM:Director searches volumes in reverse order of their definition in the DASD CONFIG file. It repeats the search for a best fit for each subsequent minidisk allocation.

The best fit is always an exact match in size between the requested disk space and an available gap. If CA VM:Director cannot find an exact match, it looks for the next best fit which is the largest gap that has sufficient available space. If two or more locations in a single extent satisfy the best fit criteria, then the HIGHEND | LOWEND options determine which gap CA VM:Director uses.

Note: SMALLGAP and LARGEGAP can sometimes help reduce the amount of fragmentation and the number and frequency of COMPRESS runs.

HIGHEND

Specifies that when CA VM:Director allocates disk space from any extent, the disk space should be as near the high end of the extent as possible.

Important! COMPRESS moves minidisks from the high end of an extent to the low end of the same extent.

LOWEND

Specifies that when CA VM:Director allocates disk space from any extent, the disk space should be as near the low end of the extent as possible.

***maxspace* | ***

Specifies the maximum kilobytes (K) or megabytes (M) of disk space that CA VM:Director can allocate in a single request. If you specify an asterisk as the maximum amount of space, no limit is imposed.

This parameter lets you specify the maximum size of any minidisk in a subpool. If two subpools share the same extent but have different *maxspace* values, the minidisks allocated using one subpool can be larger than the minidisks allocated using the other subpool.

This parameter takes precedence over the *limit* parameter on the MANAGER record in the VMDIRECT MANAGERS file. If a directory manager has an allocation limit of 200M and the subpool maximum is 2K, the largest minidisk that CA VM:Director can define in that subpool is 2K.

[EXPLICIT]

Prevents CA VM:Director from searching this subpool during automatic allocation when using predefined subpool categories. All subpool categories respect the EXPLICIT attribute and ignore that subpool when searching for space.

Note: For more information about using subpool categories, see the Subpool Categories (see page 38).

In a multiple CPU environment, a problem can occur if two systems try to allocate from the same disk space. If you use shared DASD, use the EXPLICIT parameter to prevent automatic allocation from a shared disk. This parameter gives you complete control over any allocations that CA VM:Director makes from those shared disks. You must specify the name of a subpool that is defined with the EXPLICIT attribute to have CA VM:Director allocate space from it.

Description

Each subpool definition specifies a method for allocating from extents and the maximum amount of disk space allowed for each allocation request.

CA VM:Director maintains a complete mapping of all controlled disk space within its virtual memory. Each time you define a minidisk, the subpool you specify governs where and how CA VM:Director will allocate the disk space.

In the DASD CONFIG file, a SUBPOOL record must appear before any EXTENT record that references the named subpool. You cannot continue SUBPOOL records beyond the first line.

To allow directory managers to use any individual subpools, authorize them specifically by including the subpool name on the MANAGER record in the VMDIRECT MANAGERS file.

Examples

- The following record defines a subpool, MAIN, that balances minidisk allocations across all volumes. The individual extents on each volume fill up from the low end, and there is no size limit:

  ```
  SUBPOOL MAIN ROTATING LOWEND *
  ```

- The following record defines a subpool, SALES, that allocates minidisks to the best fitting gap available. The best fitting gap is either an exactly matching gap or the smallest gap with at least 1200K of disk space in addition to the size of the requested minidisk. If two or more possible gaps exist in the same extent, CA VM:Director uses the one nearer the high end of the extent. CA VM:Director cannot allocate a minidisk from this subpool that is larger than 2400K:

  ```
  SUBPOOL SALES SMALLGAP 1200K HIGHEND 2400K
  ```

SYSOPER Record

Use the SYSOPER record to identify the system operators—the user IDs that are to receive messages generated during initialization, by commands, during termination, and when an abend occurs. There can be multiple SYSOPER records.

```
SYSOPER userid ...
```

Configuration File

SYSOPER is defined in the PRODUCT CONFIG file.

Definitions

userid ...

Specifies a user ID or list of user IDs defined as the system operators. If this statement is not present, the OPERATOR user ID is assigned as a system operator.

Description

Messages intended for the system operator are sent to this list of users. If a specified user ID is not logged on when a message is generated, the message is sent to the CA VM:Director system log.

Note: The user IDs specified on the SYSOPER record do not automatically receive authorization to issue the END command to terminate CA VM:Director. User IDs that are allowed to terminate CA VM:Director operation must have a GRANT record with explicit END, END FORCE, or END NOFORCE authorization in the AUTHORIZ CONFIG file.

USEREXIT Record

Use the USEREXIT record to specify the filename of a user exit routine.

```
USEREXIT exitname filename [EXEC | TEXT]
```

Configuration File

USEREXIT is defined in the PRODUCT CONFIG file.

Definitions

exitname

Specifies the name of a user exit that is to receive control at a certain point in the CA VM:Director operation. Valid user exit names appear in the table CA VM:Director User Exit Summary.

filename

Specifies the filename of a site-supplied user exit routine.

EXEC | TEXT

Specifies that the user exit routine is either an EXEC (EXEC, EXEC2, or REXX) or is in TEXT format.

Description

The exit identified by a USEREXIT record receives control at various points in the CA VM:Director operation. You can include several USEREXIT records in the PRODUCT CONFIG file to specify more than one user exit.

User exits are not required for CA VM:Director operation; however, we recommend that you put at least the NEWUSER user exit in place. For more information, see NEW USER Exit (see page 429).

(CA VM:Secure only) Prototype user exit files are supplied. These user exits, except for the AUDIT, POSTMOVE and POSTRULE user exits, are written in REXX and assembler. AUDIT and POSTRULE must be written in Assembler; POSTMOVE must be written in REXX, EXEC2, or EXEC. For more information about user exits, see User Exit Reference (see page 405)."

(CA VM:Director only) Prototype user exit files are supplied. These user exits, except for the AUDIT and POSTMOVE user exits, are written in REXX and Assembler. AUDIT is written in Assembler; POSTMOVE must be written in REXX, EXEC2, or EXEC. For more information about user exits, see User Exit Reference (see page 405).

Note: Do not use the user exits directly from the distribution media without modifying them to meet your site requirements. The prototype user exit files contain documentation that explains how to do this.

VOLUME Record

Use the VOLUME record to define a real DASD volume that CA VM:Director can control.

```
VOLUME volser devtype
```

Configuration File

VOLUME is defined in the DASD CONFIG file.

Definitions

volser

Specifies the DASD volume serial number.

devtype

Specifies the DASD device type. All device types supported by any z/VM system for virtual machine minidisks are valid and any customized names defined on DEVTYPE records.

Description

Each DASD volume identified to CA VM:Director can contain extents from which minidisks are allocated. For more information, see EXTENT Record (see page 370).

Minidisks can exist on volumes not listed in the DASD CONFIG file; however, CA VM:Director cannot detect overlapping minidisk definitions on uncontrolled volumes. Also, directory managers cannot use CA VM:Director to allocate new minidisks on these volumes.

Volumes dedicated to specific guest virtual machines need not be identified to CA VM:Director.

You cannot continue VOLUME records beyond one line.

Example

This VOLUME record identifies a 3380 DASD device with a volser of VMPK02:

```
VOLUME VMPK02 3390
```

Note: For more information, see VOLUME Records in the *Administration Guide*.

WITHHOLD Record

Use the WITHHOLD record to restrict users from using CA VM:Director commands, utilities, or screen selections. WITHHOLD records are also used to define exceptions to general authorizations given by GRANT records.

For CA VM:Director, the WITHHOLD record has this syntax:

WITHHOLD *authority* [OVER *target-userid(s)*] FROM *users*

Configuration File

ACCESS is defined in the PRODUCT CONFIG file.

Definitions

authority

Specifies the authorization being withheld from an entire command, a command and some of its parameters, or a list of commands; *authority* can be any of the authorizations in the following tables:

- Any command or utility authorization, User Selection Menu (USE command) authorization, or Manager Selection Menu (MANAGE command) authorization from the Authorizations to Use CA VM:Director Commands and Utilities table in the *Administration Guide*
- Any predefined variable list from the Predefined Variable Lists table in the *Administration Guide*
- Any command processing authorization from the Authorizations for Command Processing table in the *Administration Guide*
- Any lists you create using LIST records

Note: For information about how to use lists, see User ID Lists and Authority Lists in the chapter "Authorizations" in the *Administration Guide*. For more information about LIST record, see LIST Record (see page 379).

You can also use an authorization with a trailing asterisk (for example, MAN*) to indicate all authorizations that begin with the specified characters. Such authorizations will be flagged by the configuration file editor as unrecognized, but they are still functional.

OVER

Limits the scope of the *authority*; you can use the word OVER, to make it clear that the authority is invalid for user ID(s) that follow the word OVER.

target-userid(s)

Specifies the user ID(s), over which the authority cannot be exercised by users.

users

Consists of a list of user IDs, separated by blanks, that are restricted from receiving authorization. A trailing asterisk (for example, M*) indicates that all user IDs beginning with the specified characters are denied authorization.

Examples

- Prevent any user ID from being assigned to manager KEVIN:

  ```
  WITHHOLD ASSIGN * KEVIN FROM *ALL
  ```

- Prevent any use of the COMPRESS command on any extent of the volume DOSRES:

  ```
  WITHHOLD COMPRESS DOSRES FROM *ALL
  ```

- Prevent user BISHOP from issuing the PASSWORD command:

  ```
  WITHHOLD PASSWORD FROM BISHOP
  ```

- Prevent your administrators, BRENDA, SALLY, and ROB from issuing the MDSKSCAN command:

  ```
  LIST *ADMINS BRENDA SALLY ROB
  WITHHOLD MDSKSCAN FROM *ADMINS
  ```

- Prevent DAVID from issuing the PASSWORD command for any users that are managed by user MAINT.

  ```
  WITHHOLD PASSWORD OVER *DIRUSRS OF MAINT FROM DAVID
  ```

Chapter 4: User Exit Reference

CA VM:Director provides user exits that allow you to extend its flexibility for your site. This chapter explains how to implement the user exits and describes each user exit in detail.

Introducing User Exits

The table in this section summarizes the action each user exit performs. Unless otherwise noted, prototypes listed are supplied in both REXX (filetype XEXEC) and Assembler (filetype XASSEMBL). All prototypes reside on the minidisk or SFS directory identified as the LOCALSAM samples disk in the VMSES/E installation materials search order.

Note: CA VM:Director requires the NEWUSER user exit in order to generate the CA VM:Director directory database using the VMXGNR utility. The NEWUSER user exit is optional but recommended for normal CA VM:Director operation. The other user exits are optional.

User Exit	Action	Prototype
ACCOUNT	Accepts or rejects new account numbers	VMXEXIT4
AUDIT	Accepts or rejects audit records that are being written to the audit file on the AUDT minidisk	VMXEXITG XASSEMBL
CHKPASS	Allows your site to perform its own logon password verification when a logon password is required to execute CA VM:Director commands	VMXEXIT9
COMMAND	Examines commands before they are interpreted by CA VM:Director and optionally modifies or rejects them	VMXEXIT3
DISKCOPY	Allows minidisk copying and formatting with the application of your choice	VMXEXITL
DISTCODE	Accepts or rejects new distribution codes	VMXEXIT6
FORMAT	Allows minidisk formatting with the application of your choice	VMXEXITK
LINK	Accepts or rejects directory links	VMXEXIT8

User Exit	Action	Prototype
MDISK	Accepts or rejects changes to minidisk link modes and passwords The MINIDISK user exit supersedes the MDISK user exit; the MDISK user exit will still run with older releases for compatibility.	VMXEXIT7
MINIDISK	Accepts or rejects changes to minidisk link modes and passwords The virtual address of the minidisk is provided to the exit. The MINIDISK user exit supersedes the MDISK user exit.	VMXEXITM
NEWUSER	Determines the directory manager for each user ID in the directory; required by the database generation utility, VMXGNR	VMXEXIT1
PASSCHNG	Provides notification of a change to a user ID's logon password	VMXEXITJ
PASSWORD	Accepts or rejects new logon passwords	VMXEXIT2
POSTMOVE	Manipulates data that has been moved from minidisks to SFS	VMXEXITH XEXEC
PREMOVE	Provides a destination directory name for data moved from minidisk to SFS	VMXEXITI
SCRATCH	Accepts or rejects requests to scratch minidisks or remove user IDs and can prevent moving a minidisk that should not be moved, such as the CMS S-disk	VMXEXIT5
SECURITY REPORTS	Controls which data appears in audit reports	VMXEXITD

Implementing User Exits

All CA VM:Director user exits, except the NEWUSER user exit, must be modified at your site by an Assembler or REXX programmer. Prototype files, as listed in the table above, are available to make these modifications easier. Your programmers can customize these prototype user exits to meet your site's needs, and implement them using the steps below.

Note: CA VM:Director must receive control from a user exit with the environment in the same state it was when control went to the user exit. If you have a user exit directly manipulate interrupt tasks in either the PSW or control register zero, you must have it return control to CA VM:Director with the same tasks restored.

1. Modify the prototype user exit file. Each prototype user exit file contains comment records that explain how the user exit works and suggest ways to modify the user exit. CA VM:Director passes the standard CMS PLIST (parameter list) to REXX user exits and both the standard CMS PLIST and an extended PLIST to Assembler user exits.

   The standard CMS PLIST varies, depending on whether you are writing the user exit in Assembler, EXEC, or REXX.

   - With the standard CMS Assembler PLIST, the tokens begin with the name of the routine called. For CA VM:Director user exits, this is the name of the user exit, for example, ACCOUNT, NEWUSER, or SCRATCH.

   - If you are coding user exits in REXX or EXEC, the first argument is the actual value passed to the exit and not the name of the user exit. In most of the user exits, the first argument is the user ID that is attempting the action.

   If you are using IPv4 addresses, you may want to use the extended PLIST for the RULEUPDT and COMMAND user exits because CA VM:Director can pass these user exits 15-byte IP addresses.

   The extended PLIST for exits called with password parameters are also required, if password phrase support is configured. These are: PASSWORD, PASSCHNG, CHKPASS, USERPASS, and TERMPASS exits.

2. If you have coded the user exit in REXX or EXEC2, change the filetype of the user exit to EXEC. If you have used Assembler, assemble the user exit to obtain the TEXT file.

3. Add a USEREXIT record to the PRODUCT CONFIG file for that user exit. For example, to use the NEWUSER user exit when the file name of your user exit is NEW2, enter this record:

   ```
   USEREXIT NEWUSER NEW2 EXEC
   ```

   Note: For more information about using the USEREXIT record, see USEREXIT Record (see page 399).

4. The configuration file you loaded from the distribution tape contains a USEREXIT record for each prototype user exit, preceded by an asterisk. To activate these records, remove the asterisks.

5. Put the user exits on the CA VM:Director 191 minidisk after you have customized them. They will then be found by CA VM:Director initialization processing.

ACCOUNT User Exit

Use the ACCOUNT user exit to verify account number changes.

`ACCOUNT userid userid2 newacct [oldacct]`

Definitions

userid

Specifies the user ID requesting the account number change.

Userid2

Specifies the user ID whose account number is being changed.

newacct

Specifies the new account number.

[*oldacct*]

Specifies the old account number.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	New account number is accepted
4	New account number is accepted and a message is stacked; the length limit of the message is 67 bytes
8	New account number is rejected and a message is stacked; the length limit of the message is 67 bytes

Macros

The following macros call the ACCOUNT user exit:

Called by	Optimized Macro Name	Under These Circumstances
VMXADDNT	ADDENTRY	A user ID is created using the ADDENTRY command

Called by	Optimized Macro Name	Under These Circumstances
VMXASN00	ASSIGN	A user ID is assigned to a new directory manager and the account number is changed
VMXMAI00	MAINT	An account number is changed through the MAINT command
VMXMAI20	MAI00020	A user ID is created using the MAINT MANAGE command
VMXMAN20	MAN00020	A user ID is created using the MANAGE command
VMXMUL00	MULTIPLE	A user ID is created using the MULTIPLE command
VMXREPNT	REPENTRY	A user ID is changed using the REPENTRY command
VMXUSE90	USE00090	An account number is changed using the USER command

AUDIT User Exit

Use the AUDIT user exit to accept or reject audit records being written to the VMDIRECT AUDIT file on the AUDT minidisk or to perform additional processing as necessary.

All CA VM:Director commands generate audit records. The AUDIT user exit is called for each record.

(CA VM:Secure only) An audit record is generated for each of the CP commands and Diagnose codes trapped by the Rules Facility. The AUDIT user exit is called for each of these records.

Because of the widespread use of this exit, you should make speed a high-priority goal when you code this exit; slow execution can have a major impact on CA VM:Director performance. To help with performance, the AUDIT exit must be in the form of a TEXT file. An EXEC file is not permitted.

Calling Registers

Registers are as follows when the AUDIT user exit receives control:

Register	Contents
R1	Address of the actual audit record
R13	Address of 18 fullword register save area
R14	Address to which your exit must return
R15	Address of your exit routine
Others	Unpredictable

The data addressed by R1 is the actual audit record, and if the AUDIT user exit alters the record and returns 0 in register R15, the altered record will be written. The report programs VMXSRA (CA VM:Secure only) and VMXSRB will not handle a non-numeric audit code, and they make no provisions for non-standard audit records.

If the AUDIT user exit fails to restore registers properly, or alters storage owned by CA VM:Director (other than the actual audit record), CA VM:Director may abnormally terminate.

Return Codes

The AUDIT exit program is responsible for saving and restoring registers 0 through 14. Before returning to the R14 address, the exit must set register 15 to one of the following values:

Return Code (R15)	Meaning
0	CA VM:Director will write the audit record
8	CA VM:Director will not write the audit record

Audit Record Format

The audit file record format is defined in the VMXADT member of the VMXUSER MACLIB macro library. The defined assembler DSECT present in this COPY file describes the fields and length of the audit file records.

The VMXUSER MACLIB file is installed on the minidisk or SFS directory identified as LOCALSAM in the VMSES/E installation materials. The DSECT contained in this COPY file needs to be included in the AUDIT user exit or referenced by a COPY statement. The VMXUSER MACLIB should be added to the GLOBAL MACLIB list before the user exit is assembled.

When the AUDIT file record format is fixed-length records, each record will be 80 bytes long. When the record format is variable, each record may be up to 256 bytes long.

CHKPASS User Exit

Use the CHKPASS user exit to verify logon passwords entered for executing CA VM:Director commands, certain CP or CA VM product commands.

CHKPASS *userid pswd*

Definitions

userid

The user ID issuing the CA VM:Director command.

pswd

The password or password phrase entered in response to the CA VM:Director prompt for the logon password.

Description

The CHKPASS user exit verifies logon passwords entered for executing CA VM:Director commands. Without a CHKPASS user exit, CA VM:Director verifies passwords by checking the CP directory. The exit does not receive control if the user issuing the command has NOPASS authorization.

A *pswd* argument which contains embedded blanks is passed within the argument string enclosed in single quotes. The user exit sample program contains code to parse the argument string into variables. You must use this method if you use password phrases.

A CHKPASS user exit is essential for sites that use security packages where the logon password is not stored in the CP directory.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	Password entered is valid and processing continues for the command entered
<>0	Any non-zero return code indicates the password is invalid and processing terminates for the command

COMMAND User Exit

Use the COMMAND user exit to examine CA VM:Director commands before they are executed. For full-screen commands such as USER and MANAGE, the initial command line is sent to the COMMAND user exit; subsequent selections in the screen are not processed by the user exit.

Note: CA VM:Director may pass a 15-byte IP address to this user exit; consider using an extended PLIST with this exit. A standard PLIST is limited to 8-byte tokens.

COMMAND *userid* [*command* [*parameters*]]

Definitions

userid

Specifies the user ID executing the specified command.

[*command* [*parameters*]

Specifies the name of the command, which may be abbreviated, and any parameters supplied by the user invoking the command.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	Command has been accepted
4	Alternate command has been stacked
8	Command is invalid and a message is stacked; the length limit of the message is 67 bytes The message stacked by the user exit goes to the CA VM:Director console, and the user receives an INVALID COMMAND message If you want the user to receive any additional messages, use the CP MSG or CP MSGNOH command within the exit

DISKCOPY User Exit

The DISKCOPY user exit allows CA VM:Director to use an alternative application when copying the contents of one minidisk to another.

DISKCOPY *caller calltype diruser userid cuu files linkcuu accmode linkcuu2 accmode2 devtype size blksize Cmdtarget*

By default, the system uses the CMS FORMAT and COPYFILE utility programs to copy minidisk contents when disk are moved or the size of the disk changes. If the new disk has the same size and device type, the faster DDR utility is used.

Definitions

caller

The caller of the command.

calltype

The type of call to be made. Indicates one of the following parameters:

- 'COPY' for copying disks formatted with a CMS file system. The target disk is formatted, and all CMS files from the source disk are copied to it.
- 'CLONE' for exact duplication of a disk. The device type and size of the source and target disks must be identical.

diruser

The command issuer ID or the servant userid.

userid

The user ID that owns the source disk.

cuu

The device address of the source disk.

files

The number of files on the source disk.

linkcuu

The link address of the source disk.

accmode

The access mode of the source disk. An '*' indicates a CLONE operation.

Linkcuu2

The link address of the target disk.

Accmode2

The access mode of the target disk. An '*' indicates a CLONE operation.

devtype

The device type of the source disk.

size

The size of the source disk in blocks or cylinders

blksize

The block size of the target disk. An '=' indicates a CLONE operation.

label

The label for the target disk. An '=' indicates a CLONE operation.

CmdTarget

Indicates one of the following:

- 'USER 'if you are not using the servant facility
- 'SERVANT 'if you are using the servant facility

Description

Because of the requirements for issuing macro primitives, the DISKCOPY user exit cannot be written in Assembler. It must be coded in REXX or EXEC2.

The DISKCOPY user exit uses CA VM:Director macro primitives to communicate with the user or servant user ID. The user or servant user ID is the user ID where the minidisk or the SFS directory is accessed. The CA VM:Director macro primitives most likely to be used are:

- USER EXECUTE or SERVANT EXECUTE—Issues a command on the user or servant virtual machine. The command text arguments that follow are issued on the indicated machine using CMSCALL.
- USER STACK or SERVANT STACK—Places data that you will need later on, such as for a subsequent command, on the user or servant virtual machine program stack. The arguments that follow are stacked in LIFO order on the appropriate program stack.
- USER READ or SERVANT READ—Causes the data that exists on the user or servant virtual machine stack to be read to storage on the user or servant virtual machine.

Use these primitives to manipulate the data in the user or servant program stack. To use any of these primitives, precede it with the token TEST. This token allows the primitive to pass a nonzero return code back to the exit. The sample (prototype) user exit contains examples of the use of these primitives.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	The DISKCOPY user exit successfully copied the requested files to the minidisk.
4	The DISKCOPY user exit defers file copying to the default CA VM:Director copy utility program (COPYFILE or DDR).
8	An error occurred while formatting or accessing the minidisk. A message has been stacked that will be displayed to the issuing user. CA VM:Director treats this as a failure to format the minidisk.

Macros

The DISKCOPY user exit is called by the CLONEDSK macro under the following circumstances:

Called by	Optimized Macro Name	Under These Circumstances
VMXCMD00	CHGMDISK	When a disk is changed with the CHGMDISK command
VMXDUP00	DUPMDISK	When a disk is duplicated with the DUPMDISK command
VMXMANB0	MAN000B0	When a disk is changed with the MANAGE command

DISTCODE User Exit

Use the DISTCODE user exit to validate new distribution codes.

DISTCODE *userid* *userid2* *olddist* *newdist*

Definitions

userid

The user ID requesting the distribution code change.

userid2

The user ID whose distribution code is being changed.

olddist

The old distribution code.

newdist

The new distribution code.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	New distribution code is valid
4	New distribution code is accepted and a message is stacked; the length limit of the message is 67 bytes
8	New distribution code is invalid and a message is stacked; the length limit of the message is 67 bytes

Macros

The following macros call the DISTCODE user exit:

Called by	Optimized Macro Name	Under These Circumstances
VMXADDNT	ADDENTRY	A user ID is created using the ADDENTRY command
VMXMAI00	MAINT	A distribution code is changed using the MAINT command
VMXMAI20	MAI00020	A user ID is created using the MAINT MANAGE command
VMXMAN20	MAN00020	A user ID is created using the MANAGE command
VMXMUL00	MULTIPLE	A user ID is created using the MULTIPLE command
VMXREPNT	REPENTRY	A user ID is changed using the REPENTRY command
VMXUSE90	USE00090	A distribution code is changed using the USER command

FORMAT User Exit

The FORMAT user exit is called each time a minidisk needs to be formatted, allowing you to either:

- Format the minidisk according to site-determined requirements
- Or, respond back to CA VM:Director to have it use the default formatting utility program.

```
FORMAT
userid userid2 vaddr acigroup link_vaddr fmtmode fmtblksize
fmtlabel fmttype macroname fmtcmd cmdtarget
```

Definitions

userid

The user ID running the command requesting minidisk formatting.

Userid2

The owning user ID of the minidisk being formatted.

vaddr

The virtual address of the minidisk owned by *userid2*.

acigroup

The ACI group of *userid2*. If *userid2* does not have an ACI group or if the ACI group is not available, an asterisk (*) is specified.

link_vaddr

The virtual address of the link to the disk being formatted.

fmtmode

The CMS access mode to be used on the format command.

fmtblksize

The block size to be used on the format command. Expected values are 512, 1024, 2048, 4096, 1K, 2K, or 4K.

fmtlabel

The label to be used on the format command.

fmttype

Indicates if the disk being formatted is a new area (ADD) or is being deleted (DEL).

macroname

The caller of the FORMATMD macro, which calls the FORMAT user exit.

fmtcmd

Indicates the format command that CA VM:Director would use, as specified on the CA VM:Director ALTERNAT record in the PRODUCT CONFIG file.

Note: For more information, see ALTERNAT Record (see page 355).

cmdtarget

Indicates the CA VM:Director primitive to use to issue your commands. Valid entries for this field are USER or SERVANT.

Description

Because of the requirements of issuing macro primitives, the FORMAT user exit cannot be written in Assembler. It must be coded in REXX or EXEC2.

When moving a minidisk from one location to another and the device type, block size, or number of cylinders of the disk has not changed, CA VM:Director uses DDR to move the minidisk to its new location instead of using the FORMAT and COPYFILE commands. This means that only the source minidisk is formatted; therefore, the FORMAT user exit is called only for the source minidisk.

The FORMAT user exit uses CA VM:Director macro primitives, commands that are issued within the user exit, to communicate with the user or servant user ID. The user or servant user ID is the user ID where the minidisk and the SFS directory are accessed.

The CA VM:Director macro primitives that are most likely to be used are:

- USER EXECUTE or SERVANT EXECUTE—Issues a command on the user or servant virtual machine. The command text arguments that follow are issued on the indicated machine using CMSCALL.
- USER STACK or SERVANT STACK—Places data that you will need later on, such as for a subsequent command, on the user or servant virtual machine program stack. The arguments that follow are stacked in LIFO order on the appropriate program stack.
- USER READ or SERVANT READ—Causes the data that exists on the user or servant virtual machine stack to be read to storage on the user or servant virtual machine.

To use any of these primitives, precede it with the token TEST. This token allows the primitive to pass a nonzero return code back to the exit.

The sample (prototype) user exit contains examples of the use of these primitives.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	The FORMAT user exit successfully formatted and accessed the minidisk.
4	The FORMAT user exit defers formatting to the default CA VM:Director format utility program (CMS FORMAT command).
8	An error occurred while formatting or accessing the minidisk. A message has been stacked that will be displayed to the issuing user. CA VM:Director treats this as a failure to format the minidisk.

Macros

The FORMAT user exit is called by the VMXFMT00 macro, whose optimized macro name is FORMATMD, under the following circumstances:

Called by	Optimized Macro Name	Under These Circumstances
VMXCMP00	COMPRESS	While defragmenting disk space using the COMPRESS command
VMXRECLM	RECLAIM	While reclaiming DASD space using the RECLAIM command
VMXMANS0	MAN000S0	■ Table list When defining or formatting a new minidisk using selection 2 from the Manager Selection Menu ■ When using selection 6 from the Manager Selection Menu to move a minidisk ■ When using the FORMAT option with the ADDMDISK, CHGMDISK, DELMDISK, and DELENTRY commands ■ When using the MAINT MANAGE DISKMOVE, MAINT MANAGE NEWDISK, or MAINT MANAGE SCRATCH commands

Called by	Optimized Macro Name	Under These Circumstances
VMXMAI40	MAI00040	When removing a user ID with the MAINT MANAGE *userid* REMOVE command
VMXMAN30	MAN00030	When using selection 4 from the Manager Selection Menu to scratch a minidisk
VMXMAN40	MAN00040	When using selection 5 from the Manager Selection Menu to remove a user ID
VMXMANA0	MAN000A0	When using selection 9 from the Manager Selection Menu to redefine a minidisk in place

LINK User Exit

Use the LINK user exit to accept or reject a directory link being added to a virtual machine.

`LINK` *`userid target owner vaddr`* `[`*`vaddr2`* `[`*`linkmode`*`]]`

Definitions

userid

Specifies the user ID defining the directory link.

target

Specifies the user ID to receive the directory link.

owner

Specifies the user ID that owns the minidisk being linked.

vaddr

Specifies the virtual address of the minidisk being linked as specified in the owning user ID's directory entry.

Vaddr2

Specifies the virtual address being assigned to the minidisk. The default is the virtual address given the minidisk by the owning user ID (*vaddr*).

linkmode

Indicates the link mode. The default link mode is R.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	New link is valid
4	New link is accepted and a message is stacked; the length limit of the message is 67 bytes
8	New link is rejected and a message is stacked; the length limit of the message is 67 bytes

Macros

The following macros call the LINK user exit:

Called by	Optimized Macro Name	Under These Circumstances
VMXADDNT	ADDENTRY	A user ID is created using the ADDENTRY command
VMXMAI00	MAINT	A new directory link is defined using the MAINT command
VMXREPNT	REPENTRY	A user ID is changed using the REPENTRY command
VMXUSE60	USE00060	A new directory link is defined using the USER command

MDISK User Exit

Important! The MDISK user exit is called only in the absence of the MINIDISK user exit. The MINIDISK user exit is the preferred user exit for this interface.

Use the MDISK user exit to accept or reject changes made to a minidisk link mode and passwords.

`MDISK` *`userid owner`* `[`*`linkmode`* `[`*`readpwd`* `[`*`writepwd`* `[`*`multpwd`*`]]]]`

Definitions

userid

Specifies the user ID requesting the link mode or password change.

owner

Specifies the user ID that owns the minidisk being changed.

linkmode

Specifies the mode with which the minidisk will be linked.

readpwd

Specifies the read password for the minidisk.

writepwd

Specifies the write password for the minidisk.

multpwd

Specifies the multiple password for the minidisk.

(CA VM:Secure only) If you have the Password Encryption Facility installed, link passwords are encrypted.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	Minidisk link mode and password changes are accepted
4	Minidisk link mode and password changes are accepted, and a message is stacked; the length limit of the message is 67 bytes

Return Code	Meaning
8	Minidisk link mode and password changes are rejected, and a message is stacked; the length limit of the message is 67 bytes

Macros

The following macros call the MDISK user exit:

Called by	Optimized Macro Name	Under These Circumstances
VMXAMD00	ADDMDISK	A minidisk is added using the ADDMDISK command
VMXCMD00	CHGMDISK	A minidisk is changed using the CHGMDISK command
VMXMAI00	MAINT	Changes are made using the MAINT command
VMXUSE10	USE00010	Changes are made using the USER command

MINIDISK User Exit

Use the MINIDISK user exit to accept or reject changes made to a minidisk link mode and passwords.

```
MINIDISK  userid1  userid2  vaddr [mode [rpass [wpass [mpass]]]]
```

Note: The MINIDISK user exit supersedes the MDISK user exit and is called in all cases, even if the USEREXIT MDISK record exists.

Definitions

userid1

Specifies the user ID requesting the link mode or password change.

userid2

Specifies the user ID being changed.

vaddr

Specifies the virtual address of the minidisk.

mode

Specifies the mode with which the minidisk will be linked.

rpass

Specifies the read link password for the minidisk.

wpass

Specifies the write link password for the minidisk.

mpass

Specifies the multiple write password for the minidisk.

(CA VM:Secure only) If you have the Password Encryption Facility installed, link passwords are encrypted.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	Minidisk link mode and password changes are accepted
4	Minidisk link mode and password changes are accepted, and a message is stacked; the length limit of the message is 67 bytes
8	Minidisk link mode and password changes are rejected, and a message is stacked; the length limit of the message is 67 bytes

Macros

The following macros call the MINIDISK user exit:

Called by	Optimized Macro Name	Under These Circumstances
VMXAMD00	ADDMDISK	A minidisk is added using the ADDMDISK command
VMXCMD00	CHGMDISK	A minidisk is changed using the CHGMDISK command
VMXMAI00	MAINT	Changes are made using the MAINT command
VMXUSE10	USE00010	Changes are made using the USER command

NEWUSER User Exit

Use the NEWUSER user exit to make sure that you or the specified manager is allowed to be the manager of a new user ID. It also validates the account number that you assign to the new user ID.

NEWUSER *userid userid2 account newmgrid command*

Definitions

userid

Specifies the user ID that manages the directory entry of the new user ID (the user ID issuing the command), or the user ID issuing the VMXGNR utility.

userid2

Specifies the user ID of the virtual machine being created.

account

Specifies the account number associated with the new user ID.

newmgrid

Specifies the new directory manager user ID if the exit was called by the ASSIGN or ADDENTRY command, or the current directory manager if the exit was called by the CHANGE command. Otherwise, the value of *newmgrid* is an asterisk.

command

Specifies the name of the command that called the exit. In the case of a macro, it is the optimized macro name.

Description

CA VM:Director requires the NEWUSER user exit for the VMXGNR database generation utility. VMXGNR calls the NEWUSER user exit when processing the input file (usually the USER DIRECT) to assign user ID to a CA VM:Director directory manager.

The following CA VM:Director commands (macros) call the NEWUSER user exit: ADDENTRY, ASSIGN, CHANGE, MAINT MANAGE, MANAGE, and MULTIPLE. These commands call the NEWUSER user exit to validate the user ID being created, but ignore the manager user ID that is stacked by the NEWUSER user exit.

If you specify the MANAGER option on the ADDENTRY command line, the user ID specified after the MANAGER option will be used, assuming it is a valid CA VM:Director manager user ID and the return code from NEWUSER user exit is zero.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	New user ID accepted. For calls from VMXGNR, this also means the Directory manager assigned to *userid2* is stacked.
4	Name specified by *userid2* is invalid
8	Account number is invalid

Macros

The following macros call the NEWUSER user exit:

Called by	Optimized Macro Name	Under These Circumstances
VMXADDNT	ADDENTRY	Creates a user ID using the ADDENTRY command
VMXASN00	ASSIGN	Assigns a user ID to a new directory manager
VMXCHG00	CHANGE	A directory manager or administrator changes a user ID
VMXGNR	VMXGNR	Generates the directory database using the VMXGNR utility
VMXMAI20	MAI00020	Creates a user ID using the MAINT MANAGE command
VMXMAN20	MAN00020	Creates a user ID using the MANAGE command
VMXMUL00	MULTIPLE	Creates a user ID using the MULTIPLE command

PASSCHNG User Exit

Use the PASSCHNG user exit to provide information that can be used to notify other computer systems or operating systems that a logon password has been changed.

```
PASSCHNG userid target newpw status caller
```

Definitions

userid

The user ID that requested the password change.

target

The user ID whose password has been changed.

newpw

The new password, in plain text, for *target*.

If you are configured for password phrases, then the *newpw* will be a quoted string, if it contains embedded blanks.

The sample user exit provided illustrates the techniques needed for parsing quoted string arguments.

status

The new password status value on the *FL= special comment in the target user directory entry.

Value	Password Status
*	Active
E	Expired
N	New

caller

The name of the macro or command that called the PASSCHNG user exit.

Description

The PASSCHNG user exit is called by certain commands and menu selections after a logon password has been changed. It provides a list of parameters that can be used to allow notification of the change.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	Completed

Macros

The following macros call the PASSCHNG user exit:

Called by	Optimized Macro Name	Under These Circumstances
VMXPSW00	PASSWORD	A logon password is changed using the PASSWORD command
VMXUSE80	USE00080	A logon password is changed using the USER command
VMXMAI00	MAINT	A logon password is changed using the MAINT command with the PASSWORD parameter
VMXIPL00	IPLEDPGM	A logon password is changed at logon time by the IPL program (VMXIPL). Used when users are forced to change their password using the EXPIRE command.

PASSWORD User Exit

Use the PASSWORD user exit to accept or reject the new logon password being specified.

The PASSWORD user exit has this syntax:

PASSWORD *userid userid2 newpw oldpw* [*date* [*pwlist*]]

Definitions

userid

Specifies the user ID that is changing the password.

userid2

Specifies the user ID whose password is being changed.

newpw

Specifies the new password.

If you are configured for password phrases, then the *newpw* will be a quoted string, if it contains embedded blanks.

The sample user exit provided illustrates the techniques needed for parsing quoted string arguments.

oldpw

Specifies the old password.

If you are configured for password phrases, then the *oldpw* will be a quoted string, if it contains embedded blanks.

The sample user exit provided illustrates the techniques needed for parsing quoted string arguments.

date

Specifies the date on which the logon password was last changed, in the format *yy/mm/dd*. If the logon password has never been changed, no date is provided. If no date is available because the user's directory entry has no password history statement, an asterisk is passed as the date. When the user exit is called for expired passwords during a LOGON process, an asterisk is passed as the date.

pwlist

Specifies a list of previously used passwords, as a string of tokens separated by blanks. This information is supplied only if the directory entry for the virtual machine has a password history statement.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	Password is accepted
4	Password is accepted, and a message is stacked; the length limit of the message is 67 bytes
8	Password is rejected, and a message is stacked; the length limit of the message is 67 bytes

Macros

The following macros call the PASSWORD user exit:

Called by	Optimized Macro Name	Under These Circumstances
VMXIPL00	IPLEDPGM	A logon password is changed at logon time for a new user ID or a user password has been expired
VMXMAI00	MAINT	A logon password is changed using the MAINT command
VMXUSE80	USE00080	A logon password is changed using the USER command

POSTMOVE User Exit

Use the POSTMOVE user exit to manipulate data that has been moved from minidisks to SFS.

`POSTMOVE` `*entry filepool vaddr mmode dir dmode recomp*`

Definitions

entry

The directory entry that owns the data that has just been moved from a minidisk to SFS.

filepool

The name of the file pool to which the data was moved.

vaddr

The virtual address of the minidisk that was moved.

mmode

The CMS file mode at which the minidisk is currently accessed on the servant user ID.

dir

The fully qualified SFS directory name to which the data was moved.

dmode

The CMS file mode at which the SFS directory is currently accessed on the servant user ID.

recomp

Is RECOMP if the minidisk whose data was just moved has a recomputed area, or NORECOMP if the minidisk has no recomputed areas.

Description

Because of the requirements of issuing macro primitives, the POSTMOVE user exit cannot be written in assembler. It must be coded in REXX or EXEC2.

The POSTMOVE user exit uses CA VM:Director macro primitives, commands that are issued within the user exit, to communicate with the servant user ID; the servant user ID has ACCESSed both the minidisk and the SFS directory.

The CA VM:Director macro primitives that are most likely to be used are:

- SERVANT EXECUTE—Issues a command on the servant user ID

  The command text arguments that follow will be issued on the servant machine through CMSCALL.

- SERVANT READ—Places the data that exists on the servant user program stack into the CA VM:Director server machine program stack

  To manipulate the data in the servant user stack, which you may need to do as the result of a command, use the SERVANT READ * primitive. This places the contents of the servant user stack into the server program stack, in the same order in which it was stacked on the servant user stack.

- SERVANT STACK—Stacks a line of data on the servant user stack

  Place data that you will need later on, such as for a subsequent command, on the servant user program stack by using the SERVANT STACK primitive. The data will be stacked LIFO on the servant user stack.

To use any of these primitives, precede it with the token TEST. This allows a nonzero return code to be passed back to the exit.

Note: For more information about macro primitives and how they work, see the chapter "Macros" in the *Administration Guide*.

The prototype user exit contains examples of the use of these primitives.

Return Codes

The following table describes the return code:

Return Code	Meaning
0	Completed

Macros

The following macro calls the POSTMOVE user exit:

Called by	Optimized Macro Name	Under These Circumstances
VMXSFMV2	MOVE2SFS	After moving the data on each minidisk

PREMOVE User Exit

Use the PREMOVE user exit to provide the name of the destination directory in SFS for data being moved from minidisk to SFS.

`PREMOVE` *`userid userid2 filepool maddr`*

Definitions

userid

Specifies the user ID that is moving the data.

userid2

Specifies the user ID that owns the data being moved.

filepool

Specifies the name of the file pool to which the data is being moved.

maddr

Specifies the address of the minidisk being moved.

Description

The directory name stacked by the PREMOVE user exit when the return code is set to 4 should not include the filepool name or the top-level directory. It should also not include a colon or leading period.

If the directory named by the stacked line does not exist, it will be created. Only a single level of a directory can be created. You can specify multiple levels of a directory only when all but the lowest level already exists.

If you want to move the data to the top-level directory, stack a blank line and exit with a RC=4.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	SFS destination directory name has not been stacked
4	SFS destination directory name has been stacked

Macros

The following macro calls the PREMOVE user exit:

Called by	Optimized Macro Name	Under These Circumstances
VMXSFMV2	MOVE2SFS	Before moving the data on each minidisk

SCRATCH User Exit

Use the SCRATCH user exit to prevent accidental scratching or moving of vital minidisks by accepting or rejecting a scratch or move request. The SCRATCH user exit also screens requests to remove virtual machines. This user exit is optional but strongly recommended.

`SCRATCH` *`userid userid2 vaddr macroname`*

Definitions

userid

Specifies the user ID that performs the scratch or removal.

userid2

Specifies the user ID being removed or having a minidisk scratched.

vaddr

Specifies the virtual address of the minidisk being scratched. If specified as an asterisk, the exit assumes that *userid2* is being removed, assigned a new directory manager, or changed to a new user ID.

macroname

Specifies the name of the optimized CA VM:Director macro calling this user exit.

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	Minidisk or user ID scratch is accepted
4	Minidisk or user ID scratch is accepted, and a message is stacked; the length limit of the message is 67 bytes
8	Minidisk or user ID scratch is rejected, and a message is stacked; the length limit of the message is 67 bytes

Macros

The following macros call the SCRATCH user exit:

Called by	Optimized Macro Name	Under These Circumstances
VMXASN00	ASSIGN	A user ID is assigned to a new directory manager
VMXCHG00	CHANGE	A directory manager attempts to change a user ID
VMXCMD00	CHGMDISK	A minidisk is changed using the CHGMDISK command
VMXCMP00	COMPRESS	A directory manager defragments an extent
VMXDELNT	DELENTRY	A user ID is removed using the DELENTRY command
VMXDMD00	DELMDISK	A minidisk is removed using the DELMDISK command
VMXMAI40	MAI00040	A directory manager removes a user from the directory using the MAINT MANAGE command
VMXMAN30	MAN00030	A directory manager scratches a minidisk using the MANAGE command
VMXMAN40	MAN00040	A directory manager removes a user from the directory using the MANAGE command
VMXMANA0	MAN000A0	A directory manager uses the Redefine a Minidisk in Place function of the MANAGE command
VMXMANB0	MAN000B0	A directory manager moves a minidisk to another location using the MANAGE command
VMXTRN00	TRANSFER	A directory manager transfers ownership of a user ID or a minidisk

SECURITY REPORTS User Exit

Use the SECURITY REPORTS user exit to control data contained in the audit reports.

The SECURITY REPORTS has this syntax:

```
VMXSRB [infileid outfileid [( USEREXIT fn [ft | TEXT]]
```

Definitions

infileid

Specifies the input filename, filetype, and filemode.

outfileid

Specifies the output filename, filetype, and filemode.

USEREXIT *fn* [*ft*]

Specifies the filename and filetype of the user exit file to be used. The filetype can be either TEXT (the default) or EXEC.

If the filetype is TEXT, the user exit routine saves and restores registers and is called with register 1 pointing to an 80-byte buffer containing the audit record, and register 13 pointing to a standard save area.

Description

The SECURITY REPORTS user exit is specified by an option on the VMXSRB command line. If you use this user exit, the user exit file must reside on a minidisk accessed by the user ID issuing VMXSRB.

After the report program has determined which records to include in the report (based on the options you specified), the user exit is invoked to allow you to exclude any of the selected records. Each selected audit record is passed to the SECURITY REPORTS user exit as input.

Note: For more information about the security reports, see VMXSRB Report Program (see page 456).

Return Codes

The following table describes the return codes:

Return Code	Meaning
0	Record should be used
8	Record should be ignored

Chapter 5: Utility Reference

The CA VM:Director utilities are summarized in the following table:

Utility	Description
SSIENAB	In a Single System Image environment, converts the server source directory minidisk from SSI Ready format to SSI Enabled, or converts from SSI Enabled back to SSI Ready
VMXBKP01	Creates a single CP source directory file from the CA VM:Director directory database (DRCT) minidisk
VMXBKP02	Uses DDR to copy the CA VM:Director DRCT minidisk to a mirror–image minidisk
VMXBKP03	Similar to VMXBKP02 except that it uses the CMS COPYFILE command to copy the files. If an alternate utility program is specified on the ALTERNAT COPYFILE configuration record, then the alternate utility program is used to copy the files.
VMXGNR	Generates the CA VM:Director directory database and converts your CP source directory file to database format.
VMXIPL	Writes a program image on the IPLDISK minidisk. When the program is IPLed, it interacts with CA VM:Director to update a user logon password
VMXMACUP	Updates, optimizes, and replaces source macros, EXECs, and MESSAGES files on the CA VM:Director service virtual machine
VMXSRB	Generates a report of all audit data captured by the CA VM:Director Audit Facility

Note: For more information about authorizations required to use the utilities, see the chapter "Authorizations to Use CA VM:Director Commands and Utilities" in the *Administration Guide*.

SSIENAB Utility

Use the SSIENAB utility in a Single System Image environment to convert the server source directory minidisk from SSI Ready format to SSI Enabled, or to convert from SSI Enabled back to SSI Ready.

```
SSIENAB {ON | OFF} filemode
```

Definitions

ON

Converts from SSI Ready format to SSI Enabled.

OFF

Converts from SSI Enabled format to SSI Ready.

filemode

Identifies the minidisk file mode where the source directory minidisk (usually 01B0) is accessed.

Description

When SSIENAB is executed with the ON parameter, it retrieves the SSI member name from the running VM system and converts all BUILD statements in every directory entry, from the form "BUILD ON *" to the form "BUILD ON membername."

When SSIENAB is executed with the OFF parameter, it retrieves the SSI member name from the running VM system and converts all BUILD statements in every directory entry, from the form "BUILD ON membername*" to the form "BUILD ON *." SSIENAB can successfully convert from SSI Enabled to SSI Ready only when every BUILD statement in the directory references the same member name. The procedure to convert an SSI complex to a non-=SSI environment is to first remove all members from the system except one, then convert the single node SSI complex to a non-SSI system.

Note: For more information on using SSIENAB see "Converting non-SSI to SSI" in the *Administration Guide*.

VMXBKP01 Utility

Use the VMXBKP01 EXEC to back up the CA VM:Director source directory minidisk (DRCT).

```
VMXBKP01 [END] [HOLD]
```

Authorization

Your user ID must have a GRANT record with BACKUP and MAY authorization in the AUTHORIZ CONFIG file. You must also have access to the following:

- The deployed administration minidisk, usually the VMDIRECT 176 disk, where the VMXBKP01 EXEC resides
- Public minidisk where the VMDIRECT MODULE resides

Definitions

[END]

Invokes the END command when VMXBKP01 is finished and brings down CA VM:Director.

[HOLD]

Causes VMXBKP01 to include users on the HOLD minidisk in the backup. Users on both the DRCT and HOLD minidisks are put in the USER DIRECT file.

Description

Each directory entry file on the DRCT minidisk is appended to a temporary file containing a DIRECT directory control statement. After all directory entries are appended, the temporary file is renamed to USER DIRECT. VMXBKP01 copies the USER DIRECT file to the A-disk or the first read/write disk accessed and erases the BKUP minidisk, because all recent changes are now backed up in the USER DIRECT file.

The VMXBKP01 utility supports the INCLUDE directory statement.

Note: The old options NOTAPE, PROF and NOPROF are allowed for compatibility with older releases but are ignored. Directory profiles are always included in the USER DIRECT file created. This compatibility support should not be expected to continue in the future. Remove these old options.

Usage Notes

Consider the following points before using VMXBKP01:

- Execute this utility from a user ID with BACKUP and MAY authorization in the AUTHORIZ CONFIG file.
- CA VM:Director must be up and running to use this utility, but you cannot execute VMXBKP01 from the VMDIRECT user ID.
- If the CA VM:Director service virtual machine has a name other than VMDIRECT, you must modify the VMXBKP01 EXEC to use the new name.
- While the VMXBKP01 utility is executing, CA VM:Director has a lock on the directory. Updates to the directory cannot be done.

VMXBKP01 will process a DIRECT STATEMNT file if it exists. This file was originally created by the VMXGNR utility when the CA VM:Director database was created. If you change the location of your object directory, remember to update the DIRECT STATEMNT file.

In order to be processed by VMXBKP01, the DIRECT STATEMNT file must exist on the 'A' disk or 'A' disk extension of the user ID that runs VMXBKP01. We recommend that you follow these steps:

1. Link to the product server's 191 disk.
2. Access the link to the product server's 191 disk as an extension of the 'A' disk on the user ID that runs VMXBKP01.

If the DIRECT STATEMNT file is not available, CA VM:Director will dynamically determine the location of the object directory from the DIRECT record in the PRODUCT CONFIG file.

Use the VMXBKP01 utility at least once a week or whenever you change an important minidisk or user ID.

To protect against data loss in the event of hardware failures, save your backed up USER DIRECT file on a different real volume from the DRCT minidisk.

Recovering Your Object Directory

To recover your object directory in the event it has been destroyed and no users are defined to your system:

1. From the OPERATOR user ID, which is logged on at system initialization, use the CP DEFINE MDISK command to define the minidisk containing the latest USER DIRECT file.

2. Bring that directory online with the CMS DIRECTXA command.
3. Bring up the CA VM:Director service virtual machine and let it reinitialize the object directory from the current source files on its DRCT minidisk.

Recovering Your Source Directory from a Disk Backup

To recover your source directory on either type of system in the event your DRCT minidisk has been damaged:

1. End the CA VM:Director server if it is running.
2. Log on to the CA VM:Director service virtual machine.
3. Edit the USER DIRECT file and define a new DRCT minidisk for CA VM:Director if necessary. Bring that minidisk online and format it with the CMS DIRECT command.
4. Link read–only to the minidisk that contains the latest USER DIRECT file.
5. Run the VMXGNR utility against this USER DIRECT file.
6. Update the DRCT minidisk with the files on the BKUP minidisk with the COPYFILE command and the REPLACE option. All changes that accumulated between the last time VMXBKP01 was run and the present will be on the BKUP minidisk.

VMXBKP02 Utility

The VMXBKP02 EXEC uses the IBM DDR utility to create a mirror image of your source directory minidisk (DRCT) to one of your minidisks.

```
VMXBKP02 [vaddr]
```

Authorizations

The user ID that runs this utility needs BACKUP authorization in the AUTHORIZ CONFIG file, and must have access to both the VMXBKP02 EXEC on the deployed administration minidisk (usually the VMDIRECT 176) minidisk, and the public minidisk where the VMDIRECT MODULE resides.

Definitions

vaddr

Specifies the virtual address of the backup minidisk. The backup minidisk cannot be your A-disk, and you must have a write link to it. The default virtual address is 200.

Description

VMXBKP02 links to the CA VM:Director DRCT minidisk to perform the backup and detaches from this minidisk upon completion. A link password is not required because a temporary directory link is added to the issuing virtual machine's directory entry. After the backup is completed, CA VM:Director erases the directory entries on the CA VM:Director BKUP minidisk because all recent changes are now backed up.

To protect against data loss in the event of hardware failures, allocate your backup minidisk on a different real DASD volume from your DRCT minidisk.

CA VM:Director must be up and running to use this utility.

If the CA VM:Director service virtual machine has a user ID other than VMDIRECT, the VMXBKP02 EXEC must be modified to use the new user ID.

Note: VMXBKP02 erases all old files from the *vaddr* minidisk before it copies the DRCT minidisk to the backup minidisk.

Examples

- Create a mirror image of your source directory (DRCT) minidisk to a minidisk that you own:

    1. Choose a minidisk to which you want VMXBKP02 to write your mirror image.
    2. Enter the following command:

        `VMXBKP02` *`vaddr`*

- Later, to recover your source directory:

    1. Create a new DRCT minidisk.
    2. Copy the source directory database files from your backup minidisk to the new DRCT minidisk.
    3. Update the DRCT minidisk using the files that have accumulated on the CA VM:Director BKUP minidisk since VMXBKP02 last ran

VMXBKP03 Utility

Use the VMXBKP03 EXEC to copy all the files from your source directory minidisk (DRCT) to a minidisk you own.

VMXBKP03 [*vaddr* [*filemode*]]

Authorizations

The user ID that runs this utility needs BACKUP authorization in the AUTHORIZ CONFIG file, and must have access to both the VMXBKP03 EXEC on the deployed administration minidisk (usually the VMDIRECT 176) and the public minidisk where the VMDIRECT MODULE resides.

Definitions

vaddr

Specifies the backup minidisk's virtual address. You must have a write link to it. If this parameter is omitted, then the virtual address used is 192.

filemode

Specifies the filemode of your backup minidisk. If this parameter is omitted, then the first available Read/Write filemode is used.

Description

The VMXBKP03 EXEC uses the copy utility specified on the ALTERNAT COPYFILE configuration record to copy all the files from your source directory minidisk, DRCT, to a minidisk that you, or the user ID issuing this EXEC, own. If no ALTERNAT COPYFILE record exists, then CA VM:Director uses the CMS COPYFILE command.

VMXBKP03 links to the CA VM:Director DRCT minidisk to perform the backup and detaches upon completion. A link password is not required because a temporary directory link is added to the issuing virtual machine's directory entry. After the backup is completed, CA VM:Director erases the directory entries on the BKUP minidisk because all recent changes are now backed up.

CA VM:Director must be up and running to use this utility.

Note: The VMXBKP03 utility erases all files on the specified backup minidisk before it copies the DRCT minidisk to it.

To protect against data loss in the event of hardware failures, allocate your backup minidisk on a different real DASD volume from your DRCT minidisk.

If the CA VM:Director service virtual machine has a user ID other than VMDIRECT, the VMXBKP03 EXEC must be modified to use the new user ID.

Examples

- To copy files from your source directory to your 193 minidisk, enter the command:

  `VMXBKP03 193 A`

- To recover this source directory:

  1. Create a DRCT minidisk.
  2. Copy the files from your backup minidisk to the new DRCT minidisk.
  3. Update the DRCT minidisk using the files that have accumulated on the CA VM:Director BKUP minidisk since VMXBKP03 last ran.

VMXGNR Utility

Use the VMXGNR utility to generate the CA VM:Director directory database.

The syntax for the VMXGNR utility is:

```
VMXGNR fn ft fm targetfm exitname  [EXEC | TEXT]  [( NOPW]
```

Definitions

fn ft fm

Specifies the filename, filetype, and filemode of the current CP source directory file.

targetfm

Specifies the filemode of the target minidisk, DRCT. This minidisk must be linked read/write and must not contain any files.

exitname

Specifies the filename of the NEWUSER exit routine. (The filename of the prototype NEWUSER exit is VMXEXIT1.)

EXEC | TEXT

Indicates that the NEWUSER exit is an EXEC file or a TEXT file. TEXT is the default.

NOPW

Does not add password history special comments (*PW=) to each directory entry during the database generation process. If *PW= special comments already exist in the USER DIRECT file from previously used directory entries, NOPW leaves them there.

Description

The VMXGNR utility, located on the CA VM:Director 192 disk, generates the CA VM:Director directory database by copying your existing CP source directory to CA VM:Director database format.

Each user CP directory entry is placed in a separate file on the target minidisk, with the user ID as the filename and the directory manager user ID as the filetype. The VMXGNR utility checks to see that the target minidisk is labeled DRCT.

The VMXGNR utility places all CP directory statements that precede the GLOBALDEF, USER, or PROFILE statements (whichever comes first) in the DIRECT STATEMENT file on the CA VM:Director A–disk. This file is used by the CA VM:Director backup utility, VMXBKP01. All statements from the GLOBALDEF statement to the first USER or PROFILE statement are placed in the VMDIRECT GLOBALS file, with the exception of POSIXGROUP statements which are placed in the VMDIRECT POSIX file. (VMXGNR supports the POSIXGROUP, POSIXINFO, POSIXOPT, and POSIXGLIST directory statements.)

The VMXGNR utility adds password history special comments (*PW=) and edit special comments (*ED=) to directory entries when generating the database:

- The *PW= special comment contains the date (in the format yy/mm/dd) the user's logon password was last changed, and a list of most recent logon passwords for this user ID. Initially, the date of the password change is set to the current date. Up to 57 characters are available for storing old passwords. When CA VM:Director initializes, the data on the *PW= special comment is saved for use by the QUERY command.
- The *ED= special comment contains the date, time, and user ID that performed the last update to a directory entry, and the name of the macro invoked to perform the last update.

The VMXGNR utility checks for parameters on ACCOUNT statements. If it finds no parameters, it deletes the ACCOUNT statement from the directory. When an ACCOUNT statement is deleted, the account number and distribution code default to the user ID.

The VMXGNR utility converts lowercase letters to uppercase on directory control statements. Lowercase letters on comment statements remain unchanged. Additionally, upper and lowercase is preserved on POSIXINFO, POSIXGLIST, and POSIXGROUP statements.

The VMXGNR utility deletes the DIRMAINT comment statement used for storing checksums because CA VM:Director does not maintain checksums. If you need to use DIRMAINT, run VMXBKP01 to create a directory source file of the CA VM:Director directory database; then use DIRMAINT to reinitialize the checksum statements.

Valid access modes for the LINK control statement in a directory entry are R, RR, W, WR, M, MR, and MW. If a V has been appended to the right of an access mode on a LINK statement, the VMXGNR utility removes the V to ensure that a valid access mode is specified.

The VMXGNR utility truncates directory records to a logical record length of 71.

The DIRECTORY control statement allows you to specify the CPUID and a system name for each of the processors that might share the CP source directory. The SYSAFFIN statement tailors which entries and which statements within those entries are used on each processor.

The VMXGNR utility uses these DIRECTORY control statements to generate a CA VM:Director directory database containing only those user IDs that should appear on the processor where the VMXGNR utility is running. It also includes only the appropriate statements within an entry.

The VMXGNR utility processes the DIRECTORY statements and looks for a match between the CPUIDs on those statements and the CPUID of the processor that is running. When it finds a match, it saves the associated system affinity name for use in processing any subsequent SYSAFFIN statements. You see an informational message that gives you the CPUID and model number of the running processor, and the system affinity name that the VMXGNR utility will use to generate the rest of the directory database. The CPUID of the host processor, not the one of the virtual machine running the VMXGNR command, is the one used for matching.

The VMXGNR utility deviates slightly from the IBM support of the DIRECTORY statements. These differences are as follows:

- If a SYSAFFIN statement refers to a system affinity name that you did not define on a DIRECTORY statement, the VMXGNR utility treats it as not matching the current system name and acts accordingly.
- Any duplicate CPUID specifications are ignored. This includes any duplication created by use of an asterisk as the first character of the CPUID.

Do not attempt to generate an encrypted database until you have successfully installed and tested CA VM:Director.

(CA VM:Secure only)To avoid the loss of your management scheme when you run the VMXGNR utility, code the NEWUSER exit to reflect your current management scheme. A non–zero return code from the NEWUSER exit indicates that the VMXGNR utility skipped the directory entry being processed. Do not initialize CA VM:Secure if any entries have been skipped. For more information, see the NEWUSER user exit (see page 429).

To avoid the automatic insertion of *PW= special comments in user IDs, use the VMXGNR utility with the NOPW option to generate the database. If you allow the VMXGNR utility to insert *PW= special comments in all entries, and later determine that you have user IDs whose passwords do not need to be expired regularly, for instance, MAINT, edit the user directory entry and remove the *PW= special comment.

Comment statements that occur after the last directory control statement for User A and before the next USER directory statement for User B, are saved in User B's directory entry, immediately after the USER statement. Therefore, make sure all comments pertaining to User A's directory entry precede User A's last directory control statement or that they are interspersed within the directory entry.

Before you generate your CA VM:Director directory database, ensure that the source directory file to be used by the VMXGNR utility, is current and the BKUP minidisk is empty.

Use this utility during CA VM:Director installation or during disaster recovery.

Example

To generate the CA VM:Director directory database from your USER DIRECT file, storing the results on your U–disk, enter:

```
vmxgnr user direct a u vmxexit1 exec
```

VMXIPL Utility

The VMXIPL utility writes an IPLable program on the CA VM:Director IPLDISK.

```
VMXIPL vaddr [prodsysid]
```

Definitions

vaddr

Specifies the virtual address of the minidisk where the IPLable program is to be written.

prodsysid

Specifies the user ID of the CA VM:Director service virtual machine. The default user ID from which you run the utility.

Description

When the CA VM:Director IPLDISK minidisk is initialized at logon, the VMXIPL utility updates a user logon password before initializing CMS.

(CA VM:Secure only) This utility is not necessary if the CA VM:Secure Rules Facility is implemented at your site.

The IPLDISK minidisk is owned by CA VM:Director and is defined with an RR link mode. Before running this utility, relink the minidisk with an MR link mode.

The IPLDISK minidisk cannot contain any CMS files. Therefore, you need not format this minidisk; simply running this program destroys all files residing on the minidisk.

VMXSRB Report Program

Use the VMXSRB report program to generate formatted output of all audit data captured by CA VM:Director.

```
VMXSRB [fn ft fm [ofn oft ofm]] [("Options"]
```

```
Options:

[BYUSER userid]
[NOCP]
[PRINT]
[TOUSER userid]
[UPCASE]
[USEREXIT exitfn [TEXT | EXEC]]
```

Definitions

fn ft fm

Specifies the file that contains the audit data. This data is collected by the AUDITEXT command, so this file should be the same name as the one you specified on the AUDITEXT command. The default is AUDIT EXTRACT A0.

ofn oft ofm

Specifies the file to which to write the audit report. The default is REPORTB LISTING A0.

BYUSER *userid*

Includes in the report only actions issued by the specified user ID.

NOCP

(CA VM:Secure only) Excludes from the report all CP command activity. This excludes all data reported by VMXSRA. This option is meaningful only if the Rules Facility is implemented.

PRINT

Sends the report to the virtual printer instead of your A–disk.

TOUSER *userid*

Includes in the report only data where the specified user ID is the object of the action.

UPCASE

Produces the report in uppercase.

USEREXIT *fn* [*ft*]

The filename and filetype of the SECURITY REPORTS user exit. EXEC2, REXX, and assembler language are supported. The prototype SECURITY REPORTS exit is VMXEXITD. The filetype can be TEXT or EXEC; the default is TEXT.

Description

The explanation of each action being audited is an English translation of the audit codes stored with the audit record. The output is sorted by date and time.

For each record in the report, the following information displays:

- Date and time of the action
- Terminal address from which the action was issued. For more information about terminal addresses, see Terminal Addresses.
- User ID that issued the action
- Brief explanation of the action

The SECURITY REPORTS User Exit (see page 441) allows you to exclude records from the report. If you use the SECURITY REPORTS user exit, the user exit file must reside on a minidisk you have accessed.

Write the SECURITY REPORTS user exit in assembler language for faster execution.

Return Codes

The following table describes the return codes:

Return Code	Meaning
24	Invalid parameter in command (accompanied by message DMSVMX070E)
28	One of the following: ■ Input file not found (accompanied by message DMSVMX002E) ■ Missing parameter (accompanied by message DMSVMX070E)
100	One of the following: ■ Error reading input file (accompanied by message DMSVMX104S) ■ Error writing output file (accompanied by message DMSVMX105E) ■ Error printing file (accompanied by message DMSVMX123S) ■ Unrecognized error code from user exit (accompanied by message VMXSRB952E) ■ Unrecognized audit code (accompanied by message VMXSRB952E)

Note: For more information about generating the audit information used as input for this report program, see AUDITEXT Command in the chapter Command Reference (see page 31).

Index