WICAT Multi-user Control System

WMCS

Addendum to the Reference Manuals

188-190-206 A

January 1987

• Software •
Publications

WICATsystems

Copyright © 1986 by WICAT Systems Incorporated
All Rights Reserved
Printed in the United States of America

Receipt of this manual must not be construed as any kind of commitment, on the part of WICAT Systems Incorporated, regarding delivery or ownership of items manufactured by WICAT Systems.

This manual is subject to change without notice.

First Printing January 1987

Trademarks Used in this Publication

WMCS is a registered trademark of WICAT Systems
WBASIC is a registered trademark of WICAT Systems
RBTE is a trademark of WICAT Systems

Information about this Manual

Review the following items before you read this publication.

The subject of this manual

This manual describes features of the Command Interpreter Program (CIP) and system calls in the WMCS operating system that have been added since the WMCS manuals were printed in May 1985.

The audience for whom this publication was written

This manual is written for users who have read the WMCS User's Reference Manual and the various utilities manuals, for system managers who have read the WMCS System Manager's Reference Manual, and for systems programmers who have read the WMCS Programmer's Reference Manual.

Related publications

WMCS User's Reference Manual
WMCS System Manager's Reference Manual
WMCS Programmer's Reference Manual

Typographical Conventions Used in this Publication

Bold facing indicates what you should type.

Square brackets, [], indicate a function key, the name of which appears in uppercase within the brackets. For example, [RETRN], [CTRL], etc. Braces, {}, indicate a key in the keypad.

Underlining is used for emphasis.

Table of Contents

Chapter 1

User's Reference

The WMCS User's Reference Manual describes the Command Interpreter Program (CIP).

This chapter explains modifications that have been made to WMCS functions, the Command Interpreter Program (CIP), and CIP commands since the last printing of the WMCS User's Reference Manual (May 1985). The chapter also describes five new CIP commands: asap, confer, defrag, dmapper, and make.

New Features and Modifications in the CIP

This section describes modifications and additions to the CIP.

Devicenames

On computers that use SCSI disk or tape drives, such as the System 125Ø, System 1255, and System 126Ø, these are the default devicenames for the drives:

Device Name	Device
_STØ	SCSI cartridge tape drive
_SDØ	SCSI hard disk drive

Significant Command-line Characters

The character sequence //# has been added to the list of significant command-line characters. Also, the function of the double quotation mark has been modified.

//# Double Slash, Number Sign

The double slash, number sign sequence, //#, is used to open a disk device as if it were a file. When a disk is opened this way, you can read from it and write to it as if it were a file (that is, the effect is the same as if you had mounted the disk with the mnt command using the :special switch). The double slash and number sign must immediately follow the devicename, as in the following examples:

```
_sdØ//# _dcØ//#
```

" Double Quotation Mark

Switch names are no longer recognized when they are enclosed in double quotation marks. For example, the CIP no longer recognizes the :since= switch in the following example:

```
":since=today"
```

Modifications to WMCS Functions

These are the modifications that have been made to WMCS functions.

Broadcast

The format for broadcast message has changed. A header line has been added that displays username, source device, and current time. The message format was also modified to autowrap the message at the port's screen width (defined by dstat). If the screen width is below the minimum (2Ø), the width will be set to the minimum. A width of zero will be mapped to 8Ø.

These commands are affected:

```
keygen      qprint      recover      send
shutdown    wait        watchdog
```

File Types

A :filetype= switch has been added to allow the user to control the type of files being generated.

The following file types are supported directly by the WMCS:

Name	Value	Description
DATA	0	normal data file
DIRECTORY	1	directory file
IMAGE	2	image file
KSAMDATA	3	KSAM data file
KSAMKEY	4	KSAM key file
LLIMAGE	5	ll image file
ARCHCONT	6	archive continuation file
ENCRYPT	7	encrypted file
SYSTEM	8	system file
ARCHIVE	9	archive file
CIPCMD	10	cip cmd file
COBOL	11	COBOL file
BASIC	12	BASIC file
PASCAL	13	Pascal file
OBJECT	14	object file

File types from 15-255 are reserved for future use by WMCS. File types from 256-65535 are available for user definition.

Aliases for user file types can be implemented using logical names. Two logical names are required.

For example, to implement file type 300 and call it MYFILETYPE, the logical names to get the correct translations are:

```
> "@filetype300" := myfiletype
> "@myfiletype"  := 300
```

When using the :filetype= switch, unique abbreviations are allowed for all file-type names except user aliases, which must be completely spelled out. Valid characters for user file-type aliases are alphanumerics and the underscore.

The following commands are affected by this change:

```
copy        create      link        tcopy
translit    vew
```

Pause

For all commands that have the :pause switch, the length of the screen is now specified by the port's screen length (defined by dstat).

The return key, [RETRN], now advances the display one line. Other characters advance the display one screen length as before.

Terminal Type and Setup Files

For all utilities that use the terminal-independent screen handling routines, the names of the setup files have been changed. The setup files used to be found in the directory called SYS$DISK/SYSLIB/, and they were called SETUPxxx.SYS, where xxx was a number that indicated the terminal type. Setup files are now found in the directory called SYS$DISK/SYSLIB.SETUP/. The setup files now have names that indicate what type of terminal they represent. The new file extension for setup files is .STP. For example, the T7000 setup file is now called T7000.STP instead of SETUP252.SYS. Following is a list of the old setup files and their new equivalents:

Old Name	New Name
SETUP255.SYS	VISUAL200.STP
SETUP254.SYS	TVI912C.STP
SETUP253.SYS	MG8000.STP
SETUP252.SYS	T7000.STP
SETUP251.SYS	VT52.STP
SETUP250.SYS	VT100.STP
SETUP248.SYS	HYDRA.STP
SETUP247.SYS	WIT.STP
SETUP246.SYS	T7100.STP
SETUP256.SYS	IBMPC.STP

Users' local setup files are now called FTxxx.STP, where xxx is the number of the terminal type. The filename (FTxxx) is the same name that is used by the dstat command. You may assign a name to be used for your setup file and for the :termtype= switch (in dstat) by setting up two logical names. Following is an example of defining terminal type 5 to be a Beehive terminal.

```
> "@ft5" :== "beehive"
> "@beehive" :== "ft5"
```

With these definitions, the setup file in SYS$DISK/SYSLIB.SETUP/ would be called BEEHIVE.STP.

All of the commands that use the terminal-independent screen handling routines have :setupin= and :setupout= switches. These switches define which setup file to use for both input and output.

There are now four ways you can specify a setup file:

1. Specify nothing (this is the default). In this case, the CIP looks in the directory SYS$DISK/SYSLIB.SETUP/ for the setup file that matches the terminal type named by <u>dstat</u>.

2. Specify a filename (for example, T7000). In this case, the CIP looks in the directory SYS$DISK/SYSLIB.SETUP/ for a file with the name you specified (the CIP appends the .STP extension if you did not specify it).

3. Specify a pathname (for example, SYS$DISK/USERS.SETUP/). In this case, the CIP looks in the directory you specified for the setup file that matches the terminal type named by <u>dstat</u>.

4. Specify a pathname and a filename (for example, SYS$DISK / USERS.SETUP/ABC). In this case, the CIP looks for the setup file you specified in the directory you specified (the CIP appends the .STP extension to the filename if you did not specify it).

The CIP first searches for a setup file with the new name style. If the file cannot be found, the CIP searches for a setup file with the old name style. If that file cannot be found, an error is reported.

The following commands are affected by this change:

dm	nsysprof	nuserprof	sysprof
userprof	vew	zap	

<u>Wildcarding</u>

For commands that use wildcarding, three new switches have been added, and one switch has been modified. The switches are these:

:class=	Type a list of device classes separated by commas. Only files that reside on the class(es) of devices given will be included in the list of files returned. The default is all classes.

Valid device classes are:

TTY	TTYSpecial
Pipe	PipeSpecial
Tape	TapeSpecial
Sync	SyncSpecial
Disk	DiskSpecial
Queue	QueueSpecial
Network	NetworkSpecial
NonDev	NonDevSpecial
TTYNetwork	

:filesize= Type a numeric range of file sizes in Kbytes. Only files that fall within the specified size range will be included in the list of files returned. The default is all files (range Ø-).

:typeselect= Type a list of file-type names (including user definable names) and/or ranges of file-type numbers. Files that match the set of specified file types are included in the set of files returned.

Default file-type names are:

Archive	Data	KsamKey
ArchiveCont	Directory	LLImage
Basic	Encrypted	Object
CIPcmd	Image	Pascal
Cobol	KsamData	System

See the File Types section of this addendum for more information on file types and their values.

:sort= Two additional sort options have been added, FILESIZE and FILETYPE (numeric value). The default is to sort based on filename.

The following commands are affected by these changes:

arch	backup	checksum	copy
count	crypt	del	dir
dump	fstat	install	print
pu	ren	restore	scan
tcopy	translit	type	typemrl
usscopy	version	wscan	wsort

Modifications to CIP Commands

The following CIP commands have been modified as described below.

<u>Backup</u>

A :edit= switch has been added. This allows you to change the directory names that are placed inside the <u>arch</u> files. In other words, you can use the :edit switch with <u>backup</u> so you do not have to use it later with <u>restore</u>.

<u>Bkup</u>

The SCSI tape name of ST0 has been added to the list of valid devices on which <u>bkup</u> can make a backup of software programs.

<u>Btup</u>

The units of the :cache= and :usercache= switches for disk class devices have been changed from sectors to Kbytes. Also, the display of these fields has been changed to Kbytes.

The values for the :drivetype= switch have been modified. These are the valid drive types that can be used:

Drive Type	Description
FLOP09A	5.25-inch floppy
FLOP09B	5.25-inch floppy (5 sector)
FLOP015	8-inch floppy
WIN12	12 Mbyte 5.25-inch Winchester
WIN19	19 Mbyte 5.25-inch Winchester
WIN30	36 Mbyte 5.25-inch Winchester
WIN43	43 Mbyte 5.25-inch Winchester
WIN48	48 Mbyte 5.25-inch Winchester
WIN86	86 Mbyte 5.25-inch Winchester (SCSI only)
WIN101	101 Mbyte 5.25-inch Winchester (SCSI only)
WIN141	141 Mbyte 5.25-inch Winchester (SCSI only)
WIN182	182 Mbyte 5.25-inch Winchester (SCSI only)
SMD84B	84 Mbyte SMD disk
SMD168B	169 Mbyte SMD disk
SMD474B	474 Mbyte SMD disk
SMD515B	516 Mbyte SMD disk
SUBDISKA	512 Kbyte subdisk (0.5 Kbytes per sector)
SUBDISKB	512 Kbyte subdisk (1 Kbyte per sector)

The following drive types are no longer valid with the :drivetype= switch:

IMI20	20 Mbyte IMI disk
IMI40	40 Mbyte IMI disk

Checksum

The checksum command has been modified to display the filename and date as soon as the file is opened. After the filename and date have been displayed, the checksum is calculated.

A :edit= switch was added to checksum. You can now control the name of the file created by checksum. This makes checksum files (for later use by verify) much more flexible.

Chkd

The following switches were added to chkd:

:bad=	If specified, this switch marks the range of sectors bad in the bitmap of the device, without checking the sectors.
:badonly	If specified, only those sectors already marked bad on the device are checked. The default checks all sectors.
:bitmap=	If specified, this switch defines the name of a generated bitmap file to use instead of the standard one in /ROOTDIR/. It uses only the bad portion of the alternate bitmap file.
:check	If :nocheck is specified, the disk is not checked, and the current state of the disk is displayed. The default is to check.

:checkallocated	When a disk is checked, only sectors allocated to a file are read, and unallocated sectors have a pattern written to them. Then the disk is verified by reading back the sectors to see if the pattern is the same. If this switch is specified, even sectors that are allocated to files have a pattern written to them. Chkd first reads and saves the original data, then writes its pattern and verifies it, and finally restores the original data. NOTE: Data can be lost if the system goes down in the middle of this operation or if the original data cannot be rewritten. The default is to not write check allocated sectors.
:confirm	If specified, then user confirmation is asked for each time a sector should be changed in the bitmap file (that is, each time a sector that was marked good will be marked bad, or each time a sector that was marked bad will be marked good) . The default is to not confirm.
:good=	If specified, this switch marks the range of sectors good in the bitmap of the device.
:files	If :nofiles is specified, the filenames of sectors marked bad (in files) are not displayed. The default is to display these filenames.
:keepbad=	If specified, this switch defines the name of a bitmap to use instead of the standard file in /ROOTDIR/. Only the bad portion of this alternate bitmap file is written.

Chkd can now check the sectors on a disk that is mounted special. A bitmap form of this output can be saved with the :keepbad= switch.

Config

Config has been updated to allow the specification of the FPOINT and NETWORK class handlers. It has also been updated to handle all of WICAT Systems' current model numbers and the SCSI disk device driver.

Copy

A :delete switch has been added. It causes the destination file to be deleted automatically if *copy* is terminated before an entire file is copied. The default is to delete files not completely copied.

Crypt

A :verify switch has been added. It requests that the key be entered twice and verified if the key is not input on the command line. The default is to verify.

A :delete switch has been added. It causes a file to be deleted automatically if *crypt* is terminated before an entire file is encrypted or decrypted. The default is to delete files not fully copied.

Dev

The *dev* command now displays the new TTYNET class of devices.

Dinit

Dinit now has the ability to format selective tracks on a disk. By default it reads all the data it can from a track before formatting it, and then restores the data it read back to the track after formatting. It will not preserve data if the :nofilesys switch is used.

Also, *dinit* now checks for bad sectors while writing the four system files. It then maps around the bad sectors.

The definition for the :cache= and :usercache= switches for disk class devices has been changed. Instead of being in sectors, the cache size is now in Kbytes. Also, the defaults for these parameters are now based on the size of the disk.

The default sizes of the :ialloc= and :alloc= switches have been changed to be dependent on the size of the disk.

The following switches have been added to dinit:

:keepbad	This switch specifies that we should read the current state of the bad portion of the bitmap file and save it before initializing the disk. This way bad sectors stay marked across uses of dinit. The default is to preserve bad sectors across uses of dinit.
:keepbad=	This specifies an alternate bitmap file (for preserving bad sectors) other than the standard file in /ROOTDIR/. Any valid file designation may be specified.
:verbose	Display explanatory messages while performing the deinitialization. The default is to display verbose messages.
:track=	This switch formats the specified range of tracks only. It preserves what data it can from the track(s) unless the :nofilesys switch is specified.
:sector=	This switch formats the track(s) that the range of sectors is on. It preserves what data it can from the track(s) unless the :nofilesys switch is specified.
:subdkfname=	If this switch is used for a subdisk device, it specifies the filename of the subdisk.
:subdksize=	If this switch is used for a subdisk device, it specifies the maximum size (in sectors) of the subdisk being created.

The values for the :drivetype= switch have been modified. Following is the list of valid drive types that can be used:

Drive Type	Description
FLOP09A	5.25-inch floppy
FLOP09B	5.25-inch floppy (5 sector)
FLOP015	8-inch floppy
WIN12	12 Mbyte 5.25-inch Winchester
WIN19	19 Mbyte 5.25-inch Winchester
WIN30	36 Mbyte 5.25-inch Winchester
WIN43	43 Mbyte 5.25-inch Winchester
WIN48	48 Mbyte 5.25-inch Winchester (WFC or SCSI)
WIN86	86 Mbyte 5.25-inch Winchester (SCSI only)
WIN101	101 Mbyte 5.25-inch Winchester (SCSI only)
WIN141	141 Mbyte 5.25-inch Winchester (SCSI only)
WIN182	182 Mbyte 5.25-inch Winchester (SCSI only)
SMD84B	84 Mbyte SMD disk
SMD168B	169 Mbyte SMD disk
SMD474B	474 Mbyte SMD disk
SMD515B	516 Mbyte SMD disk
SUBDISKA	512 Kbyte subdisk (0.5 Kbytes per sector)
SUBDISKB	512 Kbyte subdisk (1 Kbyte per sector)

The following drive types are no longer valid with the :drivetype= switch:

IMI20	20 Mbyte IMI disk
IMI40	40 Mbyte IMI disk

Dm

A refresh-screen command has been added. It is [CTRL] -(the same as the refresh-screen command in CIP and VEW).

Dstat

The display of dstat has been modified to be more logical. Also, a line has been inserted, defining the class of the fields being displayed.

The following switches have been added:

:network	This switch allows a user to switch a TTY line in and out of network mode.
:preempt	This switch specifies that a given TTY port should run with preemptive interrupt enabled.
:length=	This switch allows the user to define the number of lines on the terminal screen. The default is 24.
:width=	This switch allows the user to define the number of columns on the terminal screen. The default is 80.
:reset	This switch resets the TTY class device.

When a port is set to autobaud mode, the port's baud rate is set to 19200 baud. This is a problem if it is a modem port that you also want to talk out of. To solve this problem, an alternate baud rate has been implemented. A DSTYTTYALTBAUD field has been added to the device-status block for the TTY driver. Whenever a new baud rate is set into the DSTYMODEREG2 field for a device (and that device is using the TTY driver), the baud rate is also inserted into the DSTYTTYALTBAUD field. This works by having the port normally set to 19200 baud for autobaud input. However, if the port is opened by a process (and there are no other users of the port), the _open SVC switches the baud rate on the port to the alternate baud rate specification. Note that dstat sets the baud rate into both of these fields when the :baud= switch is used.

Dumpdiff

A :rewind switch has been added.

Fstat

A :extents switch has been added so that each extent in the file's FCB can be displayed.

Load

The SCSI tape name of STØ has been added to the list of valid devices from which load can load software programs.

Logon

If a user does not log on in five tries, logon now hangs up on SYS$INPUT. If the port is connected to a modem, it is disconnected. This makes it harder for people to break into a system across a modem.

If a user logs on to a remote machine, logon clears out all remote association with the original machine. This makes logging on to a remote machine function exactly like logging on to a local machine.

Logon now changes the UIC to that of the user logging on before it changes to the user's desired directory. This enables users to have their default directory on a machine other than the machine they logged on to.

Makedsr

The following switches have been added:

:strip=	If the :simple switch is also specified, :strip= defines how many bytes of data are to be stripped from the front of the file. The default is 1Ø24 bytes.
:control=	This switch works only if the :simple switch is specified. It has four values: NONE - Copy from the end of the stripped data to the end of the file. BYTE - Immediately after the stripped data, there is a byte-wide count of the number of bytes of data to copy. Starting with the byte containing the count, copy for the specified number of bytes.

WORD - Immediately after the stripped data, there is a word-wide count of how many bytes of data to copy. Starting with the first byte of the word containing the count, copy for the specified number of bytes.

LONG - Immediately after the stripped data, there is a longword-wide count of how many bytes of data to copy. Starting with the first byte of the longword containing the count, copy for the specified number of bytes.

For the byte, word, and long values, after the amount of data is copied, _makedsr_ will check to see if a version string immediately follows. If so, it will be copied also.

Mnt

A :subdkfname= switch has been added. If you are mounting a subdisk device, this switch defines the name of the disk file to use. Any valid file designation may be specified.

The values for the :drivetype= switch have been modified. Following is the list of valid drive types that can be used:

Drive Type	Description
FLOP09A	5.25-inch floppy
FLOP09B	5.25-inch floppy (5 sector)
FLOP015	8-inch floppy
WIN12	12 Mbyte 5.25-inch Winchester
WIN19	19 Mbyte 5.25-inch Winchester
WIN30	36 Mbyte 5.25-inch Winchester
WIN43	43 Mbyte 5.25-inch Winchester
WIN48	48 Mbyte 5.25-inch Winchester
WIN86	86 Mbyte 5.25-inch Winchester (SCSI only)
WIN101	101 Mbyte 5.25-inch Winchester (SCSI only)
WIN141	141 Mbyte 5.25-inch Winchester (SCSI only)
WIN182	182 Mbyte 5.25-inch Winchester (SCSI only)
SMD84B	84 Mbyte SMD disk
SMD168B	169 Mbyte SMD disk
SMD474B	474 Mbyte SMD disk
SMD515B	516 Mbyte SMD disk
SUBDISKA	512 Kbyte subdisk (0.5 Kbytes per sector)
SUBDISKB	512 Kbyte subdisk (1 Kbyte per sector)

The following drive types are no longer valid with the :drivetype= switch:

IMI20	20 Mbyte IMI disk
IMI40	40 Mbyte IMI disk

Nuserprof

A comment field has been added to each record in the display. This allows users to better keep track of the purpose of each record. This also means that the /SYSLIB/NETUAF.DAT file has changed format. Nuserprof automatically converts files from the old format file to the new format.

The :setupin= and :setupout= switches are now used by Nuserprof.

Print

Print has been modified to make the file parameter handle up to 256 characters. Longer file lists are now possible.

Qprint

Qprint has been changed to write data to SYS$OUTPUT.

The :allocate switch has been added for those users that want to run Qprint through a pipe or directly to a disk file. They can specify :noallocate and qprint does not allocate the device.

Shlog

If a logical name crosses more than one line, shlog wraps it automatically.

The :user switch was added to display user process logical names. The default is to display user process logical names.

The :system switch has been changed to cause shlog to display system logical names.

Shutdown

A :stats switch has been added that causes shutdown to display the system's status (using the command pstat :header :systemstatus :port :status :size :priority :scheduled :timeslice) immediately on startup, and after each broadcast.

If a system on a network is shut down, shutdown sends a message to all remote connections that the machine is being shutdown.

The display of shutdown has been modified to display the node name.

Sp

The :kbytes switched has been removed and replaced by the :sectors switch. The default is :nosectors (which means the display is in Kbytes).

Sysprof

Two new fields were added to the screen display. They are "SUBDISK FILENAME" and "SUBDISK SIZE". If a user is defining a subdisk device, the default filename and size of the device can be specified here so they do not need to be defined each time the device is mounted.

Time

If the :prompt switch is specified, time displays a date and time prompt so the user can change it.

Translit

A :delete switch was added; it causes a file to be deleted automatically if translit is terminated before an entire file is copied. The default is to delete files not fully copied.

A :append switch has been added, which allows multiple source files to be put together into one destination file.

Type

A :tail switch has been added, which when specified with the :continuous switch, causes type to start typing at the end of the file.

A :delay= switch has been added, which when specified with the :continuous switch, defines how long to pause (in seconds) between each poll of the file being typed. The default is 1 second.

Usscopy

A :terminate switch has been added. This switch requests that the remote system's usscopy be terminated immediately.

A :receiveonly switch has also been added. Use this switch to specify to the background usscopy command that it may only receive files, and may not send any files to the local usscopy command. Should the local usscopy request the background usscopy to send files, it will return an error and terminate. The default is :noreceiveonly.

Verify

A :allversions switch has been added. It allows users to use checksum, and verify multiple versions of the same file.

Vew

A command called ft has been added that allows the user to set the file type of any file subsequently created by VEW. See the File Types section for more information about file types. The user may specify a file type number or file type name.

Default file type names are:

Archive	Data	KsamKey
ArchiveCont	Directory	LLImage
Basic	Encrypted	Object
CIPcmd	Image	Pascal
Cobol	KsamData	System

The definition of the :memory= switch has been changed. There are now 3 states. They are:

1. If a positive integer is specified, the memory buffer size is the lesser of the given number in Kbytes or the size of the file plus 4 Kbytes.

2. If zero is specified, the memory buffer size is the size of the file plus 4 Kbytes.

3. If a negative integer is specified, the memory buffer size is the absolute value of the given number in Kbytes.

Wait

Wait sets the NOWATCHDOG attribute. Wait is now installed with the SETATTR privilege.

Watchdog

The :polltime= switch has been added to allow the user to specify in minutes how often to wake up and check for processes which are to be killed. The default value is five minutes.

Zap

A gb command has been added that allows a user to go to an explicit byte in the file.

A dd command has been added that allows a user to delete from the current position to the end of the file. This command only works when zap is being used on a disk file (as opposed to a device).

New CIP Commands

This section describes the five new CIP commands:

asap
confer
defrag
dmapper
make

Functional Description

Use this command to list, insert, or delete records from the table of Create Process Indirection Records. The table of Create Process Indirection Records (CPIR) associates an application program with a file type.

Command Line Syntax

Mnemonic	asap			
Required parameters	File Type (Optional if Function is :list) Command Line (Allowed only if Function is :insert)			
Optional parameter	Function (:insert, :delete, :list)			
Switches				
System affected	:siteid=			
Display	:filetype	:header	:log	:pause
Function modifiers	:auto	:confirm		

Parameters

File Type	Function	Required if Function is :insert or :delete; optional if Function is :list. Use this parameter to specify the file type to insert or the file types to list or delete.
	Default	None if Function is :insert or :delete. All file types if Function is :list.
	Syntax	Type a recognized WMCS file type, a file-type alias, a numeral, or a range specification (range only if Function is :list or :delete).

The following file types are supported directly by WMCS:

Name	Value	Description
DATA	0	normal data file
DIRECTORY	1	directory file
IMAGE	2	image file
KSAMDATA	3	KSAM data file
KSAMKEY	4	KSAM key file
LLIMAGE	5	ll image file
ARCHCONT	6	archive continuation file
ENCRYPT	7	encrypted file
SYSTEM	8	system file
ARCHIVE	9	archive file
CIPCMD	10	cip cmd file
COBOL	11	COBOL file
BASIC	12	BASIC file
PASCAL	13	Pascal file
OBJECT	14	object file

File types from 15-255 are reserved for future use by WMCS. File types from 256-65535 are available for user definition.

NOTE: Aliases for user file types can be implemented using logical names. Two logical names are required. For example, to implement file type 300 and call it MYFILETYPE, the logical names to get the correct translations are:

```
"@filetype300" := myfiletype
"@myfiletype"  := 300
```

Valid characters for user file-type aliases are alphanumerics and the underscore. The maximum length of a user file type is 93 characters; however, if the length is greater that 13 characters, the name will cause the :list display of ASAP to shift over. If the name is greater than 11 characters, it will be truncated in other commands that display file types, such as DIR or FSTAT.

Command Line

Function: Required if function is :insert. Otherwise not allowed. This line has two functions:

This string is inserted in front of the user's current command line exactly as it is defined here.

This string is scanned from the front, looking for the first invalid file character (anything that is not alphanumeric, $, or ~) or the end of the string. Wherever it stops, this will be used as the image filename parameter to the create process. The old filename parameter is discarded.

Default: None.

Syntax: Type a string containing the desired command line. No validation is done on the string. If spaces are to be embedded in the string, it must be surrounded by double quotes.

Function	Function	Optional. Use this parameter to specify whether you want to list, insert, or delete records in the system CPIR table.
	Default	:list if Command Line parameter is not specified; :insert if Command Line parameter is specified.
	Syntax	Type one of the following: **:list** to list all the file types that are presently entered in the CPIR table, along with their associated command lines. **:insert** to insert a record into the CPIR system table for the file type specified. **:delete** to delete a record (or records) in the CPIR system table for the file type(s) specified.

Switches

:auto	Function	Use this switch to perform the deletion of records in the system CPIR table without any confirmation.
	Default	:noauto
	Syntax	Type **:auto**
:confirm	Function	Use this switch to confirm or deny each file-type record on deletion.
	Default	:noconfirm
	Syntax	Type **:confirm**
:filetype	Function	Use this switch to include the numeric file type in the :list display.
	Default	:filetype
	Syntax	Type **:nofiletype** to suppress file type in :list display.

:header	Function	Use this switch to display column headers.
	Default	:header
	Syntax	Type **:nohead** to suppress column headings.
:log	Function	Use this switch to specify whether log messages are to be displayed. (Log messages are informational displays that indicate what the utility is doing.)
	Default	The value specified by the <u>option</u> command.
	Syntax	Type **:log** or **:nolog** to override the default.
:pause	Function	Use this switch to stop the display after each screen of information. The user can then press [RETRN] to advance one line, or any other character to advance to the next screen.
	Default	The value specified by the <u>option</u> command.
	Syntax	Type **:pause** or **:nopause** to override the default.
:siteid=	Function	Use this switch to specify the system on which the action is to take place.
	Default	The system on which the calling process is executing.
	Syntax	Type **:siteid=** followed by a numeral or nodename.

Examples

```
> asap basic "sys$disk/sysexe.sgs/wbasic.exe " :insert
```

This command associates the WBASIC interpreter with the file type BASIC. The following message appears on the screen:

```
    Filetype BASIC inserted.
```

> asap :list

This command lists all the records in the system CPIR table. The following type of display appears on the screen:

```
Filetype name  Ftype  Command line
-------------  -----  ------------------------------------------------------
300              300  "SYS$DISK/USERS.MYDIR/MYPROG.EXE special_parml special
                      _parm2 special_parm3 special_parm4 "
BASIC             12  "SYS$DISK/SYSEXE.SGS/WBASIC.EXE "
CIPCMD            10  "SYS$DISK/SYSEXE/CIP.EXE:CF="
```

If an alias is defined for the user file type, then the following display appears on the screen:

```
Filetype name  Ftype  Command line
-------------  -----  ------------------------------------------------------
BASIC             12  "SYS$DISK/SYSEXE.SGS/WBASIC.EXE "
CIPCMD            10  "SYS$DISK/SYSEXE/CIP.EXE:CF="
MYFILETYPE       300  "SYS$DISK/USERS.MYDIR/MYPROG.EXE special_parml special
                      _parm2 special_parm3 special_parm4 "
```

> asap myfiletype :delete

This removes the file type MYFILETYPE from the system CPIR table. The following messages appear on the screen:

```
Filetype MYFILETYPE
Delete (Y or N)? > y
Filetype MYFILETYPE deleted.
```

If you type **asap :list** on the CIP command line, MYFILETYPE will not be displayed, and files of that type will not work as pseudo-image files.

> asap cipcmd "sys$disk/sysexe/cip.exe:CF="

This is the standard asap command that is released in LOCALUP.COM. It allows command files that have a .EXE extension and have a file type of CIPCMD to be executed as if they were programs. Note that the asap command line will be inserted in front of the users command line, but that the image filename will stop just before the colon.

Using Prompts

```
> asap :insert
Filetype       > basic
Command line   > "sys$disk/sysexe.sgs/wbasic.exe "
```

This perfoms the same function as the first example.

Notes on Usage

The asap command enables users to associate application programs with certain file types. In the first example above, the file type BASIC is associated with the WBASIC interpreter. This enables the user to just give the name of a basic program (if it has its file type set correctly) as if it were a command. The WMCS will attempt to execute the file as an image and fail; then it will look in the system CPIR table to see if an alternate command line for the file type of the file given as the "image" is available. If so, then the original command line is appended to the alternate command line, and WMCS again attempts to execute the command line. If you had a BASIC file called MYFILE.BAS, with its file type appropriately set, and if the basic file type was entered into the CPIR tables as mentioned above, then the following two command lines would be functionally equivalent:

```
> wbasic myfile.bas
> myfile.bas
```

This utility enables users to define their own file types, write an application program that works especially with that file type, and just give the name of the appropriate data file and the application will be invoked automatically.

Related CIP Commands

None.

Functional Description

Use this command to initiate or join an interactive conference between users on several different terminals. A conference may cross machine boundaries.

Command Line Syntax

Mnemonic	confer
Required parameter	Addressee
Optional parameter	Message
Switches	
Addressee selection	:exclude= :uic=
Other	:logfile= :scroll

Parameters

Addressee	Function	Required. This parameter is used to specify the username or terminal(s) with which to confer. If the parameter appears to be of the form USERNAME or NODE_USERNAME, the utility will attempt to join a conference with that user. If the attempt fails, it will send a message to all terminals with that user's uic. If the parameter is a device list, the utility will send a message to the devices on the list, requesting them to join a conference with your username.
	Default	None.
	Syntax	Type a username or nodename_username, or type a list of device names separated by commas. Wildcard symbols are allowed.
Message	Function	Optional. Use this parameter to specify additional text for the message sent to each addressee.
	Default	A generic invitation to confer (Please "CONFER nodename_username"), giving your nodename and username.
	Syntax	Type any desired message, enclosed in double quotation marks. To insert special characters in the line, accept symbols must be used. Escape sequences are not allowed in the message. The text of the message will be appended to the default message.

Switches

:exclude=	Function	Use this switch to exclude devices from the device list in the addressee parameter.
	Default	All devices that match the device list in the addressee parameter are selected.
	Syntax	Type **:exclude=** followed by a list of device designations separated by commas. Wildcard characters are allowed.

:logfile=	Function	Use this switch to define a file to be used as a record of the conference. All text of the conference will be copied to the file. This switch will also force **:scroll** mode.
	Default	No record will be kept.
	Syntax	Type **:logfile=** followed by the name of the file to be used as the record.
:scroll	Function	Use this switch to start the conference in scroll, rather than window, mode.
	Default	Start the conference in window mode.
	Syntax	Type **:scroll** to start the conference in scroll mode.
:uic=	Function	Use this switch to select only those devices that are part of the list given in the addressee parameter and that are owned by the specified user or list of users. This also forces the utility to recognize the addressee parameter as a list of devices.
	Default	All devices that match the specified list are selected.
	Syntax	Type **:uic=** followed by a list of UICs or usernames.

Examples

1. This example shows the initiation of a two-person conference by John (on _tt22), who wishes to confer with Fred (who uses terminal _tt23 on the same computer). The network node is called QED.

   JOHN> confer _tt23 "I need your advice on a problem"

   The foregoing command, when executed by John on _tt22, generates the following display on Fred's terminal, _tt23 (if _tt23 has broadcast mode enabled):

```
JOHN __QED_TT22  16-APR-1986 10:03:36.07
Please "CONFER QED_JOHN"  I need your advice on a problem
```

John's terminal clears, and the utility identifies itself on the last line of the screen. If any errors occurred in broadcasting the message, the errors are displayed on John's screen. John waits for Fred to reply.

2. Continuing the foregoing example, Fred brings his terminal to CIP command level and replies to John as follows:

   FRED> confer john

   This command causes the utility to try to connect with the CONFER_JOHN process (which should still be running on John's terminal) on the same network node as Fred. The two processes begin to communicate with each other, and Fred's screen appears as follows (in window mode):

```
-- FRED (QED_TT23) -------------------------------------------

     <space for Fred's messages to John>

-- JOHN (QED_TT22) -------------------------------------------

     <space for John's messages to Fred>

CONFER: John accepts conference on QED_TT22.
```

   At the same time, John's screen appears as follows (in window mode):

```
-- JOHN (QED_TT22) -------------------------------------------

     <space for John's messages to Fred>

-- FRED (QED_TT23) -------------------------------------------

     <space for Fred's messages to John>

CONFER: FRED joining you on QED_TT23.
```

   At this time, anything typed by either Fred or John will appear in the appropriate window on both screens. The lines may be edited by following the usual rules for editing the CIP command line. As each line is completed, the next line in the window is cleared and used. When the last line in a window is completed, the first line will be cleared and used.

   Either person may use the _confer_ internal commands to invite others to join the conference, change modes, or exit.

3. John and Fred in the previous example wish to get Dan's advice. Dan works on his own workstation computer, which has the node name DAN. Fred will invite Dan by using the [ESC][ESC] in command of the utility. The internal command <u>in</u> has exactly the same effect as the command line parameter Addressee. To invite Dan, the prompt would be:

   Invite> dan_dan

   The foregoing command will attempt to connect to the process CONFER_DAN on node DAN. Assuming that Dan is not already in a conference, that attempt will fail. The utility will then send a message to all terminals on node DAN which have a UIC matching the username DAN. Dan will receive this message on his screen (provided that he has broadcast mode enabled on his port):

```
----------------------------------------------------------------
FRED  __QED_TT23  16-APR-1986 10:33:16.57
Please "CONFER QED_FRED"
----------------------------------------------------------------
```

4. From the above example, Dan may reply with the following command:

   DAN> CONFER QED_FRED

   The foregoing command causes the utility to try to connect with the CONFER_FRED process on node QED. That connection will result in all three processes dividing the screen up into three windows, with the third window labeled as follows:

```
-- DAN (DAN_TT0) ----------------------------------------------
```

5. Fred wants ideas for the Halloween party, so he types the following command:

   FRED> confer _* :logfile=hw.log "I need ideas for a Halloween Party"

   The foregoing command causes the message to be broadcast to all terminals on the local node. It forces the resulting conference into scrolling mode, and it records the conference proceedings in the file HW.LOG.

Using Prompts

```
> confer
Port or User  > _tt23
```

This performs the same function as the first example above.

Notes on Usage

The confer utility uses the VEW setup files and should operate correctly with any terminal type known to VEW. However, the line editing commands are those of the CIP, not of VEW.

Operational Modes

There are two operational modes in confer: window mode and scroll mode. These modes are defined as follows.

Window Mode

Additional conferees will cause the windows to become smaller and smaller. If evenly sized windows do not divide the screen evenly, a blank area will be indicated at the bottom of the screen.

If different users have different screen sizes, the utility will operate using the smallest screen size, so that all conferees will have a similar display. The right edge of the larger screens will be blocked off with a vertical line. When a user leaves the conference, the screen will be divided into fewer windows, and the total screen size may be increased if the user that left had a narrower screen than all remaining conferees.

Should windows become too small, the conference will be switched automatically to scroll mode.

Scroll Mode

Typed lines may be extremely long (up to 1023 characters); long lines are scrolled horizontally during input and word-wrapped on output.

Comments are added to the screen at the third from the last line. The oldest comments are scrolled off at the top line. The bottom two lines are used for input, for messages, and to display interactive data entry of other conferees. Each comment on the screen (or in the log file) is preceded by the name of the contributor, as in a court reporter's document or a play script, except for the originators comments, which are preceded by a dash ('-') instead of a username.

Scroll mode is entered in one of five ways:

1. when any user enters a conference with the :scroll switch
2. when window sizes grow too small in window mode
3. when any user executes the [ESC][ESC] sc command
4. when any user executes the [ESC][ESC] if command
5. when any user opens a log file

The switch to scroll mode is unilateral and irreversible. All terminals switch, and no one can change back.

Functions Available in confer

The confer utility has several internal functions available. Each of the following functions is executed by first striking [ESC] [ESC], then typing the two-letter mnemonic for the function on the command line that appears at the bottom of the screen (the same as other screen-oriented utilities such as VEW, Zap, Userprof, etc.). The position of the cursor when you initiate any of these functions does not affect the execution of the function, and the cursor returns to the same location when the function is complete.

Dismiss a user from the conference.

FUNCTIONAL DESCRIPTION

The di function dismisses a user from the conference. It is "unkind" in operation, because there is no protection from any user dismissing any other user from the conference. The di function will take either a username or a nodename-devicename pair as the argument for who to dismiss. If a username is specified, all users with that name will be dismissed from the conference (except the initiator, if he or she has the same username).

CORRESPONDING CONTROL-KEY FUNCTIONS

None.

EXECUTION

Step 1 Strike **[ESC] [ESC]**

Step 2 Type **di**

This prompt appears at the bottom of the screen:

```
Dismiss Username>
```

Step 3 Type either the username or the nodename_devicename given in parentheses after each username. Uppercase and lowercase are not distinguished. Wildcard characters are permitted.

Step 4 Strike **[RETRN]**

If any attendee of the conference has the specified username or is using the specified device, that attendee is dismissed from the conference.

The following message appears at the bottom of your screen:

```
CONFER: <username or devicename> dismissed.
```

Shortly after this message appears, another message appears:

```
CONFER: <username> leaving conference.
```

In the foregoing messages, <username> is the name of the user that was dismissed. The second message appears at the bottom of the screen of all participants in the conference. Also, the following message appears, on the screen of all other attendees of the conference, in the window (or on the scroll line) of the user who initiated the dismissal:

```
<<CONFER: <username> dismissed>>
```

The following message appears at the bottom of the screen of the attendee who has been dismissed:

```
CONFER: You have been dismissed by <username>
```

This user's _confer_ process is then terminated.

ex

Exit the conference.

FUNCTIONAL DESCRIPTION

The _ex_ function terminates your participation in the conference and returns you to the CIP.

CORRESPONDING CONTROL-KEY FUNCTIONS

{-}

EXECUTION

Step 1 Strike **[ESC] [ESC]**

Step 2 Type **ex**

This message appears at the bottom the screen of all other participants of the conference:

```
CONFER: <username> leaving conference.
```

if

Insert a file into the conference for other participants to read.

FUNCTIONAL DESCRIPTION

The _if_ function is used to send the text of a file to all participants of a conference. The conference will be forced into scroll mode.

CORRESPONDING CONTROL-KEY FUNCTIONS

None.

EXECUTION

Step 1 Strike **[ESC] [ESC]**

Step 2 Type **if**

This prompt appears at the bottom of the screen:

```
Insert Filename>
```

Step 3 Type the name of the file you wish to send.

Step 4 Strike **[RETRN]**

If the conference was not in scroll mode, it is changed to scroll mode, and the following message is displayed on the screen of all participants:

```
Inserting file <full filename>, press [RETRN] when ready
```

If any conferee chooses not to view the file, he may press [CTRL] c at this point, and the file will not be written on his screen. If any character other that [CTRL] c or [RETRN] is pressed, the terminal beeps and waits for one of those two characters.

After a conferee presses [RETRN], the file begins scrolling across his screen. To have time to read the file, he or she may use [CTRL] s and [CTRL] q to stop and start (respectively) the scrolling. There is no way to stop looking at a file once it has started being written to the screen. When the entire file has scrolled across the screen, this message appears:

```
<full filename> inserted.
```

in

Invite another user to join your conference, or join your conference with another conference.

FUNCTIONAL DESCRIPTION

The _in_ function lets you invite another user or users to join your conference that is already in progress, or you can use _in_ to join your conference with another conference. This command has the same functionality as the addressee parameter at the time _confer_ is executed.

CORRESPONDING CONTROL-KEY FUNCTIONS

{enter}

EXECUTION

Step 1 Strike **[ESC] [ESC]**

Step 2 Type **in**

This prompt appears at the bottom of the screen:

```
Invite>
```

Step 3 Type the name of the user or the device name of a port. The following message appears at the bottom of the screen:

```
CONFER: Inviting <username>
```

If the specified username is not currently logged on, a message saying "No devices found." appears on the screen. If a nonexistent devicename or unknown username is specified, an error message is displayed.

rs

Refresh the screen.

FUNCTIONAL DESCRIPTION

The rs function refreshes the display on the screen after a message or other interruption has disrupted the display.

CORRESPONDING CONTROL-KEY FUNCTIONS

[CTRL] _

EXECUTION

Step 1 Strike **[ESC] [ESC]**

Step 2 Type **rs**

The screen is redisplayed to remove any data that was not generated by confer.

sa

Start saving a log file of the conference.

FUNCTIONAL DESCRIPTION

The sa function starts a new log file containing a transcript of the conference. If a log file is already being kept, it is closed, and a new log file is started. If no name is given in response to the prompt, the current log file is closed, and logging will no longer occur.

CORRESPONDING CONTROL-KEY FUNCTIONS

{,}

EXECUTION

Step 1 Strike **[ESC][ESC]**

Step 2 Type **sa**

This prompt appears at the bottom the screen:

```
Log Filename>
```

Step 3 Type the name of the file to which you wish to begin logging.

Step 4 Strike **[RETRN]**

If the conference was not in scroll mode, it will be changed to scroll mode, and all subsequent messages will be written to the log file as well as to the screen of each participant in the conference.

sc

Start scroll mode.

FUNCTIONAL DESCRIPTION

The sc function will initiate scroll mode for the entire conference. Scroll mode is entered automatically if the conference is large (if there is not enough room on the smallest screen in the conference for at least three lines per attendee), if the if command is used (to allow each attendee time to read the file), or if a log file is requested by any conferee. Scroll mode can also be entered using this command, if desired. If any attendee of the conference initiates scroll mode, the entire conference changes to scroll mode. Once scroll mode is entered, the conference will remain in scroll mode until it is terminated. There is no way to return to window mode.

CORRESPONDING CONTROL-KEY FUNCTIONS

None.

EXECUTION

Step 1 Strike **[ESC][ESC]**

Step 2 Type **sc**

This message appears at the bottom the screen of all participants of the conference:

```
CONFER: Changing to SCROLL mode.
```

If the conference was already in scroll mode, this message will appear at the bottom of your screen:

```
Already in scroll mode.
```

ss

Show the number of participants in the conference, and their names.

FUNCTIONAL DESCRIPTION

The ss function prints at the bottom of your screen the total number of participants in the conference, their usernames, and their locations (nodename/devicename). The status line has the following format:

```
USERNAME(NODE_DEVNAME)
```

If logging is enabled, the status line is also inserted in the log file.

CORRESPONDING CONTROL-KEY FUNCTIONS

None.

EXECUTION

Step 1 Strike **[ESC][ESC]**

Step 2 Type **ss**

A message similar to this appears at the bottom of the screen:

```
STATUS: 5 persons in conference: JOE(QED_TT11) FRED(QED
_TT20) DAN(DAN_TT0) JOHN(QED_TT16) DAVE(DAVE_TT0)
```

Any unknown command will generate a help display. The format of the help display is this:

```
            CONFER HELP DISPLAY - Commands...
     DI - DIsmiss a user from conference impolitely
     EX - EXit from conference
     IF - Include File
     IN - INvite user <node_username> or terminal <__node_terminal>
     RS - Refresh Screen
     SA - SAve a log file of conference
     SC - change to SCroll mode
     SS - Show Status
     on most terminals: {-} is EX, {enter} is IN, {,} is SA
```

The help display is inserted on the screen, starting at the top line. An asterisk is displayed on the next-to-last line, indicating that confer is waiting for any key to be pressed. When you press a key, signaling that you have completed reading the help display, the screen is refreshed.

Related CIP Commands

---

send	Send a message to a list of terminals
talkt	Connect terminal to another terminal port

Functional Description

Use this command to reorganize a disk, that is, to make files contiguous. This reduces seek time, enabling the disk to have better throughput.

Command Line Syntax

Mnemonic	defrag			
Required parameter	Devicename			
Optional parameter	Message			
Switches				
Message	:bell :broadcast			
File selection	:before= :mod :uic=	:exclude= :since=	:files= :sort=	:filesize= :typeselect=
Other	:condense :stats	:hibernate :verbose	:names :verify	:pause

Parameters

Devicename	Function	Use this parameter to specify the name of the disk whose file system is to be reorganized.
	Default	None.
	Syntax	Type a valid device designation.
Message	Function	Use this parameter to specify an optional message to all users on the system, informing them why the disk is being reorganized, probable duration of execution, or other message of the operator's choice.
	Default	Unless you type :nobroadcast, the following kind of message is sent to all users on the system:
	Syntax	Type a string enclosed in double quotation marks.

```
SYSTEM __NODE_TTØ  14-Apr-1986 13:39:11.ØØ
DEFRAG beginning on _DCØ
```

Switches

:before=	Function	Use this switch to select only those files that match the :files= switch and were created/modified before the specified date and time.
	Default	Selects all files that match the :files= switch.
	Syntax	Type **:before=** followed by a date and/or time in in the standard date and time syntax.

:bell	Function	Use this switch to output a bell along with the message (unless suppressed with the :nobroadcast switch).
	Default	:bell; i.e., a bell is sounded when the message is printed.
	Syntax	Type **:nobell** to suppress the bell.
:broadcast	Function	Use this switch to suppress the broadcast message.
	Default	:broadcast; i.e. a message is broadcast to every mounted terminal when defrag starts and finishes.
	Syntax	Type **:nobroadcast** to suppress the broadcast message.
:condense	Function	Use this switch to suppress the condensing of extents of files. By default, defrag compresses the empty space between extents of a file in an attempt to generate larger blocks of free space for later use. This is done before any files are moved. In some instances, such as when defrag has been run recently or when more than 50 percent of the disk space is free, this condensing is unnecessary.
	Default	:condense; i.e. do the condensing of extents.
	Syntax	Type **:nocondense** to suppress the extent-condensing section of defrag from executing.
:exclude=	Function	Use this switch to select only those files that match the :files= switch and do not match any of the files specified as the value of the :exclude= switch.
	Default	Selects all files that match the :files= switch.
	Syntax	Type **:exclude=** followed by a list of file designations, separated by commas, any one of which may contain wildcard characters.

:files=	Function	Use this switch to select files to be reorganized first. This means that these files will be put closer to the center of the disk, requiring less seek time from the FCB file to the data in the file.
	Default	Reorganize files according to file type, then by file size within each file type, in this order: File system files, DIRECTORY files, IMAGE files, all others.
	Syntax	Type **:files=** followed by a list of file designations, separated by commas, any one of which may contain wildcard characters.
:filesize=	Function	Use this switch to select only those files that match the :files= switch and fall within the specified size range (in Kbytes).
	Default	Selects all files that match file selection criteria without regard to file size.
	Syntax	Type **:filesize=** followed by a valid range specification.
:hibernate	Function	Use this switch to hibernate all other processes on the system while defrag is running. Use this switch only when defrag cannot allocate the device, that is, when other processes have files open on the device. Note that if defrag is able to allocate the device, no processes will be hibernated, even if the :hibernate switch is specified.
	Default	:nohibernate; i.e. defrag tries to allocate the device and does not hibernate processes.
	Syntax	Type **:hibernate** to cause all process to be hibernated while defrag is running.
:mod	Function	Use this switch to specify that the modification date is to be used in all date and time considerations by the :before= or :since= switches.
	Default	:nomod; i.e. the creation date is used in all date and time considerations by the :before= or :since= switches.
	Syntax	Type **:mod** to use the modification date instead of the creation date.

:names	Function	Use this switch to suppress the full filename line of the :verbose status messages.
	Default	:names; i.e. if the :verbose messages are being output, also output the full filename on the next line.
	Syntax	Type **:nonames** to suppress the lookup and output of filenames as part of the verbose output.
:pause	Function	Use this switch to stop the display after each screen of information. The user can then press [RETRN] to advance one line, or any other character to advance to the next screen.
	Default	The value specified to the option command.
	Syntax	Type **:pause** or **:nopause** to override the default.
:since=	Function	Use this switch to select only those files that match the :files= switch and were created/modified since the specified date and time.
	Default	Selects all files that match the :files= switch.
	Syntax	Type **:since=** followed by a date and/or time in in the standard date and time syntax.
:stats	Function	Use this switch to request that various statistics be displayed at the completion of defrag. Note that if statistics are requested, additional execution time to prepare the statistics will be required.
	Default	:nostats; i.e. no statistics are displayed.
	Syntax	Type **:stats** to request a statistics display upon completion of defrag.
:typeselect=	Function	Use this switch to select only those files that match the :files= switch and are of the given file type.
	Default	Selects all files that match file selection criteria without regard to file type.
	Syntax	Type **:typeselect=** followed by a list of valid file-type specifications.

:uic=	Function	Use this switch to select only those files that match the :files= switch and are owned by the specified list of users.
	Default	Selects all files that match file selection criteria without regard to the owner of the files.
	Syntax	Type **:uic=** followed by a list of UICs or usernames.
:verbose	Function	Use this switch to request more status information while defrag is executing. If specified, defrag will output a message for each file that is currently being operated on, specifying the operation taking place and the FCB and sequence number of the file.
	Default	:noverbose; i.e. no progressive status information will be output.
	Syntax	Type **:verbose** to request log messages for every file moved.
:verify	Function	Use this switch to verify the correctness of the KSAM files built by defrag before continuing execution of the defrag process. If specified, defrag will output a message for each bad sector encountered on the disk, and also a message for any sectors that are not correctly accounted for by the KSAM files.
	Default	:verify; i.e. verify the KSAM files.
	Syntax	Type **:noverify** to suppress the verify procedure.

Examples

> defrag _dcØ

This command will reorganize the file system on device _DCØ. The following kind of report is displayed on the terminal:

```
--------------------------------------------
SYSTEM  __NODE_TTØ  14-Apr-1986 13:39:11.ØØ
DEFRAG beginning on _DCØ
--------------------------------------------
        Building free sectors KSAM file.
        Building sector usage KSAM file.
        _DCØ/ROOTDIR/FCB.SYS is contiguous
        _DCØ/ROOTDIR/FCBBITMAP.SYS is contiguous
        _DCØ/ROOTDIR/ROOTDIR.DIR being made contiguous
        Checking directory files
        Checking image files
        Checking remaining files
        Writing out bitmaps
--------------------------------------------
SYSTEM  __NODE_TTØ  14-Apr-1986 14:48:31.31
DEFRAG finished on _DCØ
--------------------------------------------
```

> defrag _dcØ :hibernate "some message" :nobell

This command will reorganize the file system on device _DCØ the same as the first example, except that the following message, without a bell, will be sent to all users on the system when defrag is beginning:

```
--------------------------------------------
SYSTEM  __NODE_TTØ  14-Apr-1986 13:45:34.23
DEFRAG beginning on _DCØ
some message
--------------------------------------------
```

If defrag is unsuccessful at allocating the device, it attempts to hibernate all other processes on the system. When defrag finishes, it awakens all the processes that were hibernated. Note that even though the :hibernate switch is specified, processes are not hibernated unless the device cannot be allocated. A device can be allocated if it is not already allocated, and if there are no open files on the device.

> defrag _dcØ :stats :nobroadcast

This command will reorganize the file system on device _DCØ. The following kind of report is displayed on the terminal:

```
        Building free sectors KSAM file.
        Building sector usage KSAM file.
        _DCØ/ROOTDIR/FCB.SYS is contiguous
        _DCØ/ROOTDIR/FCBBITMAP.SYS is contiguous
        _DCØ/ROOTDIR/ROOTDIR.DIR being made contiguous
        Checking directory files
        Checking image files
        Checking remaining files
        Writing out bitmaps
        Preparing statistics

                        ** DEFRAG Statistics **
              ** BEFORE **          |             ** AFTER **
            * Extent size *         |           * Extent size *
          1 sect        883   100%  |           1 sect        3      1%
    2 to  5 sects         2     Ø%  |     2 to  5 sects     179     90%
    6 to 2Ø sects         Ø     Ø%  |     6 to 2Ø sects      17      8%
   21 to 4Ø sects         Ø     Ø%  |    21 to 4Ø sects       Ø      Ø%
    over 4Ø sects         Ø     Ø%  |     over 4Ø sects       3      1%
                                    |
        * Number of extents *       |         * Number of extents *
          1 Xtnt          6     4%  |           1 Xtnt      205    100%
    2 to 1Ø Xtnts       195    95%  |     2 to 1Ø Xtnts       Ø      Ø%
   11 to 3Ø Xtnts         1     Ø%  |    11 to 3Ø Xtnts       Ø      Ø%
   31 to 7Ø Xtnts         1     Ø%  |    31 to 7Ø Xtnts       Ø      Ø%
    over 7Ø Xtnts         2     1%  |     over 7Ø Xtnts       Ø      Ø%
                                    |
        Total extents         885   |         Total extents         2Ø2
    Largest # extents         159   |     Largest # extents           1
       Largest extent           4   |        Largest extent         159
     Avg # of extents           4   |      Avg # of extents           1
FCBs - Free =        19, Primary =       2Ø5, Secondary =         Ø
Number files moved =        274, Number sectors moved =      1543
   Total fcb reads =       1825, Total fcb writes =       784
```

Before defrag starts the disk reorganization, it reads through the FCB.SYS file and determines the usage of every sector on the disk. While it is doing this, it builds the statistics for the "Before" section of the statistics display. During execution of defrag, it keeps statistics on the number of files and sectors moved and the number of reads and writes of the FCB.SYS file. If statistics are requested, then after the disk reorganization is completed, defrag reads though the FCB.SYS file once again for the purpose of gathering data for the "After" section of the statistics display. If the FCB.SYS file is very large, this may take a significant amount of time.

> defrag _dc0 :files=_dc0/mydir/* :nobroadcast

This command will reorganize the file system on device _DC0. The files in the directory MYDIR will be processed immediately after the main file system files (FCB.SYS, FCBBITMAP.SYS, BITMAP.SYS and ROOTDIR.DIR). This will make the files in the directory MYDIR the closest files to the center of the disk and the FCB file, so that the seek time for these files will be very small, and the response when accessing these files will be the best possible. The following kind of report is displayed on the terminal:

```
        Building free sectors KSAM file.
        Building sector usage KSAM file.
        _DC0/ROOTDIR/FCB.SYS is contiguous
        _DC0/ROOTDIR/FCBBITMAP.SYS is contiguous
        _DC0/ROOTDIR/ROOTDIR.DIR being made contiguous
        Checking User specified files
        Checking directory files
        Checking image files
        Checking remaining files
```

> defrag _df0 :verbose :stats :nobroadcast

This command will reorganize the file system on device _DF0. The verbose switch enables a file-by-file progress report to be output to the terminal. The following kind of report is displayed:

```
       Free Space on _DF0 is 28%
       Building free sectors KSAM file.
       Building sector usage KSAM file.
****** Verifying KSAM files
       _DC0/ROOTDIR/FCB.SYS being made contiguous
****** Moving extent 0,5 of _DF0//#212.212
       _DF0/ROOTDIR/SECTORUSE.DAT.1
****** Moving extent 0,9 of _DF0//#109.109
       _DF0/TEST3/DSTAT.HLP.1
                 .
                 .
                 .
****** Moving FCB.SYS to target location
       _DF0/ROOTDIR/FCBBITMAP.SYS being made contiguous
****** Moving extent 0,1 of _DF0//#2.2
       _DF0/ROOTDIR/FCBBITMAP.SYS.1
****** Moving file _DF0//#3.3
       _DF0/ROOTDIR/BITMAP.SYS.1
****** Moving file _DF0//#1.1
       _DF0/ROOTDIR/ROOTDIR.DIR.1
****** Now moving in _DF0//#3.3
       _DF0/ROOTDIR/BITMAP.SYS.1
```

```
****** Now moving in _DFØ//#2.2
       _DFØ/ROOTDIR/FCBBITMAP.SYS.1
       _DFØ/ROOTDIR/ROOTDIR.DIR being made contiguous
****** Moving extent Ø,2 of _DFØ//#1ØØ.1ØØ
       _DFØ/TEST1/DUMP.HLP.1
****** Now moving in _DFØ//#1.1
       _DFØ/ROOTDIR/ROOTDIR.DIR.1
****** Condensing extents for _DFØ//#2Ø3.2Ø6
       _DFØ/ROOTDIR/DEFRAG.BAS.1
****** Condensing extents for _DFØ//#4.4
       _DFØ/ROOTDIR/TEST1.DIR.1
                 .
                 .
                 .
****** Condensing extents for _DFØ//#5.5
       _DFØ/ROOTDIR/TEST2.DIR.1
       Checking User specified files
****** Attempting to make space for _DFØ//#2Ø3.2Ø6
       _DFØ/MYDIR/DATABASE.DAT.1
****** Moving file _DFØ//#138.138
       _DFØ/TEST3/FSTAT.HLP.1
****** Moving file _DFØ//#136.136
       _DFØ/TEST2/FSTAT.HLP.1
                 .
                 .
                 .
****** Now moving in _DFØ//#2Ø3.2Ø6
       _DFØ/MYDIR/DATABASE.DAT.1
       Checking directory files
****** Attempting to make space for _DFØ//#4.4
       _DFØ/ROOTDIR/TEST1.DIR.1
****** Moving extent Ø,22 of _DFØ//#213.213
       _DFØ/ROOTDIR/SECTORUSE.KEY.1
****** Moving extent Ø,1 of _DFØ//#1ØØ.1ØØ
       _DFØ/TEST1/DUMP.HLP.1
****** Now moving in _DFØ//#4.4
       _DFØ/ROOTDIR/TEST1.DIR.1
****** Attempting to make space for _DFØ//#5.5
       _DFØ/ROOTDIR/TEST2.DIR.1
****** Moving extent Ø,21 of _DFØ//#213.213
       _DFØ/ROOTDIR/SECTORUSE.KEY.1
****** Moving extent Ø,2 of _DFØ//#1Ø1.1Ø1
       _DFØ/TEST2/DISPATCH.HLP.1
****** Now moving in _DFØ//#5.5
       _DFØ/ROOTDIR/TEST2.DIR.1
                 .
                 .
                 .
       Checking image files
****** Attempting to make space for _DFØ//#2Ø1.2Ø3
```

```
       _DFØ/MYDIR/MYIMAGE.EXE.1
****** Moving extent Ø,2 of _DFØ//#112.112
       _DFØ/TEST2/DSTAT.HLP.1
                .
                .
                .
****** Now moving in _DFØ//#201.2Ø3
       _DFØ/MYDIR/MYIMAGE.EXE.1
       Checking remaining files
****** Attempting to make space for _DFØ//#2Ø8.2Ø8
       _DFØ/ROOTDIR/MYFILE.PRN.1
****** Moving extent Ø,1 of _DFØ//#99.99
       _DFØ/TEST2/DIR.HLP.1
                .
                .
                .
****** Now moving in _DFØ//#2Ø8.2Ø8
       _DFØ/ROOTDIR/MYFILE.PRN.1
****** Attempting to make space for _DFØ//#2Ø2.2Ø5
       _DFØ/ROOTDIR/DISK.CKS.1
****** Moving extent Ø,2 of _DFØ//#87.87
       _DFØ/TEST1/DISPATCH.HLP.1
                .
                .
                .
****** Now moving in _DFØ//#2Ø2.2Ø5
       _DFØ/ROOTDIR/DISK.CKS.1
                .
                .
                .
****** Attempting to make space for _DFØ//#196.196
       _DFØ/TEST2/PASSWORD.HLP.1
****** Moving extent Ø,6 of _DFØ//#215.215
       _DFØ/ROOTDIR/SECTFREE.KEY.1
****** Now moving in _DFØ//#196.196
       _DFØ/TEST2/PASSWORD.HLP.1
****** Now moving in _DFØ//#197.197
       _DFØ/TEST3/PASSWORD.HLP.1
****** Now moving in _DFØ//#199.199
       _DFØ/TEST2/PIDLIST.HLP.1
****** Now moving in _DFØ//#2ØØ.2ØØ
       _DFØ/TEST1/PU.HLP.1
       Writing out bitmaps
       Preparing statistics
```

```
                         ** DEFRAG Statistics **
            ** BEFORE **            |               ** AFTER **
          * Extent size *           |             * Extent size *
            1 sect       883  100%  |             1 sect       3     1%
      2 to  5 sects        2    0%  |       2 to  5 sects    179    89%
      6 to 20 sects        0    0%  |       6 to 20 sects     17     8%
     21 to 40 sects        0    0%  |      21 to 40 sects      0     0%
      over 40 sects        0    0%  |       over 40 sects      3     1%
                                    |
        * Number of extents *       |          * Number of extents *
            1 Xtnt         6    3%  |             1 Xtnt     205   100%
      2 to 10 Xtnts      195   95%  |       2 to 10 Xtnts      0     0%
     11 to 30 Xtnts        1    0%  |      11 to 30 Xtnts      0     0%
     31 to 70 Xtnts        1    0%  |      31 to 70 Xtnts      0     0%
      over 70 Xtnts        2    1%  |       over 70 Xtnts      0     0%
                                    |
           Total extents      885   |           Total extents       202
      Largest # extents       159   |       Largest # extents         1
          Largest extent        4   |          Largest extent       159
        Avg # of extents        4   |        Avg # of extents         0
FCBs - Free =          19, Primary =        205, Secondary =        0
Number files moved =        274, Number sectors moved =       1543
   Total fcb reads =       1825, Total fcb writes =       784
```

Using Prompts

```
> defrag
Devicename     > _dc0
```

This command performs the same function as the first example.

Notes on Usage

The defrag utility is very closely related to the file system structure of the WMCS. The task that defrag performs is a reorganization of the file system on a disk.

After a disk has been in use for an extended period of time, it is likely that large numbers of files have been created and deleted. The process of creating and deleting files tends to fragment the free space on a disk until there are very few adjacent free sectors on the disk for the next file creation. Because the file system will always fulfill a disk allocation request if sectors are available, files are scattered randomly across the disk. However, since a typical file does not reside in contiguous sectors on the disk, several disk seeks are required when the file needs to be accessed. This causes the total disk throughput to decrease, since extra time is spent seeking for sectors, rather than just having to read consecutive sectors from the disk.

Disk performance is especially critical when trying to maintain streaming operations on a tape drive with data from disk. Given the tape controllers and drives that WICAT Systems currently has available when doing a backup of the disk, if the disk has to seek to more than the next track on the disk, the streaming effect will cease because data were not supplied to the tape in time to maintain streaming.

The _defrag_ utility performs the function of rearranging the sectors used by the various files on the disk to make all the sectors used by each file on the disk as contiguous as possible, and as a by-product, consolidating the free sectors on the disk into two large blocks near the front and end of the disk, so that future file creation may have large contiguous groups of sectors available for them.

In order to successfully execute _defrag_, there are several conditions that must be met. First, the process initiating _defrag_ must have the appropriate access to the disk. Since _defrag_ is such a powerful program, several privileges are required, unless the disk is privately owned by the user or process executing _defrag_. While _defrag_ is running, it must insure that no other process can access the disk. _Defrag_ does this by either allocating the disk or hibernating all other processes on the system. If the process executing _defrag_ owns the disk, and if the disk is not the system disk, no special privileges are required because the process can allocate the disk. Otherwise the required privileges are WORLD and GROUP access to the disk. If _defrag_ is not installed with BYPASS privilege (to let it read and write system control files such as FCB.SYS and BITMAP.SYS), then BYPASS privilege is also required.

The _defrag_ program also requires a certain amount of free space on the disk. The minimum amount of free space required is 15 percent. The reason for the free space requirement is that _defrag_ creates some files (described below) and also _defrag_ needs to be able to make a backup copy of a file before it changes the FCB to point at the new sectors on the disk, in order to make it less susceptible to crashes and power failures. After _defrag_ builds the KSAM files, it then does another space check. This check consists of totaling the sizes of the KSAM files (typically 6 to 8 percent of the disk) and the two largest files on the disk, which may be greater than the previously mentioned 15 percent. If there is not enough free space to accommodate this new requirement, _defrag_ will output a message to the effect that xxx free sectors are required to successfully run _defrag_ on the disk. It will then prompt to see if the operator wishes to continue anyway, or quit and attempt to generate more free space (by deleting unneeded files, or copying a few large files out to some secondary media). The defrag program will run in less than the optimum amount of free space, but it may not be able to move all the files. Note also that due to the organization of any particular disk, even if the amount of free space desired is available, defrag may still not be able to move all files.

The defrag program creates two KSAM files to manage the sectors on the disk while it is running. The two files are SECTORUSE.(KEY/DAT) which handles all the allocated sectors on the disk, and SECTFREE.(KEY/DAT) which handles all the free sectors on the disk. This requires that the KSAM class handler be loaded, if it is not already loaded. The two KSAM files are deleted by defrag as one of the last tasks it performs, so unless the machine crashes during a defrag, these files should never be seen. If the machine does happen to crash during defrag, you should run recover and delete the KSAM files, then run defrag again. The KSAM files are created in the default directory of the user that initiates the defrag command. This enables the operator to control where the KSAM files are created. If a second disk with sufficient space is available, it is usually a good idea to have the KSAM files created on a disk other than the target disk. If no other disk is available, defrag will correctly run with the KSAM files on the target disk, but will take longer, because there are more files to move around to make space for the desired files on the disk.

The defrag program may be terminated before it has run to completion. If the operator presses [CTRL] c while defrag is running, it will finish the current operation it is performing, write out the bitmaps and terminate. This sequence can possibly take some minutes, so there will not be any immediate reaction by defrag when [CTRL] c is pressed, but defrag will begin the termination process. If defrag is killed by another process, it is possible to lose some data, depending on whether it is doing an _write SVC in the operating system or not, so it is recommended that defrag never be killed by another process. If necessary, [CTRL] c may be used to stop defrag, and then it may be restarted. If nothing was done to the disk in the mean time, defrag will start again from the point where it was terminated. If the disk has been used, and a file larger that the last file being defragmented is created, or one of the defragmented files is deleted, defrag will have considerably more work to handle over again, as well as the initial setup (up to two hours on a large SMD disk).

If defrag runs entirely successfully, and there are no bad sectors on the disk, all the files will be contiguous, i.e., contained in one extent. It is possible that some files will not be entirely defragmented. There are two reasons for this occurrence. The first is that if there are bad sectors on the disk, defrag will split a file across the bad sector(s). This is essentially the same as if it did defrag successfully. The other possibility is if defrag was unable to move some files due to lack of space to create the open space for the file. In this case, if defrag is run again, it will usually handle these files in the second pass. In some cases, it may require several passes to get all files in the optimum defragmented state.

The _defrag_ program takes a long time to run, considerably longer than _recover_. On a 10-Mbyte Winchester disk, it takes from approximately 30 minutes to 3 hours, depending on how full the disk is, how badly the disk is fragmented, and the size of the files on the disk. For a 15-Mbyte Winchester, the typical range is about 45 minutes to 5 hours. For a 39-Mbyte Winchester disk, the time range is about 1 hour to 6 hours. For a 421-Mbyte SMD (Eagle), the time ranges from 12 hours to 48 hours and even longer in some cases.

Related CIP Commands

recover	Rebuild the file system on a corrupted disk

Functional Description

The _dmapper_ program is a disk utility intended to aid in recovering files from a bad disk. This utility is useful for identifying which files and locations within those files include particular sectors. It also does the inverse mapping identifying which sectors are used by a particular file.

Command Line Syntax

Mnemonic	dmapper			
Required parameter	Disk Name or Filename			
Switches				
display	:header	:pause	:offset=	:sector=
control	:track=			

Parameters

Filename	Function	This indicates that a file-to-sector mapping is to be displayed for the specified file.
	Default	None.
	Syntax	Type a valid file designation.
or		
Disk Name	Function	This indicates that a sector-to-file mapping is to be displayed for the specified disk.
	Default	None.
	Syntax	Type a valid device designation.

Switches

:header	Function	Use this switch to specify whether column headers should be displayed.
	Default	:header (headers are displayed).
	Syntax	Type **:noheader** to suppress column headers.
:offset=	Function	Use when a filename has been specified as the required parameter. This displays only those sectors in which the given offset resides. This offset is given in bytes from the start of the file.
	Default	All sectors used by a file are displayed.
	Syntax	Type **:offset=** followed by a valid numeric range.
:pause	Function	Use this switch to stop the display after each screen of information. The user can then press [RETRN] to advance one line, or any other character to advance to the next screen.
	Default	The value specified to the <u>option</u> command.
	Syntax	Type **:pause** or **:nopause** to override the default.

:sector=	Function	Use when a disk name has been specified as the required parameter. This displays only those files which use sectors in the specified range.
	Default	None; sector or track range must be specified.
	Syntax	Type **:sector=** followed by a valid numeric range.
:track=	Function	Use when a disk name has been specified as the required parameter. This displays only those files which use tracks in the specified range.
	Default	None; sector or track range must be specified.
	Syntax	Type **:track=** followed by a valid numeric range.

Examples

```
> dmapper _ds0/syslib/uaf.dat :offset=200-1500
```

This command will display the sectors that are used by the given file at the given file offset. The output will be similar to the following:

```
Sector map of __NODE_DS0/SYSLIB/UAF.DAT.1

File Offset  Sector #  Track #
-----------  --------  -------
   00000000    292702    11708
   00000400    292703    11708
```

Note that the file offset data is displayed in hexadecimal form on sector boundaries.

> dmapper _ds0 :sector=15 :track=346

This command will display the files that use the given sector and the given track. The output will be like this:

```
File map of __NODE_DS0

Sector #  Track #  File Offset  File
────────  ───────  ───────────  ──────────────────────────────────────
      15        0     00000000  __NODE_DS0/SOMEWHERE/TEST1.DAT.3
    8650      346     00000800  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8651      346     00000c00  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8652      346     00001000  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8653      346     00001400  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8654      346     00001800  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8655      346     00001c00  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8656      346     00002000  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8657      346     00002400  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8658      346     00002800  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8659      346     00002c00  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8660      346     00003000  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8661      346     00003400  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8662      346     00003800  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8663      346     00003c00  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8664      346     00004000  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8665      346     00004400  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8666      346     00004800  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8667      346     00004c00  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8668      346     00005000  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8669      346     00005400  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8670      346     00005800  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8671      346     00005c00  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8672      346     00006000  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8673      346     00006400  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
    8674      346     00006800  __NODE_DS0/DIR1.DIR2.DIR3/TEST2.DAT.1
```

Using Prompts

```
> dmapper
File or Disk  > _ds0/syslib/uaf.dat :offset=200-1500
```

This perfoms the same function as the first example.

Notes on Usage

The dmapper program accepts a disk name or filename as a parameter and determines the mapping function to be performed. If a disk name is specified, dmapper will perform a sector-to-file mapping. If a filename is specified, dmapper will perform a file-to-sector mapping.

Related CIP Commands

chkd	Check a disk for bad sectors

Functional Description

This utility is typically used to compile and maintain programs by using a make control file. The make control file specifies the dependencies of the program on source files, header files, include files, object files, library files, etc. If any of the files upon which the end program depends has changed since the end program was last built, make will automatically rebuild the necessary pieces and produce the end program. Using the make control file, this utility will compare the creation date of a primary file to a list of secondary files. If one or more of the secondary files has a creation date later than the primary file, all following lines that begin with a vertical line, |, will be executed. These lines may be any valid CIP command line. If none of the secondary files has a creation date later than the primary file, all following lines that begin with | will be skipped. If an error is returned from an executing program, make will terminate.

Command Line Syntax

Mnemonic	make		
Optional parameters	Make Control File Parameter 1-10		
Switches	:comments	:pause	:verbose

Parameters

Make Control File	Function	This defines the name of the make control file which describes what make is supposed to do.
	Default	If no file is specified, it will use a file called MAKE.MAK in the current directory.
	Syntax	Type a valid file designation. If no extension is specified, an extension of .MAK will be appended to the name.
Parameter 1-10	Function	Up to ten parameters may be passed on the command line down to the make control file. These parameters will be assigned to the logical names p1, p2, p3, ... pN respectively.
	Default	None of these logical names will be defined.
	Syntax	Type any valid command-line parameter. Note that items inside double quotes will be taken as one parameter.

Switches

:comments	Function	Use this switch to enable the display of comment lines while make is executing.
	Default	:comments (comments are displayed)
	Syntax	Type **:nocomments** to suppress the display of comment lines during execution.
:pause	Function	Use this switch to stop the display after each screen of information. The user can then press [RETRN] to advance one line, or any other character to advance to the next screen.
	Default	The value specified by the option command.
	Syntax	Type **:pause** or **:nopause** to override the default.

:verbose	Function	Use this switch to specify whether make should display the primary/secondary file comparison information.
	Default	:noverbose (do not display the information)
	Syntax	Type **:verbose** to display the file comparison information.

Examples

All the examples use the following make control file.

```
test.w          test.c test.h
                sys$disk/sysincl.sys/syserr.h
                sys$disk/sysincl.sys/fcbdisp.h
|compile test.c :noload 'pl'

testsub.w       testsub.c test.h
|compile testsub.c :noload 'pl'

test.exe        test.w testsub.w
|compile test.w,testsub.w 'pl'
```

> make

This command will look for a file called MAKE.MAK in the current directory. If this make control file has the contents shown above, and if the user has edited both .C source files since the last time a make was performed, then the creation dates for the two .C source files are later than the creation dates for each of the .W files. The output from the above command would be:

```
> compile test.c :noload
__node_ds0/users.test/test.c.0:
> compile testsub.c :noload
__node_ds0/users.test/testsub.c.0:
> compile test.w,testsub.w
test.exe:
```

Note that because the two .C files were recompiled, when it came time to compare the dates between the .EXE and the .W files, the .W files had a later creation date. This caused the .W files to be linked, thus creating a new .EXE file.

> make :verbose ":verbose"

With the same situation as in the above example, the output of this command would be:

```
test.w                                              -1986-04-25 22:39:15.22
   test.c                                            1986-04-25 22:40:25.78 *
   test.h                                            1986-04-24 16:36:18.27
   sys$disk/sysincl.sys/syserr.h                     1986-04-01 00:00:00.00
   sys$disk/sysincl.sys/fcbdisp.h                    1986-03-18 10:27:17.71
> compile test.c :noload :verbose
__node_ds0/users.test/test.c.0:
        sys$disk/ucc/cpp.exe >sys$tmp/systmp/CCCP2c2fa __node_ds0/users.test/test.c.
        sys$disk/ucc/ccom.exe sys$tmp/systmp/CCCP2c2fa sys$tmp/systmp/CCCQ2c2fb
        sys$disk/sysexe.sgs/alib2.exe -o sys$tmp/systmp/CCCR2c2fc sys$tmp/systmp/CCC
        sys$disk/sysexe.sgs/wimac.exe -O -o test.w sys$tmp/systmp/CCCR2c2fc
testsub.w                                           -1986-04-25 22:39:21.78
   testsub.c                                         1986-04-25 22:40:25.78 *
   test.h                                            1986-04-24 16:36:18.27
> compile testsub.c :noload :verbose
__node_ds0/users.test/testsub.c.0:
        sys$disk/ucc/cpp.exe >sys$tmp/systmp/CCCP2c34a __node_ds0/users.test/testsub
        sys$disk/ucc/ccom.exe sys$tmp/systmp/CCCP2c34a sys$tmp/systmp/CCCQ2c34b
        sys$disk/sysexe.sgs/alib2.exe -o sys$tmp/systmp/CCCR2c34c sys$tmp/systmp/CCC
        sys$disk/sysexe.sgs/wimac.exe -O -o testsub.w sys$tmp/systmp/CCCR2c34c
test.exe                                            -1986-04-25 22:39:31.65
   test.w                                            1986-04-25 22:40:58.58 *
   testsub.w                                         1986-04-25 22:41:05.70 *
> compile test.w,testsub.w :verbose
test.exe:
        sys$disk/sysexe.sgs/ll.exe <sys$tmp/systmp/CCCP2c39a
```

Note that secondary files whose dates are later than the primary file are flagged with an asterisk.

If the same command as in the previous example were executed again without changing any of the files, this would be the output:

```
test.w                                        -1986-04-24 16:53:28.93
   test.c                                      1986-04-24 16:53:14.38
   test.h                                      1986-04-24 16:36:18.27
   sys$disk/sysincl.sys/syserr.h               1986-04-01 00:00:00.00
   sys$disk/sysincl.sys/fcbdisp.h              1986-03-18 10:27:17.71
testsub.w                                     -1986-04-24 16:53:38.48
   testsub.c                                   1986-04-24 16:53:14.38
   test.h                                      1986-04-24 16:36:18.27
test.exe                                      -1986-04-24 16:53:55.58
   test.w                                      1986-04-24 16:53:28.93
   testsub.w                                   1986-04-24 16:53:38.48
```

> **make test**

This will look for a file called TEST.MAK in the current directory.

Using Prompts

None.

Notes on Usage

A make control file consists of one or more compilation units. A compilation unit consists of a primary file designation followed by a list of one or more secondary file designations (there is no limit to how many secondary files may be specified). This is followed by one or more execution lines (each beginning with |) that are to be executed if any of the secondary files have a creation date later than the primary file or if the primary file does not exist. There is no limit as to how many execution lines there may be.

Make control files have the following syntax:

- Lines beginning with the number sign, #, are comments to be displayed if the :comments switch is set (this is the default).

- Lines beginning with the exclamation point, !, are comments which are never displayed.

- Consecutive lines beginning with the vertical line, |, are commands to be executed if the preceeding primary file is not as recent as one or more of the associated secondary files, or if the primary file does not exist. Execution lines may be continued by a backslash, \, as the last character of the line. Anything that can be placed in a command file may be placed on these lines. Comment lines may be placed in the middle of a list of execution lines without affecting the execution flow.

- Lines beginning with the at sign, @, are nested _make_ control files. The default directory will be changed to the path of the given _make_ control file. Execution will then continue with the new _make_ control file. When processing of the sub-_make_ control file is completed, the default will be returned back to where it was and processing will continue with the original _make_ control file. Sub-_make_ control files may be nested as many levels as is desired.

- Lines beginning otherwise are filenames. The first file is the primary file; all others are secondary files. If the primary file does not exist, then the associated compilation unit will be executed. If a secondary file does not exist, an error will be reported and _make_ will terminate. Any combination of one or more tabs, spaces, or newlines are filename separators.

A compilation unit is terminated by encountering one of the following:

- Another compilation unit (lines that no longer begin with |).
- A sub-_make_ control file specification.
- The end of the _make_ control file.

Blank lines may be inserted anywhere inside a _make_ control file without affecting the flow of compilation units.

Use TOUCH (this is a logical name in LOCALUP.COM) to change a secondary file's creation date to the current date and time. With this a user can force a regeneration of a program without having to edit the files.

Any command that may appear in a command file may be placed on an execution line. Flow control commands (_goto_, _for_, _while_, _loop_, _call_, _if_) may not cross compilation units.

If an abort reason other than zero is returned from an execution line process, then _make_ will terminate immediately.

We wish to caution you about referencing files on different network nodes with make. Since each machine has its own local time definition and since they probably will not exactly agree, make's time comparisons between the primary and secondary files may not be valid and programs may be incorrectly built.

For an example of a make file, see SYS$DISK/SYSDSR/DISKCFG.MAK.

Related CIP Commands

cip	Execute a copy of CIP.EXE

Chapter 2

CIP Command Language

The WMCS User's Reference Manual describes the Command Interpreter Program (CIP).

This chapter describes the command language that has been added to the CIP. To use this language, you must understand the information in the WMCS User's Reference Manual as well as the information in this chapter.

This chapter is divided into two parts:

Information on general features of the language
A dictionary of statements available in the new language

Features of the Language

This section describes features and capabilities of the command language that are now part of the CIP.

Symbols

The CIP has a new class of variables called symbols. Symbols differ from logical names in that symbols are local to the CIP and are maintained inside the CIP. All symbols are maintained in string form. Logical names, on the other hand, are maintained inside the operating system and an SVC is executed every time a user accesses a logical name.

A symbol must begin with an alpha character and can contain any printable character other than expression operators (described under the Expressions heading in this chapter). All symbol names are mapped to upper case inside the CIP. A symbol array is defined by a symbol name, followed immediately by square brackets, with an expression inside the square brackets. The value of the expression is substituted for the expression. Symbol arrays can be nested within each other.

This is a list of sample symbol names:

```
a
this$is$a$very$long$symbol$name
test_symbol
test[1]
test[a*25]
test[idx[x-3]+4]
```

The following subheads contain information on the four types of symbols available in the CIP.

Note that the four groups of symbols are maintained in separate lists and that names can be duplicated across these lists without conflicts. In other words, you can have a symbol in one list whose name matches a symbol in another list and the CIP will not confuse the two symbols.

Symbol

These variables contain values that can be used in expressions. This is the most commonly used type of symbol.

Filelun Symbol

These symbols are used with the open, crfile, openpipe, close, read, write, and writeln statements (described later in this chapter). They contain the logical unit number (LUN) for a given open file. These symbols cannot be used in expression evaluation.

There are three built-in filelun symbols: input, output, and error. These can be used in any of the statements that are part of the command language.

Label Symbol

These symbols are defined by the label statements (described later in this chapter) and are referenced by goto and on statements. These symbols define a specific location in a command file. These symbols cannot be used in expression evaluation.

Procedure Symbol

These symbols are defined by the procedure statement (described later in this chapter) and are referenced by the call statement. These symbols define a subroutine in a command file. These symbols cannot be used in expression evaluation.

Expressions

The CIP now has the ability to evaluate expressions. These are the operators and the meaning of each:

Operator	Function
+	Binary addition
-	Binary subtraction and unary minus
*	Binary multiplication
/	Binary division
mod	Binary modulo
not	Unary logical not
lt	Logical binary less than
le	Logical binary less than or equal
gt	Logical binary greater than
ge	Logical binary greater than or equal
eq	Logical binary equal
ne	Logical binary not equal
lts	Logical string less than
les	Logical string less than or equal
gts	Logical string greater than
ges	Logical string greater than or equal
eqs	Logical string equal
nes	Logical string not equal
and	Logical and
or	Logical or
()	Parenthesis to control the order of evaluation

The following list illustrates the precedence of the operators. (Entries on the same line have equal precedence and are evaluated left to right.)

()	
- not	(unary operators)
* / mod	
+ -	
lt le gt ge lts les gts ges	(comparison operators)
eq ne eqs nes	(comparison operators)
and or	(logical)

Comparison operators have two forms:

> The operators without the trailing "s" do binary comparisons, converting the strings to binary and then comparing them.
>
> The operators with the trailing "s" do string comparisons, comparing the actual strings with each other.

All operators except the string comparison operators convert the symbol strings to binary before the operation is performed.

The specified operators work on four types of operands:

> Numeric literals: These are operands that begin with a number, a percent sign, %, or a dollar sign. If the operand begins with a number or a percent sign, the result is a decimal number. If the operand begins with a dollar sign, the result is a hexadecimal number.
>
> String literals: These are arbitrary strings surrounded by double quotation marks.
>
> Symbols
>
> Symbol arrays

Spaces and tabs between the operators and operands are insignificant except for those operators that are strings (not, mod, and, or, and the comparison operators). They must have at least one space or tab character between themselves and symbols.

The result of a comparison operation is a binary 1 if the comparison is true and a binary Ø if the comparison is false. The not operator changes a binary Ø to 1 or a binary 1 to Ø.

There are two predefined symbol names built into the CIP: true (which has a value of 1) and false (which has a value of Ø).

Expressions can be arbitrarily complex, and there is no limit to how deeply they can be nested.

Note that an expression need not contain an operator; it can simply be one of the operands.

These are examples of how to use the operators just described.

> **let a = 1Ø let b = "This is a test"**
>
> These commands assign to the symbols A and B the values of 1Ø and "This is a test" respectively. Note that numeric and string literals are acceptable expressions.
>
> **let a = 2Ø*tmp+tmp2/3**
>
> This command assigns the value of the given expression to symbol A. Note that the multiplication and division are done before the addition because of operator precedence.
>
> **if a gt 2Ø and b eqs "/" or c ne 3**
>
> This command performs all of the comparison operations before the and and or operations occur. Note that you can intermix string and binary comparisons.

Intrinsic functions

The CIP has several intrinsic functions. These functions can be placed anywhere on any command line. When the command line is parsed, the value of the function is substituted for the function call. All intrinsic functions begin with a percent sign, %, and are followed by the name of the intrinsic. The name is followed by an opening parenthesis, then all applicable parameters, and then a closing parenthesis. Logical name translations can be used as parameters to intrinsics. Intrinsic functions can be nested within each other to any level. All intrinsic parameters are treated as expressions and are evaluated. (Read the description of the let statement for a definition of expressions.)

These are the functions intrinsic to the CIP:

> **%copy(expression-1,expression-2[,expression-3])**
>
> Copy the number of characters specified by expression-3 from expression-1. Start the copy at the position specified by expression-2 (the first character of the string is position 1). If expression-2 is beyond the end of the string, an error will be reported. If expression-3 is missing, copy from the specified position to the end of the string.

%createtime(filelun symbol or expression)

Substitute the creation date and time of the specified file. If a filelun symbol is specified, the creation date and time of the specified open file is used. If an expression is specified, it is evaluated and the result is used as a file to open. The creation date and time of this file are used. The file is then closed.

%date()

Substitute the current day, month, and year.

%default()

Substitute the path for the default directory, in the following form:

__node_device/directory/.

%delete(expression-1,expression-2[,expression-3])

Delete the number of characters specified by expression-3 from expression-1. Start the deletion at the position specified by expression-2 (the first character of the string is position 1). If expression-2 is beyond the end of the string, an error will be reported. If expression-3 is missing, delete from the specified position to the end of the string.

%device(filelun symbol or expression)

Substitute the devicename part of the specified filename. If a filelun symbol is specified, this will use the complete name of the specified open file. If an expression is specified, it evaluates the expression and uses the result as a filename.

%directory(filelun symbol or expression[,dirnum])

Substitute the directory name part of the specified file designation. If a filelun symbol is specified, the function uses the complete name of the open file. If an expression is specified, it evaluates the expression and uses the result as a filename. If dirnum is zero or unspecified, the entire directory path is substituted. If dirnum is negative, the left most dirnum directories are substituted. If dirnum is positive, the rightmost dirnum directories are substituted.

%eval(expression)

Substitute the value of the given expression after it has been evaluated. This is used to force the evaluation of symbols and expressions anywhere on a command line.

%extension(FILELUN SYMBOL or EXPRESSION)

Substitute the extension part of the specified file designation. If a filelun symbol is specified, this function uses the complete name of the open file. If an expression is specified, it evaluates the expression and uses the result as a filename.

%file(filelun symbol or expression)

Substitute the filename part of the specified file designation. If a filelun symbol is specified, this function uses the complete name of the open file. If an expression is specified, it evaluates the expression and uses the result as a filename.

%insert(expression-1,expression-2,expression-3)

Substitute the string generated by inserting string expression-1 into string expression-2 in front of the character specified by the position at expression-3 (the first character of the string is position 1). If expression-3 is beyond the end of the string, an error is reported.

%length(expression)

Substitute the length of the evaluated string expression.

%lower(expression)

Substitute the evaluated string expression mapped to lower case.

%modtime(filelun symbol or expression)

Substitute the modification date and time of the specified file. If a filelun symbol is specified, the modification date and time of the specified open file are used. If an expression is specified, it is evaluated and the result is used as a file to open. The modification date and time of this file are used. The file is then closed.

%name(filelun symbol or expression)

Substitute the complete filename of the specified file. If a filelun symbol is specified, this will use the complete name of the given open file. If an expression is specified, it evaluates the expression and uses the result as a filename to open. If this succeeds, the complete filename is used. The file is then closed.

%node(filelun symbol or expression)

Substitute the node name part of the specified filename. If a filelun symbol is specified, this uses the complete name of the given open file. If an expression is specified, it evaluates the expression and uses the result as a filename.

%numparams()

Substitute the number of parameters that were specified on the command line to the CIP. This is valid only inside of command files.

%pos(expression-1,expression-2)

Substitute the position of expression-1 in expression-2. A value of zero is given if expression-1 is not found in expression-2.

%time()

Substitute the current system time--hours, minutes, seconds, and ticks.

%trans(expression)

Substitute the logical name translation of the evaluated string expression. If there is no translation, a null string is given.

%upper(expression)

Substitute the evaluated string expression mapped to upper case.

%version(filelun symbol or expression)

Substitute the version part of the specified filename. If a filelun symbol is specified, this uses the complete name of the given open file. If an expression is specified, it evaluates the expression and uses the result as a filename.

It is very important to realize that intrinsics do string substitution at the time the command line is parsed. Inasmuch as all parameters to the intrinsics are taken as expressions and are evaluated, you will usually need to put double quotation marks around the value of an intrinsic so that the output is taken as a string literal. If you do not, the output is taken as a symbol and is reevaluated.

The following are examples of intrinsic functions:

let a = "%default()"

Assigns the default directory to symbol a.

let b = "%lower("%directory("%default()",1)")"

Assigns to symbol b the right most subdirectory of the default directory mapped to lower case.

let c = "__%node("%default()")/%directory("%default()")/"

Assigns to symbol c the concatenation of "__" with the default node name followed by a slash, followed by the default directory, followed by a slash.

let d = "%delete("%time()",6)> "

Assigns to symbol d the first five characters of the current time (hours and minutes) followed by a right angle-bracket and a space. The time part is done by deleting from character six to the end of the string.

```
let e = "%name("'pl'")"
let f = "%copy(e,%pos("/","%copy(e,%pos("/",e)+1)")+%pos("/",e)+1)"
```

This example assigns to symbol e the pathname of the given parameter. It then assigns to the symbol f the filename, extension, and version portion of the pathname. This is done by locating the two slashes and getting the portion of the string immediately after the second slash. The above example can also be accomplished by executing the following line:

```
let f = "%file(e).%extension(e).%version(e)"
```

```
for i=1,i le %numparams(),i=i+1
  writeln "P%eval(i) = %trans("p%eval(i)")"
endfor
```

This _for_ loop displays the value of each parameter that was specified on the command line. If these commands are in a command file called FOR.COM, and if the following commands were executed, the following output would result:

```
@for this is "a test" of this
P1 = this
P2 = is
P3 = a test
P4 = of
P5 = this
```

Flow control constructs (_if_, _for_, _loop_, _while_, etc.) can also be entered interactively on the command line. If a starting flow control construct is entered, the CIP changes the current prompt to that construct's name followed by a number. The number is the current nesting level. Any command can be entered while in this state. If another starting flow control construct is entered (it can be the same one or a different one), the prompt is changed to that construct's name and the nesting level number is incremented. Note that all commands entered in this mode are not executed, but are saved for later execution. When an ending flow control contruct is entered (endif, endfor, endloop, endwhile, etc.), the nesting level is decremented and the command line prompt is changed back to the previous nesting construct's name. When the nesting level goes back to zero (the initial state, before the first starting flow control construct was entered), the CIP starts executing the commands beginning with the first flow control construct entered. When the flow control construct terminates, the user's prompt is changed back to its original state.

Multiple commands can be entered on the same line separated by semicolons (including loop control constructs).

Dictionary of CIP Statements

The WMCS User's Reference Manual contains descriptions for each of the commands constituting the CIP.

The descriptions in this section are for the CIP statements that have been added to the CIP as part of the command language. Most of these statements are new and are intended to be used almost exclusively in command files. Nevertheless, the commands marked with an asterisk existed previously and were described in the WMCS User's Reference Manual. Those commands have been enhanced for use with the new CIP command language.

call	for	log*	pause	scrnpos
cd*	goto	loop	pd	symbol
close	if	on	procedure	termopen
cmdst*	label	open	read	while
crfile	let	openpipe	return	write
echo	lineclr	option*	scrnclr	writeln

Functional Description

Use this command to call a procedure. The procedure must be defined elsewhere within the same command file, either before or after you invoke _call_. Upon return from the called procedure, control returns to the CIP statement immediately following the _call_.

Command Line Syntax

Mnemonic	call
Required parameter	Procedure name

Parameters

Procedure name	Function	Required. This parameter defines the name of the procedure to be called.
	Default	None.
	Syntax	Type a valid procedure symbol.

Switches

None.

Examples

call get_dir

This command calls a procedure named GET_DIR that is defined by a procedure command elsewhere in the same CIP command file. When GET_DIR concludes, control returns to the statement following call.

Notes on Usage

You define the functionality of the procedure specified in the call statement.

Calls can be nested to any level and are limited only by available memory.

Recursive calls are allowed.

No parameters are explicitly passed to the specified procedure; all variables used in a command file are global.

When the CIP encounters endprocedure or return in the specified procedure, control is transferred to the statement after call.

The specified procedure can be defined anywhere in the same command file containing the call statement. If the procedure is defined before the call statement, the CIP transfers control directly to the procedure. If the procedure definition appears after the call statement, the CIP searches for the definition and then transfers control. (If no definition is found, the CIP exits the command file and generates a diagnostic message.)

Procedure symbols are distinct from symbols, label symbols, and filelun symbols, and can therefore have the same names as symbols of other types.

Procedure symbols cannot be used in expressions.

Related CIP Commands and Statements

return	Return from a procedure
procedure	Define a procedure

Functional Description

Use this command to change the default directory.

Command Line Syntax

Mnemonic	cd	
Optional parameter	Directory	
Switches	:log	:perm

Parameters

Directory	Function	Optional. Use this parameter to specify the node, device, or directory that you want as your new default directory.
	Default	The directory you are in (that is, the directory that is already the default).
	Syntax	Use the standard syntax for directory paths. Wildcard symbols are not allowed.

Switches

:log	Function	Specifies whether or not log messages are displayed by utilities. (Log messages report on what the utility is doing.)
	Default	The value specified for the option command. (The default for option is :log.)
	Syntax	Type **:nolog** or **:log.**
:perm	Function	Specifies whether or not you want the change (specified by the command) to last even after execution of the current CIP is complete. For example, :perm in a command file makes the change specified on that line persist after execution of the command file is complete. Used with log, this switch logs you out of all nested command files and puts you in your last (or most recent) interactive CIP.
	Default	:noperm
	Syntax	Type **:perm** or **:noperm.**

Examples

cd

When you strike [RETRN] after typing the foregoing command, this kind of report appears on the screen:

```
__BARTLEBY_DS0/USERS.AL/
```

The foregoing report tells you what your default directory is.

cd _dc0/users

This command changes the default directory to /USERS/, on disk _DC0. The following display appears on your screen when you strike [RETRN]:

```
_DC0/USERS/
```

This report tells you that _DC0/USERS/ is now the default directory.

cd .march

This command changes the default directory to /MARCH/, a subdirectory of the default directory. (The period preceding "march" tells the CIP that /MARCH/ is a subdirectory of the default directory.) Note that this command does not change the default device. Were /MARCH/ a subdirectory of /USERS/, the following display would appear when you strike [RETRN]:

```
    _DC0/USERS.MARCH/
```

cd -

This command moves you to the parent directory of the directory you are in. For example, were you in _DC0/USERS.MARCH/, the foregoing command would move you to _DC0/USERS/.

cd _dc0/users.march/ :perm :nolog

The :perm switch has meaning only when you use cd in a command file. For example, when the execution of a command file ends, the CIP returns you to the directory you were in when you executed the file. When you use :perm with a cd command in a command file, the CIP puts you in that directory when execution of the command file ends--regardless of where you were when you executed the file. (If you use :perm with several cd commands in a command file, the CIP puts you in the directory specified by the last cd statement that was executed before the end of the command file.)

The :nolog switch tells the CIP not to display the report generated by cd to tell you what the default is.

Using Prompts

None.

Notes on Usage

If you type **cd** on the command line and strike **[RETRN]**, the designation for the default directory appears on the screen.

Related CIP Commands and Statements

def	Display the name of the default device and directory
crd	Create a directory
pd	Change back to your previous default directory

Functional Description

This command closes the file associated with the given filelun symbol. (See the crfile statement for information on assigning the filelun symbol.)

Command Line Syntax

Mnemonic	close
Required parameter	Filelun
Switches	:mode=

Parameters

Filelun	Function	Required. The name of the filelun symbol associated with the file to be closed. If the filelun symbol is undefined, a diagnostic message results.
	Default	None.
	Syntax	Type a filelun symbol.

Switches

:mode=

Function: Specifies the mode to be used when a file (or pipe) is created, opened, or closed.

Default: Depends on what you are doing to the file or pipe:

Action	Default
File: create	write
File: open	read
File: close	none desired
Pipe: open	read

Note that if you type this switch, you must also specify the kind of access you want (see the list, under syntax, of what you can specify). In other words, if you type **:mode=** and you want read access, you must type **read** as one of the values for this switch even though it may be the default.

Syntax: Type **:mode=** followed by any combination of the modes listed under the action you want to perform. If you specify more than one mode, separate the modes by commas. Unique abbreviations of the mode names are allowed.

For example, if you are creating a file, you can choose any (as well as any combination) of the modes listed under "Create a file", but if you are opening a pipe, only four modes are available.

Create a file:

append	noremote	read	zerodelete
delete	notruncfile	readlock	
nextfile	openifthere	write	
noreadahead	openshared	writelock	

Open a file:

append	noreadahead	openshared	write
delete	noremote	read	writelock
nextfile	notruncfile	readlock	zerodelete

Close a file:

delete	nodelete	supalldelete
forcedwrite	notruncfile	zerodelete

Open a pipe:

read	readlock	writelock	write

Examples

close fnam

This command closes the file associated with the filelun symbol fnam.

close fnam :mode=delete

This command closes the file associated with the filelun symbol fnam. After the file is closed, it will be deleted.

Using Prompts

None.

Notes on Usage

Close is generally used to close a data file from within a command file.

The status of this operation is assigned to the logical name SYS$RESULT.

Filelun symbols are distinct from symbols, label symbols, and procedure symbols, and can therefore have the same names as symbols of other types.

Filelun symbols cannot be used in expressions.

Related CIP Commands and Statements

crfile	Create a file and associate a filelun with it
open	Open a file and associate a filelun with it
openpipe	Open a pipe associated with output from a previous command
read	Read from a file
write	Write to a file
writeln	Write a line to a file

Functional Description

Use this command to manage the list of previous CIP commands entered at a user's terminal.

Command Line Syntax

Mnemonic	cmdst			
Switches	:delete	:full	:pause	:restore
	:restore=	:save	:save=	

Parameters

None.

Switches

:delete	Function	Deletes all commands in the current command stack.
	Default	:nodelete
	Syntax	Type **:delete** to delete all commands from the command stack.

:full	Function	Tells the CIP to display a list of all commands entered since the user logged on to the system.
	Default	:nofull
	Syntax	Type **:full** to display all commands in the command stack, or **:nofull** to display only the top 20 lines from the command stack (the 20 most recently entered or executed commands).
:pause	Function	Specifies whether or not the command pauses after displaying a screenful of information.
	Default	The value specified by the option command.
	Syntax	Type **:pause** or **:nopause** to override the default.
:restore	Function	Tells the CIP whether or not to load the command-stack buffer from CIPCMD.STK in your home directory. All commands already in the command-stack buffer are lost.
	Default	:norestore
	Syntax	Type **:restore** to load a new command stack from a file.
:restore=	Function	Tells the CIP the file designation for the file from which the CIP is to load the command-stack buffer. All commands already in the command-stack buffer are lost.
	Default	Do not load from a saved command stack file.
	Syntax	Type **:restore=** followed by a file designation.
:save	Function	Specifies whether or not the CIP saves the current command stack buffer in file CIPCMD.STK in your home directory. If CIPCMD.STK exists already, :save tells the CIP to replace it.
	Default	:nosave
	Syntax	Type **:save** to have the CIP save a copy of the command-stack buffer, or type **:nosave** to tell the CIP not to save a copy.
:save=	Function	Specifies the name of the file to which you want the CIP to save the current contents of the command-stack buffer. If the specified file exists already, the CIP deletes it before creating the new file by the same name.
	Default	Do not save the current command stack buffer.
	Syntax	Type **:save=** followed by a file designation.

Examples

cmdst

This command shows either all the commands you have executed or the last 20 commands you have executed, whichever is the smaller quantity. If this is the first command typed after you log on, the following display appears:

```
> cmdst
```

If you have been working in the CIP for several minutes, your display would look something like this:

```
> mnt _dxl
> copy *.dat _dxl/rootdir/*
> cd .work
> dir
> copy *.exe _dxl/*/* :build
> sp _dxl
> dmnt _dxl :auto
> def
> pu
> dm
> stat
> vew test.pas
> link test
> test
> dstat
> vew
> pas test.pas
> link test
> test
> cmdst
```

cmdst :full :pause

This command displays all of the information in the previous example plus all commands executed before the 20th command. The :pause switch causes the display to pause after each 20 commands.

cmdst :save

This command will save the current command stack in a file called CIPCMD.STK in the user's home directory.

cmdst

cmdst :restore=sys$disk/users.homedir/mysave.stk

This command will load the current command stack buffer from the file SYS$DISK/USERS.HOMEDIR/MYSAVE.STK. The previous contents of the command stack buffer are lost.

Using Prompts

None.

Notes on Usage

The CIP has an internal buffer of 1.5 Kbytes that lets it save between 80 and 140 of the previously executed commands. When the buffer is full, the commands begin scrolling through the buffer, and the oldest commands are lost.

Related CIP Commands and Statements

None.

Functional Description

This command creates a file and associates it with a filelun symbol.

Command Line Syntax

Mnemonic	crfile		
Required parameters	Filename Filelun		
Switches	:fileid=	:filetype=	:mode=
	:protection=	:uic=	

Parameters

Filename	Function	Required. The name of the file to be created.
	Default	None.
	Syntax	Type a file designation.
Filelun	Function	Required. The name of the filelun symbol to be associated with the file to be created. If the filelun symbol is already in use, the CIP generates a diagnostic message.
	Default	None.
	Syntax	Type a filelun symbol.

Switches

:fileid=	Function	Specifies the file ID to be used when creating a file.
	Default	Ø.
	Syntax	Type **:fileid=** followed by a number between Ø and 65535.
:filetype=	Function	Specifies the type of the file to be created.
	Default	Ø (See the list, below.)
	Syntax	Type **:filetype=** followed by a recognized WMCS file type, a file type alias, or a numeral.

The following file types are supported directly by the WMCS:

Name	Value	Description
data	Ø	normal data file
directory	1	directory file
image	2	image file
ksamdata	3	KSAM data file
ksamkey	4	KSAM key file
llimage	5	ll image file
archcont	6	archive continuation file
encrypt	7	encrypted file
system	8	system file
archive	9	archive file
cipcmd	1Ø	CIP cmd file
cobol	11	COBOL file
basic	12	BASIC file
pascal	13	Pascal file
object	14	object file

File types 15-255 are reserved by WICAT Systems for development and enhancement of the WMCS. File types 256-65535 can be defined by users.

:mode=	Function	Specifies the mode to be used when a file (or pipe) is created, opened, or closed.
	Default	Depends on what you are doing to the file or pipe:

Action	Default
File: create	write
File: open	read
File: close	none desired
Pipe: open	read

Note that if you type this switch, you must also specify the kind of access you want (see the list, under syntax, of what you can specify). In other words, if you type **:mode=** and you want read access, you must type **read** as one of the values for this switch even though it may be the default.

Syntax: Type **:mode=** followed by any combination of the modes listed under the action you want to perform. If you specify more than one mode, separate the modes by commas. Unique abbreviations of the mode names are allowed.

For example, if you are creating a file, you can choose any (as well as any combination) of the modes listed under "Create a file", but if you are opening a pipe, only four modes are available.

Create a file:

append	noremote	read	zerodelete
delete	notruncfile	readlock	
nextfile	openifthere	write	
noreadahead	openshared	writelock	

Open a file:

append	noreadahead	openshared	write
delete	noremote	read	writelock
nextfile	notruncfile	readlock	zerodelete

Close a file:

delete	nodelete	supalldelete
forcedwrite	notruncfile	zerodelete

Open a pipe:

read	readlock	writelock	write

:protection=	Function	Specifies the protection mask for the file to be created.
	Default	The protection mask specified by the <u>option</u> command.
	Syntax	Type **:protection=** followed by a protection mask. Only the fields you specify in the protection mask are altered: unspecified fields remain unchanged.
:uic=	Function	Specify the UIC to be assigned to the new file.
	Default	Current uic of CIP.
	Syntax	Type **:uic=** followed by a UIC or username.

Examples

crfile report.dat fnam

This command creates the file REPORT.DAT in the default directory and associates the filelun symbol FNAM with it. The CIP will have write access to the file.

crfile myfile2.dat fnam :mode=read,write

This command creates the file MYFILE.DAT in the default directory and associates the filelun symbol FNAM with it. The CIP will have read and write access to the file.

Using Prompts

None.

Notes on Usage

Crfile is typically used for creating and accessing a data file from within a command file.

The status of this operation is assigned to the logical name SYS$RESULT.

Filelun symbols are distinct from symbols, label symbols, and procedure symbols, and can therefore have the same names as other types of symbols without conflict.

Filelun symbols cannot be used in expressions.

Related CIP Commands and Statements

close	Close a file specified by a filelun
open	Open a file and associate a filelun with it
openpipe	Open a pipe associated with output from a previous command
read	Read from a file
write	Write to a file
writeln	Write a line to a file

Functional Description

This command parses a command line and displays the result. It is useful for seeing exactly what a particular expression might look like after it has been processed by the CIP.

Command Line Syntax

Mnemonic	echo
Optional parameters	CIP string

Parameters

CIP String	Function	Optional. The string that is to be parsed by the CIP and then displayed.
	Default	Null string.
	Syntax	A CIP command.

Examples

echo the sky is blue

This command will write to SYS$OUTPUT the string "the sky is blue".

echo

echo %time()

This command writes the time to SYS$OUTPUT.

Using Prompts

None.

Notes on Usage

This is a CIP intrinsic which evaluates its parameters as a CIP expression. See the description of CIP (in this dictionary) for information on expressions.

Related CIP Commands and Statements

write	Write a string
writeln	Write a string, followed by a new-line

Functional Description

Use this command as a general-purpose looping construct within a CIP command file or from an interactive CIP. The <u>for</u> construct allows for an optional initial assignment statement, a conditional test expression, and another assignment statement (this is very similar to the **for** command in C).

Command Line Syntax

Mnemonic	for
	endfor

Optional parameters	Initial assignment statement Loop condition expression Post loop assignment statement

Parameters

Initial assignment statement	Function	Optional. This assignment statement is used to initialize the control variable and is performed only once, when the <u>for</u> loop is first encountered. A comma follows this parameter and separates it from the optional test expression. A null initial expression is permissible, in which case only a comma will precede the second expression.

	Default	No assignment operation is performed.
	Syntax	Type a CIP assignment statement. See the let statement for information on the syntax for assignment statements.
Loop condition expression	Function	Optional. This expression is used to test the control variable. The expression is tested at the beginning of each iteration of the for loop (including the first). Based on the result of this expression, two things may happen: If the result is true (the expression is not equal to zero), all lines immediately after the for statement will be executed until an endfor is encountered. Processing then jumps back to the for statement for re-execution. If the result of the expression is false (the expression is equal to zero), all lines immediately after the for statement will not be executed until an endfor is encountered. Normal processing then resumes.
	Default	A null parameter is evaluated as false.
	Syntax	Type a CIP expression.
Post loop assignment statement	Function	Optional. This statement is separated from the optional test expression by a comma. The assignment statement can be used to increment the control variable and is executed every time endfor is encountered, except after the for condition expression has failed and the for loop has been terminated.
	Default	No assignment operation is performed.
	Syntax	Type a CIP assignment statement. See the CIP let statement for information on syntax for assignment statements.

Examples

```
for a=1,a lt 5,a=a+1
  writeln "a=%eval(a)"
endfor
```

This example assigns the value 1 to variable A and then enters the _for_ loop. At the beginning of each iteration of the loop, including the first iteration, the conditional expression is evaluated. If the expression proves to be true, the main body of the loop is entered. The body of this loop contains one statement that writes the contents of the loop variable. Evaluation of loop statements continues until an _endfor_ statement is encountered. At that time the third parameter of the _for_ construct is executed. Control then returns to the top of the loop where the conditional statement is again evaluated.

```
let a=1
for ,a lt 5,
  for b=a, b lt 5,b=b+1
     writeln "a=%eval(a)   b=%eval(b)"
  endfor
  let a = a+1
endfor
```

This example shows two _for_ loops, one nested inside the other. _For_ loops can be nested to any level. The innermost _for_ and _endfor_ statements constitute one loop, and the outermost _for_ and _endfor_ statements constitute another loop. The outermost _for_ statement shows that optional expressions can be left null.

Notes on Usage

For loops can contain any CIP command as part of the body of the _for_. It is permissible to _goto_ a label outside the body of a _for_, but it is not permissible to _goto_ a label nested inside a _for_. _For_ loops can be nested to any level. The range of each loop is marked by an _endfor_ statement.

The control symbols that are assigned in the _for_ statements are normal symbols that can be used elsewhere. At the termination of the _for_, the control symbol has the value it had at the last compare time.

Related CIP Commands and Statements

if	Conditional execution of commands
loop	General purpose looping construct
while	Loop while a condition is true

Functional Description

This command will transfer the flow of a CIP command file to the specified label symbol.

Command Line Syntax

Mnemonic	goto
Required parameters	Label

Parameters

Label	Function	Required. Specifies the label of the location in the command file to which the CIP will transfer command execution.
	Default	None.
	Syntax	A CIP label symbol.

Examples

goto nextfoo

This command tells the CIP to transfer control to the label symbol "nextfoo".

Using Prompts

None.

Notes on Usage

A goto statement cannot reference the label of an inner or separate block (that is, a procedure, if, loop, or other nested constructs).

The label can be defined anywhere in the same command file that contains the goto statement. If the label has been defined in the command file before the goto statement, the CIP transfers control directly to the label. If the label is defined after the goto, the CIP searches for the label and then transfers control. (If no definition is found, the CIP exits the command file and generates a diagnostic message.)

Label symbols are distinct from symbols, procedure symbols, and filelun symbols, and can therefore have the same names as symbols of other types. Label symbols cannot be used in expressions.

Related CIP Commands and Statements

label	Declares a CIP label and marks the destination of a goto
on	Go to a label if certain error conditions occur

Functional Description

Use this command as a general-purpose flow-control construct to selectively execute instructions based on various conditions.

Command Line Syntax

Mnemonic	if
	else
	elseif
	endif

Optional parameters	If-expression Elseif-expression

Parameters

If-expression

Elseif-expression

Function Optional. This expression is evaluated each time the _if/elseif_ statement is encountered. Based on the result of this expression, one of several things happens:

If the result is true (the expression is not equal to zero), all lines immediately after the _if_ are executed until an _else_, _elseif_ or _endif_ is encountered. If an _else_ or _elseif_ is encountered, all subsequent lines are skipped until the _endif_ is encountered.

If the result of the expression is false (the expression is equal to zero), all lines immediately after the _if_ are not executed until an _else_, _elseif_, or _endif_ is encountered. If an _elseif_ is encountered, it evaluates that expression following the same rules as an _if_.

Once an _if_ or _elseif_ expression is determined to be _true_ and the commands associated with them have been executed, all following _elseif_ statements are not executed even if their expressions are true.

The statements following an _else_ are executed if the corresponding _if_ or _elseif_ expression was found to be _false_.

When the _endif_ is encountered, normal processing resumes.

Default	A null parameter is evaluated as false.
Syntax	Type a CIP expression.

Examples

```
if a*3+9 gt 5
   time
elseif b eqs "'pl'"
   dir
elseif c nes ""
   dstat
else
   pstat
endif
```

This statement evaluates each of the _if_ and _elseif_ expressions. The first expression that is true causes the command under it to be executed. Then all other commands are skipped until the _endif_ is encountered. If none of the _if_ or _elseif_ commands is true, the command under the _else_ is executed. Note that the indentation is unnecessary, but makes the code more readable.

Notes on Usage

_If_ statements can contain any CIP command as part of the body of the _if_. It is permissible to _goto_ a label outside the body of an _if_, but not to _goto_ a label nested inside the body of an _if_. _If_ statements can be nested to any level.

Related CIP Commands and Statements

loop	General purpose looping construct
while	Loop while a condition is true
for	Loop for a specified set of conditions

Functional Description

This command defines a label symbol at the current location. Labels are referenced by _on_ and _goto_ statements.

Command Line Syntax

Mnemonic	label
Required parameters	Label symbol

Parameters

Label symbol	Function	Required. Defines a label symbol that can be referenced by either _goto_ or _on_ statements.
	Default	None.
	Syntax	A CIP label symbol.

Examples

label nextfoo

This defines the label symbol nextfoo to be at the current location.

Using Prompts

None.

Notes on Usage

If a label is inside a procedure, if, loop, or other nested construct, it cannot be the target of a goto or on statement that is outside the nested construct.

The label can be defined anywhere within the command file containing the goto statement. If the label definition precedes the goto statement, the CIP transfers control directly to the label. If the label definition follows the goto, the CIP searches for the definition and then transfers control. (If no definition is found, the CIP exits the command file and generates a diagnostic message.)

Label symbols are distinct from symbols, procedure symbols, and filelun symbols, and can therefore have the same names as symbols of other types.

Label symbols cannot be used in expressions.

Related CIP Commands and Statements

goto	Begins program execution at a specified label
on	Go to a label if certain error conditions occur

Functional Description

This statement assigns an expression to a symbol.

Command Line Syntax

Mnemonic	let
Required parameter	Symbol
Optional parameter	Expression

Parameters

Symbol	Function	Required. This is the symbol to which the expression will be assigned. If this symbol has not been defined already, it will be defined after the assignment is made. If this symbol has a value already, the new value is assigned to the symbol.
	Default	None.
	Syntax	See the CIP command description for information on symbol syntax. This symbol <u>must</u> be followed by an equal sign.

Expression	Function	Optional. After this expression is evaluated (and if it is permissible), the value is assigned to the specified symbol.
	Default	A null string.
	Syntax	Type a CIP expression. See the CIP command for information on expression syntax.

Switches

None.

Examples

let a = 3*temp+9

This command assigns, to the symbol A, the value of the expression.

let a = "This is a test string"

This command assigns "This is a test string" to the symbol A.

let a =

This command assigns a null string to the symbol A.

Using Prompts

None.

Notes on Usage

Any permissible expression can be assigned to a symbol. Note that the _let_ and the equal sign are required.

Symbols are local to the current CIP only. They cannot be communicated from a child CIP to a parent CIP or vice versa. Once a symbol has been defined, there is no way to rid the current CIP of it.

Symbols are distinct from label symbols, procedure symbols, and filelun symbols and can therefore have the same names as symbols of other types.

Related CIP Commands and Statements

None.

Functional Description

This command erases from the current, or a specified, position to the end of a line on the screen.

Command Line Syntax

Mnemonic	lineclr
Optional parameters	Row number Column number

Parameters

Row number	Function	Optional. Specifies row number to clear from.
	Default	Current row number.
	Syntax	Any valid expression.
Column number	Function	Optional. Specifies the number of the column at which the clearance begins.
	Default	Current column number.
	Syntax	Any valid expression.

Examples

lineclr

This command erases from the cursor to the end of the line and leaves the cursor unaffected.

lineclr 17 4

This erases from row 17, column 4 to the end of the line and leaves the cursor at row 17, column 4.

Using Prompts

None.

Notes on Usage

The upper left-hand corner of the screen is position 1,1.

This is a CIP intrinsic that evaluates its parameters as a CIP expression.

You can omit the row and column parameters only if your terminal has a hardware erase-to-end-of-line command. If not, the CIP generates a diagnostic message when you try to execute the statement.

Specify both parameters, or none.

If termopen has not been called before the first invocation of lineclr, scrnclr, or scrnpos, termopen will be called automatically.

You can specify any set of coordinates: no check is performed to see whether the position is on the screen. Terminals may react differently and unpredictably if an invalid screen address is given.

Related CIP Commands and Statements

scrnclr	Clear to end of screen
scrnpos	Position screen cursor
termopen	Initialize screen routines

Functional Description

Use this command to terminate the CIP in which you are working.

Command Line Syntax

Mnemonic	log			
Switches	:hangup	:log	:perm	:result=
	:save			

Parameters

None.

Switches

:hangup	Function	Tells the CIP whether or not to send a hangup command to SYS$INPUT.
	Default	:hangup
	Syntax	Type **:nohangup** to keep the CIP from sending a hangup to SYS$INPUT.

:log	Function	Specifies whether or not log messages are displayed by utilities. (Log messages report on what the utility is doing.)
	Default	The value specified for the option command. (The default for option is :log.)
	Syntax	Type **:nolog** or **:log.**
:perm	Function	Specifies whether or not you want the change (specified by the command) to last even after execution of the current CIP is complete. For example, :perm in a command file makes the change specified on that line persist after execution of the command file is complete. Used with log, this switch logs you out of all nested command files and puts you in your last (or most recent) interactive CIP.
	Default	:noperm
	Syntax	Type **:perm** or **:noperm.**
:result=	Function	Tells the CIP whether or not to return an explicit abort reason to the parent process.
	Default	CIP returns an abort reason of zero.
	Syntax	Type **:result=** followed by a CIP expression.
:save	Function	Specifies whether or not the CIP saves the current command-stack buffer in file CIPCMD.STK in your home directory.
	Default	:save.
	Syntax	Type **:nosave** to keep the CIP from saving the copy of the command-stack buffer.

Examples

log

This command terminates the CIP the user is executing and displays the following kind of message:

```
AL logged off at 15-Apr-1985 08:40:37
```

Note that the username, date, and time are displayed when the user logs off.

log :nolog

This command terminates the CIP the user is executing, but suppresses the display of the log message.

log :perm

If this command is executed from inside a command file, it terminates all levels of the command file back to the previous interactive CIP. If LOG is executed from an interactive CIP, it terminates that CIP only. Then this kind of message appears:

```
AL logged off at 15-Apr-1985 08:40:37
```

log :result=1000

The parent process receives an abort reason of 1000. This is a way for a command file to signify status to a parent process or command file.

Using Prompts

None.

Notes on Usage

As the oldest parent CIP is terminating, it executes a command file called LOGOFF.COM located in SYS$DISK/SYSLIB. This command file in turn executes SYS$DISK/SYSLIB/LOCALOFF.COM and USEROFF.COM located in the user's home directory.

NOTE: If log is executed from within a command file or from a nested CIP, LOGOFF.COM, LOCALOFF.COM, and USEROFF.COM are not executed.

Related CIP Commands and Statements

cip	Execute a copy of CIP.EXE

Functional Description

Use this command as a general-purpose looping-construct within a CIP command file or from an interactive CIP. This statement instructs processing to loop while one or more given expressions are true.

Command Line Syntax

Mnemonic	loop exitif endloop
Optional parameter	Exit conditional expression (exitif only)

Parameters

Exit conditional expression	Function	Optional. This parameter specifies the condition under which the loop is to be exited. If the result of the expression is true (non-zero), then the loop is exited and control passes to the statement immediately following <u>endloop</u>. If the result of the expression is false (zero), control passes to the statement after <u>exitif</u>.

Default	If no expression is specified, a value of _false_ will be used.
Syntax	Type a CIP expression.

Examples

```
let a = 1
loop
  writeln "a=%eval(a)"
  let a = a + 1
  exitif a > 10
endloop
writeln "All done"
```

This example assigns the value 1 to the symbol A and then enters the loop. The body of this loop contains three statements. The first statement writes the contents of A. The next statement increments the symbol A. The next statement in this loop shows the use of _exitif_ that causes the loop to be exited when the value of the symbol A exceeds 10. Evaluation of _loop_ statements continues until an _endloop_ statement is encountered. Control then returns to the top of the loop.

Notes on Usage

Loop statements can contain any CIP command as part of the body of the loop. You can _goto_ a label outside the body of a loop, but not to a label nested inside a loop. _Loop_ statements can be nested to any level. The range of each loop is marked with _endloop_.

If a loop is not exited by _log_ or _goto_, then the only way to exit the loop is by satisfying the conditional expression of an _exitif_ statement within the body of the loop. Any number of _exitif_ statements (or none at all) can be specified in a _loop_.

Related CIP Commands and Statements

---

if	Conditional execution of commands
for	Loop for a specified set of conditions
while	Loop while a condition is true

Functional Description

This command defines what the CIP should do when it is executing a command file and a [CTRL] c is pressed, or an error or warning is returned from a command (or process) that it is executing.

Command Line Syntax

Mnemonic	on
Required parameters	Error type Error state

Parameters

Error type	Function	Required. This parameter specifies which of the three types of errors you want to handle.
	Default	None.
	Syntax	Type one or more of the following names separated by commas (unique abbreviations are allowed): **controlc, error, warning.**
Error state	Function	Required. This parameter specifies what will be done when the given state(s) occur(s).
	Default	None.
	Syntax	You can type **continue, abort,** or a label symbol defined in the command file. Continue means to ignore the error type(s) and go on. Abort means to terminate execution of the command file. A label symbol means to locate the given

label and transfer the flow of execution to that label. Note that there is no way to know exactly where you came from when a label is entered this way. Note that continue and abort are reserved words and cannot be used as label symbols with this command.

Switches

None.

Examples

on controlc continue

After this command is executed in a command file, if the user should press [CTRL] c, the CIP will ignore it and continue the normal execution of the command file.

on error,warning handle

After this command is executed in a command file, if an error or warning is returned from any command, the CIP will transfer process flow to the label symbol "handle".

Using Prompts

None.

Notes on Usage

When the CIP initiates execution of a command file, the default state of the three error types is:

Error type	Error state
controlc	abort
error	continue
warning	continue

You can have as many on statements as you want in a command file. In the case of conflicting on statements, the most recently executed statement is in effect.

This command has no meaning if executed from an interactive CIP.

Related CIP Commands and Statements

goto	Go to a label
label	Define a label

Functional Description

This command opens an existing file and associates it with the specified filelun symbol.

Command Line Syntax

Mnemonic	open
Required parameters	Filename Filelun
Switches	:mode=

Parameters

Filename	Function	Required. The name of the file to be opened.
	Default	None.
	Syntax	Type a file designation.
Filelun	Function	Required. The name of the filelun symbol to be associated with the file to be opened. If the filelun symbol is already in use, a diagnostic message results.
	Default	None.
	Syntax	Type a filelun symbol.

Switches

:mode=	Function	Specifies the mode to be used when a file (or pipe) is created, opened, or closed.
	Default	Depends on what you are doing to the file or pipe:

Action	Default
File: create	write
File: open	read
File: close	none desired
Pipe: open	read

Note that if you type this switch, you must also specify the kind of access you want (see the list, under syntax, of what you can specify). In other words, if you type **:mode=** and you want read access, you must type **read** as one of the values for this switch even though it may be the default.

Syntax

Type **:mode=** followed by any combination of the modes listed under the action you want to perform. If you specify more than one mode, separate the modes by commas. Unique abbreviations of the mode names are allowed.

For example, if you are creating a file, you can choose any (as well as any combination) of the modes listed under "Create a file", but if you are opening a pipe, only four modes are available.

Create a file:

append	noremote	read	zerodelete
delete	notruncfile	readlock	
nextfile	openifthere	write	
noreadahead	openshared	writelock	

Open a file:

append	noreadahead	openshared	write
delete	noremote	read	writelock
nextfile	notruncfile	readlock	zerodelete

Close a file:

delete	nodelete	supalldelete
forcedwrite	notruncfile	zerodelete

Open a pipe:

read	readlock	writelock	write

Examples

open "myfile.dat" fnam

This command opens the file MYFILE.DAT and associates the filelun symbol fnam with it. The CIP will have read access to the file.

open "myfile2.dat" fnam :mode=write,writelock,append

This command opens the file MYFILE.DAT and associates the filelun symbol fnam with it. The file is opened for write access. It will be opened write-locked to prevent any other users from opening the file while this process has the file open, and the file position will be set to the end of the file to facilitate appending data to the existing file.

Using Prompts

None.

Notes on Usage

Open is generally in a command file to access a data file.

A report of the status of this operation is sent to the logical name SYS$RESULT.

Filelun symbols are distinct from symbols, label symbols, and procedure symbols, and can therefore have the same names as symbols of other types. Filelun symbols cannot be used in expressions.

Related CIP Commands and Statements

crfile	Create a file and associate a filelun with it
close	Close a file specified by a filelun
openpipe	Open a pipe associated with output from a previous command
read	Read from a file
write	Write to a file
writeln	Write a line to a file

Functional Description

Use this command to read the output of a program whose output or error was directed to a pipe. Openpipe opens that pipe for reading with the associated filelun symbol.

Command Line Syntax

Mnemonic	openpipe
Required parameters	Filelun
Switches	:mode=

Parameters

Filelun	Function	Required. The name of the filelun symbol to be associated with the file opened via openpipe. If the filelun symbol is already in use, a diagnostic message results.
	Default	None.
	Syntax	Type a filelun symbol.

Switches

:mode=	Function	Specifies the mode to be used when a file (or pipe) is created, opened, or closed.
	Default	Depends on what you are doing to the file or pipe:

Action	Default
File: create	write
File: open	read
File: close	none desired
Pipe: open	read

Note that if you type this switch, you must also specify the kind of access you want (see the list, under syntax, of what you can specify). In other words, if you type **:mode=** and you want read access, you must type **read** as one of the values for this switch even though it may be the default.

Syntax: Type **:mode=** followed by any combination of the modes listed under the action you want to perform. If you specify more than one mode, separate the modes by commas. Unique abbreviations of the mode names are allowed.

For example, if you are creating a file, you can choose any (as well as any combination) of the modes listed under "Create a file", but if you are opening a pipe, only four modes are available.

Create a file:

append	noremote	read	zerodelete
delete	notruncfile	readlock	
nextfile	openifthere	write	
noreadahead	openshared	writelock	

Open a file:

append	noreadahead	openshared	write
delete	noremote	read	writelock
nextfile	notruncfile	readlock	zerodelete

Close a file:

delete	nodelete	supalldelete
forcedwrite	notruncfile	zerodelete

Open a pipe:

read	readlock	writelock	write

Examples

dir * :nohead :path :suppress=version |^openpipe flun

This command forks dir and redirects output and error to a pipe. It then opens that pipe for reading with the associated filelun symbol. The CIP will have read access to the pipe.

Using Prompts

None.

Notes on Usage

Openpipe is generally used from within a command file to facilitate CIP access of data generated by another process.

You should redirect both SYS$ERROR and SYS$OUTPUT to the pipe if the command will be submitted to a batch queue.

Filelun symbols are distinct from symbols, label symbols, and procedure symbols, and can therefore have the same names as symbols of other types. Filelun symbols cannot be used in expressions.

Related CIP Commands and Statements

crfile	Create a file and associate a filelun with it
close	Close a file specified by a filelun
open	Open a file for access by specified filelun
read	Read from a file
write	Write to a file
writeln	Write a line to a file

Functional Description

Use this command to display or specify a CIP control option.

Command Line Syntax

Mnemonic	option			
Switches				
Options for utilities	:error :pause	:log	:message	:path=
Options for CIP	:perm :verify	:prompt=	:protection=	:trace

Parameters

None.

Switches

:error	Function	Tells the CIP whether or not to have utilities display the standard 3-line diagnostic message when an error occurs.
	Default	:error
	Syntax	Type **:noerror** to prevent CIP utilities from displaying diagnostic messages.
:log	Function	Specifies whether or not log messages are displayed by utilities. (Log messages report on what the utility is doing.)
	Default	The value specified for the <u>option</u> command. (The default for <u>option</u> is :log.)
	Syntax	Type **:nolog** or **:log.**
:message	Function	Tells the CIP whether or not utilities are to display the third (or explanation) line of the standard 3-line diagnostic message that is sent to SYS$OUTPUT when an error occurs.
	Default	:message
	Syntax	Type **:nomessage** to suppress the explanation line of diagnostic messages.
:path=	Function	Specifies the path the CIP follows to search for image files and command files for which no path is specified when the filename is typed on the command line.
	Default	:path=,/sysexe/,/sysexe.sgs/,/sysexe.users/
	Syntax	Type **:path=** followed by a pathname or list of pathnames (items in a list are separated by commas). A null path means the default directory. If there is a leading slash (such as those shown, above, for the default) in a directory string, SYS$DISK is appended to the front of the directory string. If a leading slash is not part of a directory string, the CIP looks at the default disk. You can include devicenames in these strings.

:pause	Function	Specifies whether utilities will automatically pause after presenting a screenful of data.
	Default	:nopause
	Syntax	Type **:pause** to engage pausing, :nopause to disengage it.
		NOTE: You can override the setting of this switch, when you execute a utility, by typing :pause or :nopause on the command line.
:perm	Function	Specifies whether or not you want the change (specified by the command) to last even after execution of the current CIP is complete. For example, :perm in a command file makes the change specified on that line persist after execution of the command file is complete. Used with <u>log</u>, this switch logs you out of all nested command files and puts you in your last (or most recent) interactive CIP.
	Default	:noperm
	Syntax	Type **:perm** or **:noperm.**
:prompt=	Function	Tells the CIP what kind of prompt to print while the CIP waits for the next command.
	Default	A right angle-bracket is the default prompt for a CIP created at logon. All child CIPs have this prompt: CIP>
	Syntax	Type **:prompt=** followed by the prompt. If the prompt string contains an intrinsic function preceded by an accept character (for example, \%time()), the CIP re-evaluates the prompt string each time the prompt string is written.
:protection=	Function	Specifies the default protection mask for the CIP. When a utility creates a file, this protection mask is assigned to it.
	Default	The protection mask specified in SYSLIB/UAF.DAT.
	Syntax	Type **:protection=** followed by a protection mask. Only the fields you specify are modified: unspecified fields remain unchanged.

:trace	Function	Specifies whether or not you want to see what lines are being executed after the CIP has parsed the lines in the command file.
	Default	:notrace
	Syntax	Type **:trace** to show the lines as they are executed.
:verify	Function	Tells the CIP whether or not to display the lines in the command file, before they are executed, exactly as they appear in the file.
	Default	:noverify
	Syntax	Type **:verify** to have lines displayed before they are executed.

Examples

---

option

This command produces a display of the status of all CIP control options. For example:

```
SYS$RESULT : 0
Home       : SYS$DISK/USERS.TEST/
Path       : ,/sysexe/,/sysexe.sgs/,/sysexe.users/
Prompt     : TEST>
Protection : S:  RE,P:    ,G: WRE,O:DWRE
Username   : TEST
Error      : Yes
Log        : Yes
Message    : Yes
Pause      : No
Trace      : No
Verify     : No
```

option :verify :nomessage

The :verify switch causes all command files to display the contents of the command file before the file is executed. The :nomessage switch suppresses the message line of an error display when an error occurs.

option :prompt="Test> " :perm

This command changes the CIP prompt to Test>. If this command is executed inside a command file, it changes the prompt of the interactive CIP from which the command file was executed—the new prompt is Test>. By placing this kind of command in your USERUP.COM file you can define the kind of prompt assigned to you when you log on.

option :prompt="/\%directory("\%default()",1)/ \%copy("\%time()",1,5)> "

This command changes the user's prompt to show the default directory and the hours and minutes of the current time. Each time a new command is executed, the user's directory and the time are re-evaluated and the prompt is updated. Were the user in SYS$DISK/SYSEXE.USERS/ and were the time 10:14 am, the prompt would look like this:

```
/.USERS/ 10:14>
```

Using Prompts

None.

Notes on Usage

Use the :perm switch in command files to assign <u>option</u> characteristics for an interactive CIP. Note that <u>option</u> applies to only the most recent interactive CIP.

Related CIP Commands and Statements

None.

Functional Description

This command delays processing for a specified number of ticks (1ØØ ticks = 1 second).

Command Line Syntax

Mnemonic	pause
Optional parameters	Delay

Parameters

Delay	Function	Optional. Specifies number of ticks to delay (1ØØ ticks = 1 second).
	Default	Ø ticks.
	Syntax	A CIP expression.

Examples

pause 15Ø

This command delays processing for 1.5 seconds.

Using Prompts

None.

Notes on Usage

This is a CIP intrinsic that evaluates its parameters as a CIP expression.

Related CIP Commands and Statements

wait	Wait until specified time, then send a message

Functional Description

Every time a user executes cd, the directory just entered is saved on a stack. Pd moves the user to the top directory on the stack and then removes that entry from the stack. This stack holds only the last eight directories.

Command Line Syntax

Mnemonic	pd	
Switches	:log	:perm

Parameters

None.

Switches

:log	Function	Specifies whether or not log messages are displayed by utilities. (Log messages report on what the utility is doing.)
	Default	The value specified for the option command. (The default for option is :log.)
	Syntax	Type **:nolog** or **:log.**

:perm	Function	Specifies whether or not you want the change (specified by the command) to last even after execution of the current CIP is complete. For example, :perm in a command file makes the change specified on that line persist after execution of the command file is complete. Used with log, this switch logs you out of all nested command files and puts you in your last (or most recent) interactive CIP.
	Default	:noperm
	Syntax	Type **:perm** or **:noperm.**

Examples

---

cd .test
_DCØ/USERS.TEST/
pd
_DCØ/USERS/

In the foregoing example, the user changes the default directory from /USERS/to /USERS.TEST/. Pd then changes the default back to /USERS/.

Using Prompts

None.

Notes on Usage

If pd is executed and there are no more entries on the stack, pd does not change the default directory, but reports the name of the default directory already assigned.

Related CIP Commands and Statements

def	Display the name of the default device directory
crd	Create a directory
cd	Change to a given device and directory

Functional Description

Use this command to define a procedure that can be called by a CIP command file. A procedure can be executed only by using a call statement (that is, procedures are not executed as CIP encounters them in a command file, but only in response to a call that specifies that procedure). When a procedure is called, it will execute until it encounters either a return statement or an endprocedure statement. Any number of return statements can appear in a procedure definition.

Command Line Syntax

Mnemonic	procedure endprocedure
Required parameter	Procedure name

Parameters

Procedure name	Function	Required. Use this parameter to specify the name of the procedure you want to define.
	Default	None.
	Syntax	Type a procedure symbol.

Examples

```
procedure get_sysexe_dir
 if 'pl' eqs ""
   return
 endif
 dir sys$disk/sysexe/'pl' >exedir.dat :nohead :path :suppress=version
 if 'pl' eqs "*"
   return
 else
   myprogram 'pl'
 endif
endprocedure
```

This example defines a procedure named GET_SYSEXE_DIR. This procedure does the following. It returns immediately if the value of parameter Pl is a null string. If the Pl parameter is not null, a specialized directory listing of a particular file or set of files in SYS$DISK/ SYSEXE/ is made to an output file called EXEDIR.DAT. The procedure then returns immediately or executes another program based on the value of parameter Pl.

Notes on Usage

The programmer determines the functionality of a procedure. Procedures can manipulate global parameters, symbols, and logical names; call other procedures; invoke programs; use other commands and statements intrinsic to the CIP, etc.

A procedure can be of any length, limited only by available memory.

Control returns from a procedure to its caller when the CIP encounters a return or endprocedure.

The specified procedure can be defined anywhere in the command file that contains the call statement. If the definition of the procedure precedes the call statement, the CIP transfers control directly to the procedure. If the definition follows the call statement, the CIP searches for the definition and then transfers control. (If no definition is found, the CIP exits the command file and generates a diagnostic message.)

The return statement can appear anywhere in the procedure, even inside control structures such as loop, if, etc. If a return statement is found outside of a called procedure, the CIP exits the command file and generates a diagnostic message. A procedure can contain any number of return statements.

Endprocedure acts as an implicit return statement at the end of a procedure.

A procedure must have an endprocedure as its last statement regardless of whether there are any return statements in the procedure.

Related CIP Commands and Statements

---

call	Call a procedure

Functional Description

This command reads from an open filelun symbol into a symbol.

Command Line Syntax

Mnemonic	read		
Optional parameter	Filelun		
Required parameter	Symbol		
Switches	:editmode=	:emmodify=	:numrecs=
	:recnum=	:timeout=	

Parameters

Filelun	Function	Optional. The name of the filelun symbol associated with the file to be read. If the filelun symbol has not been defined, a diagnostic message results.
	Default	If only one parameter is given for this command, the CIP assumes that this is the symbol. The predefined filelun symbol SYSIN is used.
	Syntax	Type a filelun symbol.

Symbol	Function	Required. The name of the symbol to read the data into.
	Default	None.
	Syntax	A symbol.

Switches

:editmode=	Function	Specifies the edit mode to be used when reading.
	Default	lineall
	Syntax	Type **:editmode=** followed by one of the following edit-mode values: line linewchr raw lineall linewchrall
:emmodify=	Function	Specifies the modifiers to the edit mode switch.
	Default	None.
	Syntax	Type **:emmodify=** followed by any combination of the following modifiers, separated by commas (unique abbreviations are allowed): lock noecho
:numrecs=	Function	Specifies the number of bytes to be read.
	Default	127 bytes. Inasmuch as a line is usually requested, the amount of data actually read varies.
	Syntax	Type **:numrecs=** followed by an expression.
:recnum=	Function	Specifies the position in the file at which the operation is to be performed.
	Default	Current record position.
	Syntax	Type **:recnum=** followed by an expression.
:timeout=	Function	Specifies how long the CIP must wait before canceling the operation.
	Default	-1. This is essentially unending.
	Syntax	Type **:timeout=** followed by an expression. This number is in 1/100ths of a second.

Examples

read fnam mysymb

This command reads a line of data from the file associated with the filelun fnam and saves the data in the symbol mysymb.

read resp :timeout=1000

This command reads from SYS$INPUT and saves the data in the symbol resp. If nothing is read before the 10-second timeout has expired, a null string is inserted in the symbol resp. The status of this operation appears in the logical name SYS$RESULT.

Using Prompts

None.

Notes on Usage

Read is generally used for accessing a data file from a command file.

The status of this operation is sent to the logical name SYS$RESULT. A standard 3-line message display is given for all errors except ERRREADLEOF (140) and ERRTIMEOUT (128).

Filelun symbols are distinct from symbols, label symbols, and procedure symbols, and can therefore have the same names as symbols of other types. Filelun symbols cannot be used in expressions.

Related CIP Commands and Statements

close	Close a file specified by a filelun
crfile	Create a file and associate a filelun with it
open	Open a file and associate a filelun with it
openpipe	Open a pipe associated with output from a previous command
write	Write to a file
writeln	Write a line to a file

Functional Description

Use this command to return from a procedure.

Command Line Syntax

Mnemonic	return

Parameters

None.

Switches

None.

Examples

```
procedure test
  for i=1, i le 100, i=i+1
    if i >= ret
        return
    endif
    ...
  endfor
endprocedure
```

This procedure causes an immediate return from the procedure if the loop counter reaches the value of the variable ret.

Notes on Usage

A return can appear anywhere in a procedure. It can also be placed inside other flow-control constructs (for, if, while, etc.). A procedure can contain any number of return statements. Note that a return statement is unnecessary at the end of a procedure because endprocedure acts as an implicit return.

Even if a return statement is used, endprocedure is required to define the end of the body of the procedure.

Related CIP Commands and Statements

call	Call a procedure
procedure	Define a procedure

Functional Description

This command erases from the current position (or a given position) to the bottom of the screen.

Command Line Syntax

Mnemonic	scrnclr
Optional parameters	Row number Column number

Parameters

Row number	Function	Optional. Specifies row number from which clearance begins.
	Default	Current row number.
	Syntax	An expression.
Column number	Function	Optional. Specifies column number at which clearance begins.
	Default	Current column number.
	Syntax	An expression.

Examples

scrnclr

This command erases from the cursor to the end of the screen and leaves the cursor unaffected.

scrnclr 17 4

Erases from row 17, column 4 to the end of the screen and leaves the cursor at row 17, column 4.

Using Prompts

None.

Notes on Usage

The upper left-hand corner of the screen is position 1,1.

This is a CIP intrinsic that evaluates its parameters as a CIP expression.

You can omit the row and column parameters only if your terminal has a hardware erase-to-end-of-screen command. If not, a diagnostic message results when you omit the row and column numbers.

Specify both parameters, or neither.

If termopen has not been called before the first invocation of lineclr, scrnclr, or scrnpos, the CIP automatically calls termopen.

You can specify any coordinates—the CIP does not check to determine whether the specified position is on the screen. Terminals may react differently and unpredictably if an invalid screen address is given.

Related CIP Commands and Statements

lineclr	Clear to end of line
scrnpos	Position screen cursor
termopen	Initialize screen routines

Functional Description

This command positions the cursor.

Command Line Syntax

Mnemonic	scrnpos
Required parameters	Row number Column number

Parameters

Row number	Function	Required. Specifies the row-number coordinate.
	Default	None.
	Syntax	An expression.
Column number	Function	Required. Specifies the column-number coordinate.
	Default	None.
	Syntax	An expression.

Examples

scrnpos 17 4

This command positions the cursor at row 17, column 4.

Using Prompts

None.

Notes on Usage

The upper left-hand corner of the screen is position 1,1.

This is a CIP intrinsic that evaluates its parameters as a CIP expression.

If _termopen_ has not been called before the first invocation of _lineclr_, _scrnclr_, or _scrnpos_, the CIP automatically calls _termopen_ automatically.

You can specify any coordinates: the CIP does not check to see whether the specified position is on the screen. Different terminals react differently and unpredictably if an invalid screen address is given.

Related CIP Commands and Statements

lineclr	Clear to end of line
scrnclr	Clear to end of screen
termopen	Initialize screen routines

Functional Description

Use this command to display all symbols that have been defined in the current CIP, along with the current value of each.

Command Line Syntax

Mnemonic	symbol

Parameters

None.

Switches

None.

Examples

```
symbol
poem = "mary had a little lamb"
num = "354"
x= "-34"
```

In this example, the values of the symbols available to this process are displayed.

Notes on Usage

The CIP uses a hashing algorithm to store symbols. This algorithm provides rapid access to specific symbols, but does not lend itself to alphabetical or other ordered-storage organization. Therefore, the output from symbol is not in any recognizable order; it reflects the order of the symbols as they are found in the hash table. To obtain a sorted listing of symbols you must redirect the output of symbol to a file and then use wsort to sort the file.

Related CIP Commands and Statements

let	Assign an expression to a symbol
for	Loop for a specified set of conditions

Functional Description

This command initializes the terminal output routines of the CIP for a particular terminal.

Command Line Syntax

Mnemonic	termopen
Optional parameter	Setupout

Parameters

Setupout	Function	Optional. Specify the setup file to be used for terminal output from <u>lineclr</u>, <u>scrnclr</u>, and <u>scrnpos</u>.
	Default	Uses the setup file associated with SYS$OUTPUT.
	Syntax	The syntax for each allowable case is described below:

Just a name can be specified (for example, vt100). If you specify a name, the CIP searches for the setup file in SYS$DISK/SYSLIB.SETUP/. The given name will be used with the extension .STP if it was not specified already.

Just a path can be specified (for example, SYS$DISK/USER.SETUP/). In this case the CIP searches the specified directory for the setup file. A name is generated as described above. If no extension is specified, .STP is used.

A path and a name can be specified (for example, SYS$DISK/USER.SETUP/ABC). In this case the CIP searches for the specified file in the directory specified. If no extension is specified, .STP is used.

Examples

termopen

This command initializes the terminal output routines using the appropriate setup file for SYS$OUTPUT in SYS$DISK/SYSLIB.SETUP/.

termopen wit

This command initializes the terminal output routines using the setup file SYS$DISK/SYSLIB.SETUP/WIT.STP.

Using Prompts

None.

Notes on Usage

If _lineclr_, _scrnclr_, or _scrnpos_ are called before _termopen_ is called, they perform an implicit call to _termopen_ to initialize the screen parameters. It will function as if you did not specify a parameter to _termopen_.

Related CIP Commands and Statements

lineclr	Clear to end of line
scrnclr	Clear to end of screen
scrnpos	Position screen cursor

Functional Description

Use this command as a general-purpose looping construct within a CIP command file or from an interactive CIP. This construct loops while a given expression is true.

Command Line Syntax

Mnemonic	while
	endwhile

Optional parameter	Expression

Parameters

Expression	Function	Optional. This expression is evaluated each time the _while_ statement is encountered. Based on the result of this expression, two things happen:

If the result is true (the expression is not equal to zero), all lines immediately after _while_ are executed until _endwhile_ is encountered. Control then returns to the _while_ statement for re-execution.

If the result of the expression is false (the expression is equal to zero), all lines immediately after while are not executed until endwhile is encountered. Normal processing then continues.

Default	A null string is evaluated as false.
Syntax	Type an expression.

Examples

```
let a = 1
while (a lt 10)
  writeln "a=%eval(a)"
  let a = a + 1
endwhile
writeln "All done"
```

This example assigns the value 1 to variable A and then enters the while loop. At the beginning of each iteration of the loop, the conditional expression is evaluated, and if the expression is true, the main body of the loop is entered. The body of this loop contains two statements. One statement writes the value of the symbol A. The other statement increments the symbol A. Evaluation of loop statements continues until an endwhile statement is encountered. Control then returns to the top of the while statement, where the conditional statement is reevaluated. If the conditional statement is found to be false, control is transferred to the statement immediately after the endwhile statement that belongs to the while statement.

Notes on Usage

While statements can contain any CIP command as part of the body of the while statement. You can goto a label outside the body of a while statement, but not to a label nested inside the body of a while statement. While statements can be nested to any level.

Related CIP Commands and Statements

if	Conditional execution of commands
loop	General purpose looping construct
for	Loop for a specified set of conditions

Functional Description

This command writes to an open filelun symbol from a local symbol, expression, or literal string.

Command Line Syntax

Mnemonic	write		
Optional parameter	Filelun		
Required parameter	Expression		
Switches	:editmode= :recnum=	:emmodify= :timeout=	:numrecs=

Parameters

Filelun	Function	Optional. The name of the filelun symbol associated with the file to be written. If the filelun symbol is not defined, a diagnostic message results.
	Default	If only one parameter is given for this command, then it is assumed to be the symbol. The predefined filelun symbol SYSIN is used.
	Syntax	A filelun symbol.

Expression	Function	Required. This is the expression that is to be written to the specified filelun.
	Default	None.
	Syntax	Any CIP expression (including a literal string).

Switches

:editmode=	Function	Specifies the edit mode to be used when writing.
	Default	line.
	Syntax	Type **:editmode=** followed by one of the following edit-mode values: line raw
:emmodify=	Function	Specifies the modifiers to the edit mode switch.
	Default	none.
	Syntax	Type **:emmodify=** followed by any combination of the following modifiers, separated by commas (unique abbreviations are allowed): forcedwrite unlock
:numrecs=	Function	Specifies the number of bytes to be written.
	Default	The size of the expression.
	Syntax	Type **:numrecs=** followed by an expression.
:recnum=	Function	Specifies the position in the file at which the operation is to be performed.
	Default	Current record position.
	Syntax	Type **:recnum=** followed by an expression.
:timeout=	Function	Specifies how long the CIP must wait before canceling the operation.
	Default	-1. This is essentially unending.
	Syntax	Type **:timeout=** followed by an expression. This number is in 1/100ths of a second.

Examples

write fnam mysymb

This command writes the data from the symbol mysymb to the file associated with the filelun fnam.

write "This is a string to write"

This command writes the specified string to SYS$OUTPUT.

Using Prompts

None.

Notes on Usage

Write is generally used for accessing a data file from a command file.

The report of the status of this operation is sent to the logical name SYS$RESULT. A standard 3-line message display is given for all errors except ERRREADLEOF (140) and ERRTIMEOUT (128).

Filelun symbols are distinct from symbols, label symbols, and procedure symbols, and can therefore have the same names as symbols of other types. Filelun symbols cannot be used in expressions.

Related CIP Commands and Statements

close	Close a file specified by a filelun
crfile	Create a file and associate a filelun with it
open	Open a file and associate a filelun with it
openpipe	Open a pipe associated with output from a previous command
read	Read from a file
writeln	Write a line to a file

Functional Description

This command writes to an open filelun symbol from a local symbol, expression, or literal string, and inserts a new line after the data written.

Command Line Syntax

Mnemonic	Writeln		
Optional parameter	Filelun		
Required parameter	Expression		
Switches	:editmode=	:errmodify=	:numrecs=
	:recnum=	:timeout=	

Parameters

Filelun	Function	Optional. The name of the filelun symbol associated with the file to be written. If the filelun symbol is not defined, a diagnostic message results.
	Default	If only one parameter is given for this command, it is assumed to be the symbol. The predefined filelun SYSIN is used.
	Syntax	A CIP filelun.

Expression	Function	Required. This is the expression that is to be written to the specified filelun.
	Default	None.
	Syntax	A CIP expression (including a literal string).

Switches

:editmode=	Function	Specifies the edit mode to be used when writing.
	Default	line.
	Syntax	Type **:editmode=** followed by one of the following edit-mode values: line raw
:emmodify=	Function	Specifies the modifiers to the edit mode switch.
	Default	none.
	Syntax	Type **:emmodify=** followed by any combination of the following modifiers, separated by commas (unique abbreviations are allowed): forcedwrite unlock
:numrecs=	Function	Specifies the number of bytes to be written.
	Default	The size of the expression.
	Syntax	Type **:numrecs=** followed by an expression.
:recnum=	Function	Specifies the position in the file at which the operation is to be performed.
	Default	Current record position.
	Syntax	Type **:recnum=** followed by an expression.
:timeout=	Function	Specifies how long the CIP must wait before canceling the operation.
	Default	-1. This is essentially unending.
	Syntax	Type **:timeout=** followed by an expression. This number is in 1/100ths of a second.

Examples

writeln fnam mysymb

This command writes the data from the symbol mysymb to the file associated with the filelun fnam. A new-line character is written after the end of the symbol.

writeln "This is a string to write"

This command writes the given string to SYS$OUTPUT. A new-line character is written after the end of the symbol.

Using Prompts

None.

Notes on Usage

Writeln is generally used to access a data file from within a command file.

A report of the status of this operation is sent to the logical name SYS$RESULT. A standard 3-line diagnostic message results for all errors except ERRREADLEOF (140) and ERRTIMEOUT (128).

Filelun symbols are distinct from symbols, label symbols, and procedure symbols, and can therefore have the same names as symbols of other types. Filelun symbols cannot be used in expressions.

Related CIP Commands and Statements

close	Close a file specified by a filelun
crfile	Create a file and associate a filelun with it
open	Open a file and associate a filelun with it
openpipe	Open a pipe associated with output from a previous command
read	Read from a file
write	Write a line to a file

Chapter 3

System Manager's Reference

This chapter describes modifications and new features that pertain to system management.

Booting the System

On systems equipped with SCSI disk or tape drives, the order of boot devices for default booting is SMD, SCSI, WD3. In other words, if an SMD controller is installed in the system, the boot ROMs will attempt to boot from drive ØAØ on the SMD controller. If that drive fails for any reason, or if there is no SMD controller present, the system will attempt to boot from the SCSI disk—if an SCSI host adapter is installed. Again, the system will attempt to boot from disk drive ØAØ. If that drive fails for any reason, or if no SCSI host adapter is installed, the system will attempt to boot from the WFC controller, drive ØAØ.

NOTE: If an SMD controller is installed but there is no drive ØAØ, the system may take up to 1.5 minutes before it determines there is no disk drive present. The delay is necessary to allow slow disks adequate time to spin up before continuing the boot process.

If the system cannot boot from any of the default boot devices, or if you are using the universal boot, the menu of available drive types appears on the screen. This menu contains, among others, the following entries (if an SCSI host adapter is installed on the system):

```
DS  - SCSI Disk Controller
TS  - SCSI Tape Controller
```

If you select either of these two options, a slightly different prompt comes up next. For all other devices, the next prompt is

```
Drive #, Board # (d,b):
```

But for the SCSI devices, the next prompt is

```
Drive #, Type #, Board # (dtb):
```

The extra field (Type #) is used by the SCSI drivers to select the address of the device on the SCSI bus. Your response to this prompt should be the WMCS drive ID of the drive from which you want to boot. For example, if you want to boot from SCSI drive 1AØ, you would type **1AØ** in response to the prompt.

The default drive IDs for disks and tapes on the SCSI bus are as follows:

Disk Name	Drive ID	Tape Name	Drive ID
_SDØ	ØAØ	_STØ	ØGØ
_SD1	1AØ	_ST1	ØFØ
_SD2	2AØ		
_SD3	3AØ		
_SD4	ØBØ		
_SD5	1BØ		
_SD6	2BØ		
_SD7	3BØ		
_SD8	ØCØ		
_SD9	1CØ		
_SD1Ø	2CØ		
_SD11	3CØ		
_SD12	ØDØ		
_SD13	1DØ		
_SD14	2DØ		
_SD15	3DØ		

In systems with the universal boot ROMs, the name of the boot device driver does not need to be accurately specified in the SYSCONFIG.xxx file (edited by <u>sysprof</u>), because the boot is automatically attempted from the SMD, SCSI, and WD3 controllers, as explained at the beginning of this section. However, the name of the boot device driver is still recorded in the SYSCONFIG.xxx file so that systems with older ROMs will still work by using the name in the file.

TTY Networking

You can now do point-to-point networking using RS 232 serial connections across asynchronous TTY ports. Point-to-point means that computers can talk directly with other computers to which they have a direct TTYNET link. The full networking functionality implemented for ETHERNET is available on TTYNET, except that it is much slower.

As far as speed goes, TTYNET is just slightly slower than <u>usscopy</u> when used for copying files, because TTYNET uses a much more comprehensive protocol to gain the increased reliability and functionality.

TTYNET will run on any asynchronous I/O board supported by WICAT (i.e. Ø8Ø, 1Ø4, IPE and ICI boards). It is most efficient when running on IPE and ICI boards because much more of the interrupt overhead is handled by the boards. TTYNET does its extensive protocol processing at interrupt level 1, so it will not interfere with normal level 2 processing for other TTY I/O on normal ports.

Multiple TTYNET links may be connected between the same two host computers. The TTYNET code will automatically detect that there are multiple links, and it will share them evenly when transferring data. TTYNET shares the links based on data width (7 or 8 bit) and baud rate. A 96ØØ-baud link will get twice as much data as a 48ØØ-baud link. The protocol control packets will always be transferred on the fastest link. Only the data packets are split up across the multiple lines. This sharing of multiple TTYNET links adds a greater bandwidth to the data transmissions. However, adding more than 2-3 lines between two computers will probably not increase throughput.

TTYNET runs reliably on slow, modem-speed lines (12ØØ baud), but it is very slow and should not be used for more than one virtual circuit connection at a time. More than one circuit will work if the network activity is very light. But for instance, if the line is being used to transfer one 4-Kbyte packet on a virtual circuit and another user attempts to dial on a new virtual circuit over the same line, the 3Ø-second built-in timeout on the dial will expire before the dial can be completed, because the dial and dial-response packets are queued up after the 4-Kbyte packet.

TTYNET should run on 8-bit lines if at all possible. However it will support 7-bit data lines. During the initial configuration, and after transmission failures, small test-pattern packets are put out on the line. These test-pattern packets are used to determine the data width available on the line. If only 7 bits is reliable, the network code will enable both ends of the connection for 7 bit communication.

Any TTY ports which you desire to use for networking must first be mounted using the normal TTY drivers.

You must also mount a generic TTYNET device, which will not be associated with any particular hardware or port. Only one of these generic devices can be mounted. This generic TTYNET device contains the network layer and datalink layer code, which implements the TTYNET protocol. To mount the TTYNET device, execute the following command:

```
mnt _ttynet
```

Once the TTYNET driver is mounted and the appropriate RS 232 ports are also mounted, you can enable TTY networking on the RS 232 port by executing the following command (where "portname" is the name of the port):

```
dstat "portname" :network
```

The foregoing command causes the TTY driver to dynamically link up with the TTYNET driver, and from that time forward the TTY driver will act as an I/O device for the TTYNET driver.

As soon as a port is in network mode (enabled via <u>dstat</u>), it begins to periodically output configuration commands over the RS 232 line. When a remote terminal (on another node) begins to talk on the other end of the line, the two will "link up" with each other and exchange site IDs and nodenames. This process can take anywhere from 1Ø to 7Ø seconds, depending on when a network update packet occurs. Network update packets happen every minute, and the nodename exchange takes about ten seconds.

When you want to turn a TTYNET port back into a normal TTY port, execute the following command (where "portname" is the name of the port):

```
dstat "portname" :nonetwork
```

The foregoing command causes the TTY driver to unlink from the TTYNET driver and return to normal operation.

If you set a TTY port into network mode while the port is in use (currently open by a process), the port will be marked internally as "network pending" and will remain in normal TTY mode until the port is idle (last process closes), at which time the port will then switch into "network" mode. If a "network pending" port is set back into normal mode, the "network pending" state is cleared. This feature lets you dial into a remote machine, log on, set the remote port into network mode, and then log off. When you log off, the port will go into network mode.

NOTE: The CIP does an implicit hangup on a modem line during log off. To prevent this, execute the following command:

```
log :nohangup.
```

The "remote" bit of the device status flags (displayed or set with dstat) adds even more functionality. If the remote bit is set and a network connection loses its carrier (DSR drops), or if the connection is broken for any reason, then the port will revert back to normal TTY mode. With the operation of this bit, you can call into a remote machine as described above and set the remote bit on that port (on the remote machine). Then when the connection is broken on the remote machine, the remote port will go back into normal TTY mode so someone else can call in and log on to it.

When a port is put into network mode, it changes its class in the device table to TTYNET. One effect of this is that utilities such as dev and dstat will show that the device is a network device. Also, this prevents logflush from trying to fork logon on the port. When a port returns to normal TTY mode, its class is changed back to what it was before entering network mode.

Process Priority/Swapping

Preemptive interrupts have been implemented on TTY devices. This means that if a process wakes up and its priority is higher than the priority of the process currently scheduled, the WMCS will immediately do a context swap to the process that just woke up.

Preemptive interrupts are enabled by executing the dstat command with the :preempt switch.

Print Queue

All processes that are created by the QUEMGR will have the NOWATCHDOG attribute assigned. This way print jobs will not be killed by WATCHDOG.

SCSI tapes

The cartridge tape drives we are using for the SCSI tape subsystem are industry standard Ø.25-inch streaming tape drives. They use the standard QIC-36 interface. This interface has several limitations compared to the standard Ø.5-inch tape interface.

The first limitation is that the tape drives will write only 512-byte fixed blocks on the physical tape media. In order to maintain some compatibility with previous tape software, we simulate variable blocks in the driver software. To do that efficiently, we had to embed some control information in the physical tape blocks and to require that the block size be rounded up to the nearest 512-byte boundary, minus the space for the control information. That is why if a tape is initialized (with <u>dinit</u>) with a block size of 1024 bytes, the actual block size is 1018 bytes, or if it is initialized with a block size of 4096 bytes (a typical default), the actual block size is 4084 bytes. The tape software does all the conversion from user-requested block sizes to the actual block sizes. A side effect of this special blocking is that no logical block skipping is supported. Any forward tape positioning within a file will have to be done using read commands. No reverse tape positioning will be allowed at this time.

The second limitation is that these drives will not do reverse positioning themselves. Many utilities depend on the ability to skip back one or more files in order to reread them for various reasons. This requires that we simulate reverse file positioning in software. This can only be done by rewinding and then skipping forward the appropriate number of files to reach the desired location. This is reliable, but it is very slow. The slowness is particularly noticeable in the <u>backup</u> utility when using the :verify switch. We recommend that you not use :verify on SCSI tapes.

The third limitation is that these drives will write data only at BOT (physical beginning of tape) or EOD (the logical end of data on the tape). This means that to write on one of these tapes, you may only write at the end of the tape. To reuse a tape, you must reinitialize (with <u>dinit</u>) it every time you wish to start writing at some location other than EOD. This is because of the write electronics in the tape drive, and the tape format. The tape drive does not overwrite blocks, and it erases data only when writing on the first track, but when it does erase, it erases the whole tape. This causes the entire tape to be erased when the first track is written. This is a generic limitation of the class of 9-track serpentine 0.25-inch cartridge tape drives.

NOTE: Tapes made for the ADEI cartridge system are not compatible with tapes made for the SCSI system, and they cannot be interchanged. The SCSI cartridge drives require a different cartridge media (able to record at higher density) than does the ADEI.

"Hydra" Audio and Graphics

The new device drivers that support the "Hydra" audio and graphics boards are /SYSDSR/HAB$100.DSR (audio) and /SYSDSR/HGB$100.DSR (graphics). To make these new drivers transparent and to allow use of the old device names, logical names (hydra$audio$driver for audio, hydra$graphics$driver for graphics) are set to the appropriate driver names. A new command file, /SYSLIB/SETHYDRA.COM, is called by /SYSLIB/DEVICEUP.COM at boot time to determine which type of graphics and audio boards are present in the system. SETHYDRA.COM sets the logical names for the drivers accordingly. To run "Hydra" graphics or "Hydra" audio on a WICAT-proprietary bus system (Systems 2220 and 300), you _must_ remove the exclamation mark (comment mark) from the following line in the file / SYSLIB/DEVICEUP.COM:

```
@sys$disk/syslib/sethydra
```

Subdisk Devices

A subdisk device is a file (on a disk) that "looks" (to the CIP) like another disk. A subdisk can be created by executing the _dinit_ command. After the subdisk is created, it can be mounted and dismounted with the _mnt_ and _dmnt_ commands respectively. Dismounting the subdisk will not cause the subdisk file to be deleted.

When a subdisk device is initialized (with _dinit_) and the :format switch is specified, the subdisk driver allocates space to the subdisk file for the device's maximum size. If the :format switch is not specified, the subdisk driver allocates space to the subdisk file only as it is needed.

The name and size of the subdisk file can be determined the following ways:

1. Using the _sysprof_ command, you can specify the subdisk filename and the subdisk size.

2. When you initialize a disk (using _dinit_), you can specify one or both of the following switches:

   ```
   :subdkfname=
   :subdksize=
   ```

   These switches override the subdisk filename and subdisk size specified by _sysprof_.

3. When you mount the disk (using _mnt_), you can specify the following switch:

   ```
   :subdkfname=
   ```

   This switch overrides the subdisk filename specified by _sysprof_.

4. If the subdisk is not defined by any of the foregoing methods, the default filename of the subdisk will be SYS$DISK/ SYSDSR/"DEVNAME".DSK, where "DEVNAME" is the user-specified devicename. The maximum size of the subdisk will be determined by the :drivetype= switch (used with _dinit_).

When a subdisk device is initialized (with _dinit_), a drive type must be associated with it. The possible drive types for subdisks are SUBDISKA and SUBDISKB. Both define a subdisk of 512 Kbytes; SUBDISKA has a sector size of 0.5 Kbytes, and SUBDISKB has a sector size of 1 Kbyte.

NOTE: You can specify the size of a subdisk only if the SUBDISKA or SUBDISKB drive type is used.

When a subdisk is defined, you may use any of the drive type entries that are valid with the :drivetype= switch. This is useful if you want to build a copy of a smaller, slower disk on a larger, faster disk. You can define a subdisk using the appropriate drive type entry. Then you can mount the subdisk and build it up as desired. When the subdisk is complete, you can use the _xfer_ command to copy the subdisk file directly to the physical device. Note that this can be done with any of the drive type entries. The only requirement is that the entire subdisk file must reside on the given volume.

You may also take an existing device (like a floppy) and use _xfer_ to copy the device to a file on a larger disk.

A subdisk file does not need to reside on the same node that the device is mounted on. Since subdisks have their own local disk cache, access to subdisk files on remote machines is fast because the file system does not have to go across the network for all sectors that it may need.

Subdisks can have a sector size of 0.5 Kbytes or 1 Kbyte. It is not necessary for the device that the subdisk file resides on to have the same sector size, but it is more efficient if it does.

Driver Files

The following driver files are new and can be used to configure your system:

/SYSDSR/BSC$xxx.DSR	This driver used to be released as part of the RBTE product. It is now part of the WMCS because the driver is independent of the RBTE. This means that the RBTE will continue to work with the WMCS.
/SYSDSR/HAB$156.DSR	This is a new "Hydra" audio device driver that runs on the WICAT multi-bus systems (Systems 1250, 1255, and 1260).

/SYSDSR/HGB$156.DSR	This is a new "Hydra" graphics device driver that runs on the WICAT multi-bus systems (Systems 125Ø, 1255, and 126Ø).
/SYSDSR/SDISK$xxx.DSR	This is the new SCSI disk device driver.
/SYSDSR/STAPE$xxx.DSR	This is the new SCSI tape device driver.
/SYSDSR/SUBDK$xxx.DSR	This is the new subdisk device driver.
/SYSDSR/TTYNT$xxx.DSR	This is a new driver that will support TTYNET networking. It works in conjunction with the TTY$xxx.DSR driver.

Chapter 4

Programmer's Reference

This chapter contains the description of SVCs that were modified since the last printing of the WMCS Programmer's Reference Manual (May 1985): _getpcb, _hibern, _mount, _setfcb, _wake, and _write. It also includes descriptions of new SVCs that are not in the WMCS Programmer's Reference Manual.

Translate a string with a pid, return a BIG logical name.

Description:

Given a logical name, return the BIG equivalent. The equivalent string can be up to 4 Kbytes in length. If no translation can be found, the equivalent is a copy of the original.

When a translation for a name is found, the equivalent string will be translated again until one of the following occurs:

- The equivalent does not translate into anything else.
- The equivalent is defined in terms of itself, (a recursive definition is detected.
- The equivalent has been translated 16 times.

This feature allows logical names to be defined in terms of other logical names.

Given a pid, searches the logical name table of the specified process. If the name is not found, continues searching in the logical name table of the parent of the specified process, and so on with the grandparents until either the name is found or there are no other parents. If it is still not found, it will search the system logical name table.

Abbreviations are allowed in logical names. An asterisk (*) in the logical name is a marker that indicates the minimum string that is a recognized abbreviation of the logical name. For example, if the logical name is "PR*INT", a translation of any of the strings "PR", "PRI", "PRIN", or "PRINT" will return the equivalence.

If there is more than one occurrence of a name, the first one found is used. (Note that there can be only one instance of a given name in a process's logical name table)

Related Privileges:

none	- Allows the translation of logical names with the logical name table of any process with the same owner id and group id (uic) as the calling process.
group	- Allows the translation of logical names with the logical name table of any process with the same group id as the calling process.
world	- Allows the translation of logical names with the logical name table of any process.

Parameters:

pid	- A long word containing the process ID of the process whose logical name tables are to be used. Ø refers to the current process, -1 refers to the parent of the current process.
lname	- Address of a null terminated string containing the logical name to be translated. The maximum length of this string is 93 significant characters followed by a null. If this string is longer than 93 characters, the string is truncated, and no attempt is made to translate it.
equiv	- Address of an EQUIVSZ buffer to receive the equivalent string associated with the logical name. It can be up to 4 Kbytes in length. If the buffer is to small to hold the string, the string will be truncated. If an error is detected, this buffer will remain unmodified.
equivsz	- Holds the size of the equiv string buffer in bytes.
status	- Address of a long word to receive the result of the operation.

Diagnostics:

errinsufpriv	(1)	The process lacks the privileges required to perform the operation.
errprcsnotfnd	(2)	The specified process is not in the system process table.

See Also:

```
    _assign  - Assign a logical name
    _gassign - Assign a global logical name
    _getbglb - Retrieve a BIG global logical name.
    _getblog - Retrieve a BIG logical name.
    _getglb  - Retrieve a global logical name
    _getlog  - Retrieve a logical name.
    _tranpid - Translate another processes logical name.
    _trans   - Translate a logical name
```

Assembler Calling Sequence:

```
    push    pid                 ;value - process id
    push    lname               ;address - logical name
    push    equiv               ;address - equivalent string
    push    equivsz             ;value - the size of the equiv buffer
    push    status              ;address - result of operation
    jsr     _btrnpid            ;translate another processes logical name
```

C function declaration:

```
                                /* translate another processes logical name */
    long                        /* returns result of operation */
    _btrnpid(pid, lname, equiv, equivsz)
            long pid;           /* process id */
            char lname[94];     /* logical name */
            char equiv[4096];   /* equivalent string */
            long equivsz;       /* size of equiv buffer*/
```

Fortran Subroutine Declaration:

```
    c                           ! translate another processes logical name
            subroutine btrnpid(pid, lname, equiv, equivsz, status)
                integer*4 pid          ! process id
                character*94 lname     ! logical name
                character*4096 equiv   ! equivalent string
                integer*4 equivsz      ! size of equiv buffer
                integer*4 status       ! result of operation
```

_btrnpid

Pascal Procedure Declaration:

```
    procedure _btrnpid(        {** translate another processes logical name}
            pid     : longint;        {** process id}
            lname   : string[93];     {** logical name}
        var equiv   : string[4095];   {** equivalent string}
            equivsz : longint;        {** size of equiv buffer}
        var status  : longint;        {** result of operation}
    ); external;
```

Delete CPIR record

Description:

This svc is used to remove a file type from the system Create Process Indirection Record (CPIR) table. Once a file type is removed, users can no longer perform a _CRPROC svc call on a file of the given type.

Eventually the WMCS will be modified to have a different CPIR list per process and the user can access the SYSTEM CPIR list with the special id of -3 in the low word, thus the reason there is a PID parameter. But the per process list has not been implemented yet so the low word of the PID must be -3 for this to work. The high word of the PID is the siteid of the node to delete from. A value of zero will delete from the current node.

Related Privileges:

none	- The process is not allowed to delete CPIR records.
operator	- The process can delete any CPIR record.

Parameters:

pid	- The low word of the PID must be -3. The high word of the PID is the siteid of the node to insert into. A value of zero will do it to the current node.
filetype	- Contains the file type of the entry to delete.
status	- Address of a long word to receive the result of the operation.

Diagnostics:

errinsufpriv	(1)	The process lacks the privileges required to perform the operation.
errinvsiteid	(8)	The specified site id does not exist.
errnotimp	(45)	This item is not implemented yet.
erridxrange	(56)	The table ends before the specified occurrence.

See Also:

_getcpir - Get CPIR records.
_inscpir - Insert CPIR record.

Assembler Calling Sequence:

```
push    pid                     ;value - process id
push    filetype                ;value - file type
push    status                  ;address - result of operation
jsr     _delcpir                ;delete CPIR record
```

C Function Declaration:

```
                                /* delete CPIR record */
long                            /* returns result of operation */
_delcpir(pid, filetype)
        long pid;               /* process id */
        long filetype;          /* file type*/
```

Fortran Subroutine Declaration:

```
c                               ! delete CPIR record
        subroutine _delcpir(pid, filetype, status)
            integer*4 pid       ! process id
            integer*4 filetype  ! file type
            integer*4 status    ! result of operation
```

Pascal Procedure Declaration:

```
procedure _delcpir(             {** delete CPIR record}
        pid     : longint;      {** process id}
        filetype: longint;      {** file type}
    var status  : longint       {** result of operation}
); external;
```

Get a BIG global logical name

Description:

Given an index into a system's global logical name table, returns the logical name and equivalence associated with that index. The translation of this logical name can be up to 4 Kbytes in length.

Related Privileges:

None.

Parameters:

index	- which entry in the logical name table is desired.
siteid	- Site id of the system whose global logical name table is being accessed. Zero (Ø) corresponds to the system on which the calling process is executing.
lname	- Address of a 94 byte buffer to receive the logical name. String will be null terminated (up to 93 valid characters plus a null). If an error is detected, this buffer will remain unmodified.
equiv	- Address of an EQUIVSZ buffer to receive the equivalent string associated with the logical name. It can be up to 4 Kbytes in length. If the buffer is to small to hold the string, the string will be truncated. If an error is detected, this buffer will remain unmodified.
equivsz	- Holds the size of the equiv string buffer in bytes.
status	- Address of a long word to receive the result of the operation.

Diagnostics:

errprcsnotfnd	(2)	The specified process is not in the system process table.
errinvsiteid	(8)	The specified site id does not exist.
erridxrange	(56)	The table ends before the specified occurrence.

See Also:

```
_assign  - Assign a logical name
_btrnpid - Translate another processes BIG logical name.
_gassign - Assign a global logical name
_getblog - Retrieve a BIG logical name.
_getglb  - Retreive a global logical name
_getlog  - Retrieve a logical name.
_tranpid - Translate another processes logical name.
_trans   - Translate a logical name
```

Assembler Calling Sequence:

```
push    index               ;value - index into the table
push    siteid              ;value - system id
push    lname               ;address - logical name
push    equiv               ;address - equivalent
push    equivsz             ;value - the size of the equiv buffer
push    status              ;address - result of operation
jsr     _getbglb            ;retrieve a global logical name
```

C function declaration:

```
                            /* retrieve a global logical name */
long                        /* returns result of operation */
_getbglb( index, siteid, lname, equiv, equivsz)
        long index;         /* index into the table */
        long siteid;        /* system id */
        char lname[94];     /* logical name */
        char equiv[4096];   /* equivalent */
        long equivsz;       /* size of equiv buffer*/
```

Fortran Subroutine Declaration:

```
    c                              ! retrieve a global logical name
         subroutine getbgl(index, siteid, lname, equiv, equivsz, status)
             integer*4 index        ! index into the table
             integer*4 siteid       ! system id
             character*94 lname     ! logical name
             character*4096 equiv   ! equivalent
             integer*4 equivsz      ! size of equiv buffer
             integer*4 status       ! result of operation
```

Pascal Procedure Declaration:

```
    procedure _getbglb(          {** retrieve a global logical name}
            index   : longint;        {** index into the table}
            siteid  : longint;        {** system id}
        var lname   : string[93];     {** logical name}
        var equiv   : string[4095];   {** equivalent}
            equivsz : longint;        {** size of equiv buffer}
        var status  : longint         {** result of operation}
    ); external;
```

Get a BIG logical names

Description:

Given an index into a given process's logical name table, returns the logical name and equivalence associated with that index. The translation of this logical name can be up to 4 Kbytes in length.

Related Privileges:

None	- Allows retrieval of logical names from tables of processes with the same user and group id (uic) as the current process.
group	- Allows retrieval of logical names from tables of processes with the same group id as the current process.
world	- Allows retrieval of logical names from tables of any process in the system.

Parameters:

index	- which entry in the logical name table is desired.
pid	- Process id of the process whose logical name table is being accessed. Ø=current process, -1=parent process.
lname	- Address of a 94 byte buffer to receive the logical name. String will be null terminated (up to 93 valid characters plus a null). If an error is detected, this buffer will remain unmodified.

equiv	- Address of an EQUIVSZ buffer to receive the equivalent string associated with the logical name. It can be up to 4 Kbytes in length. If the buffer is to small to hold the string, the string will be truncated. If an error is detected, this buffer will remain unmodified.
equivsz	- Holds the size of the equiv string buffer in bytes.
status	- Address of a long word to receive the result of the operation.

Diagnostics:

errinsufpriv	(1)	The process lacks the privileges required to perform the operation.
errprcsnotfnd	(2)	The specified process is not in the system process table.
erridxrange	(56)	The table ends before the specified occurrence.

See Also:

_assign	- Assign a logical name
_btrnpid	- Translate another processes BIG logical name.
_gassign	- Assign a global logical name
_getbglb	- Retrieve a BIG global logical name.
_getglb	- Retreive a global logical name
_getlog	- Retrieve a logical name.
_tranpid	- Translate another processes logical name.
_trans	- Translate a logical name

Assembler Calling Sequence:

```
push    index              ;value - index into the table
push    pid                ;value - process id
push    lname              ;address - logical name
push    equiv              ;address - equivalent
push    equivsz            ;value - the size of the equiv buffer
push    status             ;address - result of operation
jsr     _getblog           ;retrieve a logical name
```

C function declaration:

```
                                    /* retrieve a logical name */
long                                /* returns result of operation */
_getblog( index, pid, lname, equiv, equivsz)
        long index;                 /* index into the table */
        long pid;                   /* process id */
        char lname[94];             /* logical name */
        char equiv[4096];           /* equivalent */
        long equivsz;               /* size of equiv buffer*/
```

Fortran Subroutine Declaration:

```
c                                   ! retrieve a logical name
        subroutine getblo(index, pid, lname, equiv, status)
            integer*4 index         ! index into the table
            integer*4 pid           ! process id
            character*94 lname      ! logical name
            character*4096 equiv    ! equivalent
            integer*4 equivsz       ! size of equiv buffer
            integer*4 status        ! result of operation
```

Pascal Procedure Declaration:

```
procedure _getblog(                 {** retrieve a logical name}
        index   : longint;          {** index into the table}
        pid     : longint;          {** process id}
    var lname   : string[93];       {** logical name}
    var equiv   : string[4095];     {** equivalent}
        equivsz : longint;          {** size of equiv buffer}
    var status  : longint           {** result of operation}
); external;
```

Get a CPIR record

Description:

This svc is used to obtain a list of CPIR records. Given an index into the system table of installed files, this call returns the corresponding entry, which is composed of a file type and a command-line string.

Eventually the WMCS will be modified to have a different CPIR list per process and the user can access the SYSTEM CPIR list with the special id of -3 in the low word, thus the reason there is a PID parameter. But the per process list has not been implemented yet so the low word of the PID must be -3 for this to work. The high word of the PID is the siteid of the node to get the list from. A value of zero will look at the list on the current node.

Related Privileges:

None.

Parameters:

pid	- The low word of the PID must be -3. The high word of the PID is the siteid of the node to insert into. A value of zero will do it to the current node.
index	- The index into the system table of the file type being requested. The first entry in the table has an index of zero.
fstr	- Address of a string to receive the command line string. The string may contain up to 93 significant characters and will be null terminated.
filetype	- Address of a long word to receive the file type.

status - Address of a long word to receive the result of the operation.

Diagnostics:

errinvsiteid	(8)	The specified site id does not exist.
errnotimp	(45)	This item is not implemented yet.
erridxrange	(56)	The table ends before the specified occurrence.

See Also:

_delcpir - delete CPIR record.
_inscpir - Insert CPIR record.

Assembler Calling Sequence:

```
push    pid                         ;value - process id
push    index                       ;value - index into table
push    fstr                        ;address - receives command line
push    filetype                    ;address - receives file type
push    status                      ;address - result of operation
jsr     _getcpir                    ;Get CPIR record
```

C Function Declaration:

```
                                    /* get CPIR record */
long                                /* returns result of operation */
_getcpir(pid, index, fstr, filetype)
        long pid;                   /* process id */
        long index;                 /* index into table */
        char fstr[94];              /* receives command line */
        long *filetype;             /* receives file type */
```

Fortran Subroutine Declaration:

```
c                                   ! get CPIR record
        subroutine getcpi(pid, index, fstr, filetype, status)
            integer*4 pid           ! process id
            integer*4 index         ! index into table
            character*94 fstr       ! receives command line
            integer*4 filetype      ! receives file type
            integer*4 status        ! result of operation
```

Pascal Procedure Declaration:

```
procedure _getcpir(                  {** get CPIR record}
        pid     : longint;           {** process id}
        index   : longint;           {** index into table}
    var fstr    : string[93];        {** receives command line}
    var filetype: longint;           {** receives file type}
    var status  : longint            {** result of operation}
); external;
```

Get process control block.

Description:

Given the process ID (PID) of a process in the system, copy the process control block (PCB) for that process into the buffer of the calling process.

CAUTION: The format of the process control block may change with each release of the operating system. The current definition is included in each release in the file named /SYSINCL.SYS/PCBDISP.*. The name of the record is "pcbtable", i.e. in your program, you can declare a variable whose type is "pcbtable".

The format of the PCB is as follows:

Name	Length (bytes)	Description
pcbnextlink	4	Forward link to next pcb on same priority level
pcbbacklink	4	Backward link to previous pcb on same priority level
pcbsysidnum	2	Contains the system ID number (the most significant word of the PID)
pcbidnum	2	Contains the least significant word of the PID
pcbname	16	The process name

pcbstatus	4	A bit encoded long word representing the process status. If the bit is asserted (1), the corresponding status applies.

Bit name	Bit #	Description
pcbsttoabort	Ø	Process is to be scheduled for deletion (i.e. the next time this process is scheduled, send it to the delete process routines)
	1	Reserved
pcbsttohibernate	2	Process is to be hibernated
pcbstabrinprgs	3	Process is currently being deleted. (i.e. process is currently executing the delete process routines)
pcbstexhinprgs	4	Process is executing its exit handler
pcbstrealtime	5	Process is in real time mode
pcbstswapped	6	Process has been swapped
pcbsthaschild	7	Process is in a child wait state
pcbstnocontc	8	Process may receive [CTRL] c without aborting
pcbstremchwait	9	If set process is waiting on remote child process
pcbsterrreport	1Ø	Process is reporting a system error
	11	Reserved
pcbstextndfcb	12	Process is extending the FCB.SYS file
pcbstbadseclog	13	Process is logging a bad sector
pcbstksam	14	Process is accessing a KSAM file
	15	Reserved
pcbstcrprcs	16	Process is loading an image
pcbstcleanup	17	Set when closing files while dying
pcbstinque	18	Process is waiting in a queue
pcbstcrashdisp	19	If set, suppress crash displays

pcbstalarmset	20	An alarm has been set
pcbstsupervisor	21	The call was issued while the processor was in supervisor mode
pcbstmulcrps	22	Multiple create process is in progress.
pcbstdisperr	23	If set, a crash report has been displayed
pcbsttracing	24	If set, process is tracing
pcbstfppending	25	If set, a floating point exception is pending
pcbstsurrogate	26	If set, this is an NSP for networking
pcbstsurrchild	27	If set, this is the child of a surrogate
	28-31	Reserved

pcbtimeslice	2	The process time slice value, i.e., the maximum amount of time (specified in .01 milliseconds. That is, a time slice of 100 represents 1 millisecond.) that the non-real time process will be allowed to run each time it is scheduled.
pcbmathtype	1	The type of floating point hardware in use The valid types are: 1 - skyl board 2 - ndp2 board 3 - ffpl board
pcbmathptr	1	The math pointer. Contains the index of this process's window on the hardware floating point board.
pcbprsize	2	The number of pages of memory currently allocated to this process. Each page is 4 Kbytes.
pcbprivilege	2	The privileges granted to the current process. This is a bit encoded field. The privilege is granted when the corresponding bit is set.

Bit Name	Bit #	Description
pcbpvsetpriv	0	setpriv - Process may assign more privileges than it currently has.
pcbpvsystem	1	system - Process has system access to files
pcbpvreadphys	2	readphys - Process can

		do physical read operations to devices and memory
pcbpvwritephys	3	writephys - Process can do physical write operations to devices and memory
pcbpvsetprior	4	setprior - Process can increase the process priority
pcbpvchngsuper	5	chngsuper - Process can change to supervisor mode of execution
pcbpvbypass	6	bypass - Process can access files and devices independently of file protection
pcbpvoperator	7	operator - Process can perform operator functions
pcbpvaltuic	8	altuic - Process can have access to files as though it had the same user and group i (uic) as the owner of the process image
pcbpvworld	9	world - Process can affect any process in the system
pcbpvgroup	10	group - Process can affect any process with the same group id as itself
pcbpvnetwork	11	network - Process can do network accesses
pcbpvsetattr	12	setattr - Process can modify its attributes
	13-15	Reserved

pcbuserid	2	The owner ID of the process (most significant word of the uic)
pcbgroupid	2	The group ID of the process (least significant word of the uic)
pcbchildpcbadr	4	Address of the pcb for the child process of this process
pcbparntpcbadr	4	Address of the pcb for the parent process of this process
pcbcurpriority	2	The current priority level
pcbalarmtime	8	The date and time at which to issue the alarm

pcbtimeout	8	The date and time at which the current operation will time out
pcbnondelcnt	2	Non-delete count
pcbcriticalcnt	2	Critical code count
pcbusp	4	The user stack pointer
pcbssp	4	The system stack pointer
pcbevntfl	4	The process event flags
pcbimgsiteid	2	Site ID of the image file
pcbattributes	2	Attributes pertaining to the current process. This is a bit encoded field. The attribute is given when the corresponding bit is set. Note that these offsets are defined for being in the high order word of a longword. Because it is only a word in the PCB, if you access the PCB directly you will have to subtract 16 from these numbers.

Bit Name	Bit #	Description
pcbattrdesencrypt	16	If set, do network encryption with DES algorithm
pcbattrfastencrypt	17	If set, do network encryption with fast algorithm
pcbattruser1	23	If set, user attribute bit 1
pcbattruser2	24	If set, user attribute bit 2
pcbattruser3	25	If set, user attribute bit 3
pcbattruser4	26	If set, user attribute bit 4
pcbattrnowatchdog	27	If set, cannot be killed by WATCHDOG utility
pcbattrnotswappable	28	If set, cannot swap this process
pcbattrprezeromem	29	If set, pages are zeroed as they are allocated
pcbattrpostzeromem	30	If set, pages are zeroed as they are released
pcbattrforceset	31	If set, other set bits will be set

pcbimgdevseqnum	2	The mount sequence number of the device that contains the image file from which this process was initiated

pcbimgfcbnum	4	The FCB number of the image file from which this process was initiated
pcbimgseqnum	2	The sequence number of the image file from which this process was initiated
pcbstacktop	4	Address of the top of the system stack
pcbparabortsts	4	Address of where to put status in parent
pcbexithdr	4	Address of the process's exit handler
pcbabortreason	4	A code indicating why this process terminated
pcblogiclink	4	Address of the logical name table for process
pcblogicque	4	Queue for linking logical names
pcbdefdevadr	4	Address of the device table for the default device for this process
pcbdefdevseqnum	2	The mount sequence number of the default device for this process
pcbdeffcbnum	4	FCB number for the current default directory
pcbdefseqnum	2	sequence number for the current default directory
pcbdefstrlen	2	Length of the default device string
pcbdefdiradr	4	Address of the default directory string
pcbdefdirlen	2	Length of the default directory string
pcbofpadr	4	List head to open files
pcbkpfdadr	4	List head to open KSAM files
pcbqueadr	4	Address of the pcb of next entry in whatever queue this process is waiting
pcbnetpcktnum	2	Network packet number
pcbtrapvecs	64	Trap handler addresses
pcbØdivide	4	Divide by zero trap handler address
pcbchktrap	4	Check trap handler address
pcbtrapv	4	Overflow trap handler address
pcbtracetrap	4	Trace trap handler address
pcbline1Ø1Ø	4	1Ø1Ø emulation trap handler address
pcbline1111	4	1111 emulation trap handler address
pcbdefexithand	4	Define exit tran handler
pcbfpinthand	4	Floating point interrupt handler
pcbtrapreserved	16	Reserved space for future trap handlers
pcbloaderaddr	4	Address of loader routine
pcbevntflque	4	Queue for event flag synchronization
pcbtrapreturn	4	Trap Ø return address
pcbtrapnum	2	The current trap number
pcbmailptr	4	Address of the head node for pending mail
pcbmailque	4	Queue for processes waiting for mail
pcbdefaultprot	2	The default protection mask
pcbaltuserid	2	The user ID number of the image file
pcbaltgroupid	2	The group ID number of the image file
pcbhibercnt	2	Count of how many times this process has been hibernated
pcbschedcnt	4	Count of how many times this process has been scheduled.
pcbnsmaddr	4	List head for named shared memory regions that are currently mapped into this process

pcbnetpageaddr	4	Holds network packet page address
pcbmldrlisthead	8	List head for control information by various WMCS loaders.
pcballochdr	4	List head for devices that are allocated to this process
pcborigprivilege	2	Holds original privileges process was created with before any installed privileges were added in.
pcbdefaultnode	4	Contains siteid of current default node
pcbcurtrapnum	4	The number of current SVCs being executed
pcbcurtrapprm	4	The stack address of current trap parameters
pcbremotepid	4	If this is an NSP, this is PID of originator
pcbremoteuic	4	If this is an NSP, this is UIC of originator
pcbremotepriv	2	If this is an NSP, this is priv of originator
pcbrctadr	4	List head for remote connection table
pcbbasepriority	2	Holds base priority level
pcbcurstate	4	Index into scheduling queues for current state

Queue Name	Offset	Description
pcbcst_toswapin	0	List for processes to be swapped in
pcbcst_active	4	List for active processes
pcbcst_asleep	8	All processes above here are in normal sleeps
pcbcst_iowait	8	List for processes in I/O wait
pcbcst_hibernate	12	List for processes in hibernation
pcbcst_childwait	16	List for processes in child wait
pcbcst_sqsize	20	Holds size of this scheduling queue

pcbswaptslice	2	Holds # of timeslices allocated to the process after it is swapped in, before it is eligible to be swapped out again
pcbremotetslice	2	If this is an NSP, timeslice of originator
pcbremoteattr	2	If this is an NSP, attributes of originator
pcbremoteprior	2	If this is an NSP, priority of originator
pcbnoswapcnt	2	If non-zero, process is swap critical
pcbpagecnt	2	Holds size of this pcb in pages
pcbreserved	16	Reserved space
pcbidfield	2	Table ID tag value
pcbidtag	$3333	Table ID value
pcbmemory	1024	The process's memory mapping registers
pcbdevstr	94	The default device/directory string

Related Privileges:

None.

Parameters:

pid	- Process ID of the process whose PCB is desired.
pcbuff	- Address of the buffer to receive the PCB
len	- The number of bytes requested. This number of bytes will be copied into the users buffer.
retlen	- Address of where to return the number of bytes actually copied into the users buffer.
status	- Address of a long word to receive the result of the operation.

Diagnostics:

errinsufpriv	(1)	The process lacks the privileges required to perform the operation.
errprcsnotfnd	(2)	The specified process is not in the system process table.

See Also:

_gencqy	- Get PID of ancestor process
_getpid	- Get process ID (PID) from name
_getpnam	- Get process name from PID
_prclst	- Get PIDs on a priority level

Assembler Calling Sequence:

```
push    pid              ;value - process id
push    pcbuff           ;address - buffer to receive pcb
push    len              ;value - length of buffer
push    retlen           ;address - # of bytes transferred
push    status           ;address - result of the operation
jsr     _getpcb          ;get process control block
```

C Function Declaration:

```
#include "sys$disk/sysincl.sys/pcbdisp.h"
                            /* get process control block */
long                        /* returns result of the operation */
_getpcb(pid, pcbuff, len, retlen)
        long pid;                   /* process id */
        pcbtable *pcbuff;           /* buffer to receive pcb */
        long len;                   /* length of buffer */
        long *retlen;               /* # of bytes transferred */
```

FORTRAN Subroutine Declaration:

```
c                                   ! get process control block
        subroutine _getpcb(pid, pcbuff, len, retlen, status)
            integer*4 pid       ! process id
            character*(*) pcbuff ! buffer to receive pcb
            integer*4 len       ! length of buffer
            integer*4 retlen    ! # of bytes transferred
            integer*4 status    ! result of the operation
```

Pascal Procedure Declaration:

```
%%sys$disk/sysincl.sys/pcbdisp.pas
procedure _getpcb(                  {** get process control block}
        pid     : longint;          {** process id}
        pcbuff  : ^array_of_char;   {** buffer to receive PCB}
        len     : longint;          {** length of buffer}
    var retlen  : longint;          {** # of bytes transferred}
    var status  : longint           {** result of the operation}
); external;
```

Hibernate a process.

Description:

Remove a process from consideration by the scheduler. This will increment a hibernate reference count and set the hibernate status bit so the process can no longer be scheduled. There are two ways to wake a hibernated process. A call to _wake will set the reference count to zero and clear the hibernate status bit. On the other hand a call to _wakec will decrement the hibernate count and clear the hibernate status bit when the count goes to zero. A hibernated process will exist indefinitely in the process table but in a dormant state until either the process is terminated by another process, or is awakened by another process.

Related Privileges:

none	- Allows process to hibernate any process with the same owner id and group id (uic) as the calling process.
group	- Allows process to hibernate any process with the same group id as the calling process.
world	- Allows process to hibernate any process in the system.

Parameters:

pid	- This is a long word uniquely specifying the process ID of the process to be hibernated. The high word refers to the site ID, with Ø representing the node you are on. The low word refers to the process: Ø refers to the calling process, -1 refers to the parent of the calling process, -2 is undefined, and -3 hibernates all processes except the calling process.
status	- Address of a long word to receive the result of the operation.

Diagnostics:

errinsufpriv	(1)	The process lacks the privileges required to perform the operation.
errprcsnotfnd	(2)	The specified process is not in the system process table.

See Also:

_wait	- Pause for a period of time
_wake	- Wake a hibernated process
_wakec	- Wake a hibernated process with count

Assembler Calling Sequence:

```
push    pid                 ;value - process id
push    status              ;address - result of the operation
jsr     _hibern             ;hibernate a process
```

C function declaration:

```
                            /* hibernate a process */
long                        /* returns result of the operation */
_hibern(pid)
        long pid;           /* process id */
```

Fortran Subroutine Declaration:

```
c                                   ! hibernate a process
        subroutine hibern(pid, status)
            integer*4 pid           ! process id
            integer*4 status        ! result of the operation
```

Pascal Procedure Declaration:

```
procedure _hibern(              {** hibernate a process}
        pid     : longint;      {** process id}
    var status  : longint       {** result of the operation}
); external;
```

Insert a CPIR record

Description:

Allows a process to insert a Create Process Indirection Record (CPIR) into the system list. If a record already exists of the given file type, the old record is deleted first.

This record consists of a file type and a string. If a user does a _CRPROC svc on a file that is not an image type file, then the WMCS scans the CPIR records for the given file type. If a CPIR record exists for it, then the given string is inserted in front of the current command line. A new image file name is located by scanning the new command line string for the first invalid filename character, all characters before it are assumed to be the new image name.

This operation is recursive in that the new image file will be located and if necessary the CPIR table will be scanned again.

Eventually the WMCS will be modified to have a different CPIR list per process and the user can access the SYSTEM CPIR list with the special id of -3 in the low word, thus the reason there is a PID parameter. But the per process list has not been implemented yet so the low word of the PID must be -3 for this to work. The high word of the PID is the siteid of the node to insert into. A value of zero will insert into the current node.

Related Privileges:

none	- The process cannot successfully insert a new file type into the CPIR list.
operator-	Allows the calling process to insert a new file type into the CPIR list.

_inscpir

Parameters:

pid	- The low word of the PID must be -3. The high word of the PID is the siteid of the node to insert into. A value of zero will do it to the current node.
fstr	- Contains the command line string to be inserted infront of the users command line.
filetype	- Contains the file type associated with the given command-line string.
status	- Address of a long word to receive the result of the operation.

Diagnostics:

errinsufpriv	(1)	The process lacks the privileges required to perform the operation.
errinvsiteid	(8)	The specified site id does not exist.
errnotimp	(45)	This item is not implemented yet.

See Also:

_delcpir - delete CPIR record.
_getcpir - Get CPIR records.

Assembler Calling Sequence:

```
    push    pid                         ;value - process id
    push    fstr                        ;address - command line string
    push    filetype                    ;value - file type
    push    status                      ;address - result of operation
    jsr     _inscpir                    ;insert CPIR record
```

C Function Declaration:

```
                                        /* insert CPIR record*/
    long                                /* returns result of operation*/
    _inscpir(pid, fstr, filetype)
            long pid;                   /* process id*/
            char fstr[94];              /* command-line string*/
            long filetype;              /* file type*/
```

Fortran Subroutine Declaration:

```
c                                   ! insert CPIR record
        subroutine inscpi(pid, fstr, filetype, status)
            integer*4 pid           ! process id
            character*94 fstr       ! command-line string
            integer*4 filetype      ! file type
            integer*4 status        ! result of operation
```

Pascal Procedure Declaration:

```
procedure _inscpir(                 {** insert CPIR record}
        pid     : longint;          {** process id}
        fstr    : string[93];       {** command line string}
        filetype: longint;          {** file type}
    var status  : longint           {** result of operation}
); external;
```

Mount a logical device.

Description:

This system call is used to announce the existence of a device to the system. The system then mounts the device by loading a driver and initializing the device. If the device driver is already present in memory, a new one is not loaded; rather, the current driver is shared.

For disk and tape class devices which are not mounted "special", the owner of the volume and the protection specification for each class of user is specified in the volume label.

For TTY, pipe and sync class devices, the owner of the device becomes the UIC of the process issuing the call to _mount. The protection mask for the device will be the default protection mask associated with the calling process.

For devices mounted "special", the owner of the device becomes the UIC of the process issuing the call to _mount. The protection mask for the device will be the default protection mask associated with the calling process.

The process must have read privilege to the device containing the device driver, execute privilege to all directories in the path to the device driver, read privilege to the directory containing the device driver and read privilege to the file containing the device driver.

If the process has operator privilege, it can mount a device using a device driver that is not installed. If the process does not have operator privilege, it can only mount devices using installed device drivers. In either case, the process must satisfy the following requirements.

If the driver is specified by fcb.seq number, the process must have read privilege to the device containing the driver and read privilege to the file containing the driver.

In addition, the process must have execute access to the device being mounted according to the owner and group ID (UIC) of the volume and its protection mask. Note that any process can mount a TTY, pipe or sync class device.

The process must have operator privilege in order to mount any device as "special" (diskspc, ttyspc, etc.).

Related Privileges:

none	- Allows mounting of device if the process has privileges as described above and the driver has been installed.
altuic	- Allows mounting of device if the owner of the image file of the current process has access to the file and device as described above.
bypass	- Allows mounting of device independent of the file protection.
operator	- Allows mounting of devices as 'special'. Also allows mounting devices with uninstalled drivers.
system	- Allows mounting of device if the system has access to the file and device as described above.

Parameters:

dname	- This parameter consists of two components, separated by commas. The first component is the devicename. The second component is the drive ID. Address of a null terminated string containing the name by which the device will be known. This string will be translated automatically by the WMCS to its logical equivalent. This string may contain up to 93 valid characters followed by a null.
driver	- Address of a null terminated string containing the name of the file which contains the device driver. If a driver with the same identifier (irrespective of file name) is found in the system, the driver is not loaded. This string will be translated automatically by the WMCS to its logical equivalent. This string may contain up to 93 valid characters followed by a null.

class	- The device class. Valid classes are:

Ø,1	- Character class device (TTYSpecial, TTY)
2,3	- Tape class device (TapeSpecial, Tape)
4,5	- Disk class device (DiskSpecial, Disk)
6,7	- Network class device (NetworkSpecial, Network)
8,9	- Pipe class device (PipeSpecial, Pipe)
1Ø,11	- Sync class device (SyncSpecial, Sync)
12,13	- Queue class device (QueueSpecial, Queue)
14,15	- Nondev class device (NondevSpecial, Nondev)

dstat - The address of a 128 byte buffer containing the initial device status to be assigned the device when it is mounted. If this parameter is zero the default device status is used.

This parameter has two purposes:

1) To provide an opportunity to set device characteristics that, unless set properly, would not allow the device to be mounted, e.g., the tape block size.
2) To set device characteristics that could otherwise not be changed once the device is mounted, e.g., disk cache size.

This parameter is not meant to be a substitute for _setdst. As such, not all of the values that can be specified with _setdst can be specified in this parameter.

The device status table is divided into two parts. The first half is device independent and is composed of the following fields:

Name	Length (bytes)	Settable	Description
dsclassid	2	No	The device class. Uses the class parameter to the _mount system call.
dsdriverid	2	No	Unique ID number for this device

			driver
dsblksz	2	Yes	block size of the device, e.g., sector size. For disks, the sector size is either 512 bytes or 1024 bytes, determined by the driver. Note that disk sector size cannot be changed. For tapes, the default is 1024 bytes. Specify zero to accept the default.
dsharderr	2	No	The hard error count for the device
dssofterr	2	No	The soft error count for the device
dsreadoper	4	No	Number of read operations on this device
dswriteoper	4	No	Number of write operations on this device
dsmaxnumdev	2	No	Maximum # of devices this driver can handle
dscurnumdev	2	No	Number of devices currently mounted using this device driver
dsnumtoretry	2	Yes	Number of times to retry before reporting a hard error. The default is determined by the driver. Specify zero (0) to accept the default.
dserrorreason	4	No	Hardware error code for the last error
dsreserved	30	No	Reserved
dsnexttableptr	4	No	Address of next device status table

The second half of the device status table is device class dependent. For TAPE class devices the second part is defined as follows:

Name	Length (bytes)	Settable	Description
dstpstatus	2	No	Tape device status
dstpflagsl	2	Yes	Tape status information. Bit encoded. If zero is specified, the default is used.

Bit name	bit #	Description
dstpdoraw	Ø	Ø=Read after write disabled 1=Read after write enabled
dstpreadahead	1	Ø=Read ahead enabled 1=Read ahead disabled
dstperrintenb	2	Ø=Error interrupts are enabled 1=Error interrupts are disabled

Name	Length (bytes)	Settable	Description
dstpspeed	1	Yes	Tape speed - Specify zero to use default

	Ø - Reserved
dstpspeed12ips	1 - 12 ips
dstpspeed25ips	2 - 25 ips
dstpspeed3Øips	3 - 3Ø ips
dstpspeed5Øips	4 - 5Ø ips
dstpspeed9Øips	5 - 9Ø ips
dstpspeed1ØØips	6 - 1ØØ ips
dstpspeed125ips	7 - 125 ips

Name	Length (bytes)	Settable	Description
dstpdensity	1	Yes	Tape density - Specify Ø to use default

	Ø - Reserved
dstpdens8ØØbpi	1 - 8ØØ bpi
dstpdens16ØØbpi	2 - 16ØØ bpi
dstpdens32ØØbpi	3 - 32ØØ bpi
dstpdens625Øbpi	4 - 625Ø bpi
dstpdens64ØØbpi	5 - 64ØØ bpi

Name	Length (bytes)	Settable	Description
dstpiopbcnt	2	Yes	Number of IOPBs allocated to the drive. The default is determined

			by the driver. Specify zero to use the default
dstpcachesz	2	Yes	# of sectors in disk cache. Default is determined by the value in the boot block. Specify Ø to use the default.
dstpreserved	46	No	Reserved
dstpuserfield	8	Yes	User defined status. The default is determined by the driver. Specify zero to use the default.

For DISK class devices the second half of the device status table is defined as follows:

Name	Length (bytes)	Settable	Description
dsdkintfac	2	No	Disk interleave factor
dsdkiopbcnt	2	Yes	Number of IOPB's allocated to the drive The default is determined by the driver Specify zero to use the default
dsdknumbsect	4	No	The number of sectors on the volume
dsdksectrack	2	No	The number of sectors on a track
dsdkheads	2	No	The number of heads on the device
dsdkcylinders	2	No	The number of cylinders on the volume
dsdkflagsl	2	No	Disk status information. Bit encoded word
dsdkcurcyl	2	No	Current cylinder position
dsdkcachesz	2	Yes	# of sectors in disk cache. Default is determined by the value in the boot block. Specify Ø to use the default.
dsdkentryname	16	No	The name of the drive type
dsdkreserved	2Ø	No	Reserved
dsdkuserfield	8	Yes	User defined status. The default is determined by the driver. Specify zero to use the default.

For TTY class devices the second half of the device status table is defined as follows:

Name	Length (bytes)	Settable	Description
dstymodereg1	1	No	Uart mode register 1
dstymodereg2	1	No	Uart mode register 2
dstycmdreg	1	No	Uart command register
dstytermtype	1	No	Terminal type definition
dstystatreg	1	No	Uart status register
dstypacketterm	1	No	Packet termination conditions
dstyflagsl	2	No	Terminal status information
dstyinputcnt	2	No	Characters in input interrupt buffer

dstyoutptcnt	2	No	Characters in output interrupt buffer
dstycolumnpos	2	No	Current column position
dstyinbufsz	2	Yes	Input buffer size in bytes. The default is determined by the driver. Specify zero to use the default.
dstyoutbufsz	2	Yes	Output buffer size in bytes. The default is determined by the driver. Specify zero to use the default.
dstywidth	2	No	Holds terminal width
dstylength	2	No	Holds terminal length
dstysubreadoper	4	No	Holds sub-read operations count
dstysubwriteoper	4	No	Holds sub-write operations count
dstyreserved	26	No	Reserved
dstyuserfield	8	Yes	User defined status. The default is determined by the driver. To use the default, specify zero.

For PIPE class devices the second half of the device status table is defined as follows:

Name	Length (bytes)	Settable	Description
dsppreaderpid	4	No	Process ID of pending reader
dsppwriterpid	4	No	Process ID of pending writer
dspppipeid	4	No	The pipe's ID
dsppbuffersz	2	No	The buffer size in bytes
dsppbuffercnt	2	No	Number of characters in the pipe buffer
dsppreadque	4	No	Address of read queue
dsppwriteque	4	No	Address of write queue
dsppreserved	32	No	Reserved
dsppuserfield	8	Yes	User defined status. The default is determined by the driver. Specify zero to use the default.

For SYNC class devices the second half of the device status table is defined as follows:

Name	Length (bytes)	Settable	Description
dssymodereg1	1	No	Mode register 1 of the uart
dssymodereg2	1	No	Mode register 2 of the uart
dssycmdreg	1	No	Command register of the uart
dssytermtype	1	No	Terminal type definition
dssystatreg	1	No	Status register of uart
dssynumbsync	1	No	Number of sync characters to write
dssyflags1	2	No	Device Status flags. Bit encoded.
dssyinputcnt	2	No	Number of characters in input interrupt buffer

dssyoutputcnt	2	No	Number of characters in output interrupt buffer
dssyinbufsz	2	Yes	Input buffer size in bytes. The default is determined by the driver. Specify zero to use the default.
dssyoutbufsz	2	Yes	Output buffer size in bytes. The default is determined by the driver. Specify zero to use the default.
dssyprevrderr	4	No	Error from previous unverified read
dssyprevwrerr	4	No	Error from previous no-wait write
dssyprevrdtype	1	No	Type of previous read
dssynumbtrpad	1	No	Number of trailing pads to write
dssyrecsize	2	No	Used in transparent mode with ITBs
dssyreserved	28	No	Reserved
dssyuserfield	8	Yes	User defined status. The default is determined by the driver. Specify zero to use the default.

For NETWORK class devices the second half of the device status table is defined as follows:

Name	Length (bytes)	Settable	Description
dsnkflags	2	No	Device status flags. Bit encoded.
dsnkwindowsize	1	No	Window size the circuit will use
dsnkreserved	53	No	Reserved
dsnkuserfield	8	No	User defined status

Bit Name	Bit #	Description
dsnkbyte	Ø	Ø=datagram mode 1=byte mode
dsnkmodemctrl	1	Ø=not enabled 1=modem ctrl enabled

For NONDEV class devices the second half of the device status table is defined as follows:

Name	Length (bytes)	Settable	Description
dsnduserfield	64	No	User defined status

For QUEUE class devices the second half of the device status table is defined as follows:

Name	Length (bytes)	Settable	Description
dsquassocdev	9	No	A null terminated string containing the name of the physical printer device
dsqusenddev	9	No	A null terminated string containing the name of the physical device that control messages are to be sent to
dsquformname	10	No	A null terminated string containing the current form name
dsqunumexec	2	No	This is the maximum number of entries that can execute concurrently
dsqucurnumexec	2	No	This is the number of entries that are currently active
dsquflags	2	Yes	Device Status flags. Bit encoded.

Bit Name	Bit #	Description
dsquflupdating	0	Updating current queue control file
dsquflqmstay	1	Queue manager process will remain running even when the queue is empty
dsquflnorestart	2	When the queue is mounted it does not restart jobs in queue

Name	Length (bytes)	Settable	Description
dsqulength	2	No	Holds the length of the forms of the printer associated with this queue
dsquwidth	2	No	Holds the width of the forms of the printer associated with this queue
dsqunextentry	4	No	Entry number of the next entry queued
dsqutype	1	Yes	The type of queue this is. Values are:

Value Name	Value	Description
dsqutpprint	1	Print type queue
dsqutpjob	2	Job entry queue

Name	Length (bytes)	Settable	Description
dsqubaseprior	1	No	Priority that entries will be queued at if they specify the default priority
dsqureserved	20	No	Reserved
dsquuserfield	8	No	User defined status

label	- Address of a 17-byte string to receive the device label. The returned string will be null terminated (up to 16 valid characters and a null).
status	- Address of a long word to receive the result of the operation.

Diagnostics:

errinsufpriv	(1)	The process lacks the privileges required to perform the operation.
errnomemavail	(7)	All available memory has been allocated.
errinvdevnam	(130)	The specified devicename is syntactically incorrect.
errundevnam	(131)	The WMCS does not recognize the devicename. Is the device mounted?
errfilnotfnd	(133)	The specified file could not be found.
errreadlock	(135)	The specified file is read-locked.
errreadleof	(140)	The process tried to read past the logical end of a file.
errnoexecpriv	(143)	The process does not have Execute Privilege for the file.
errnoreadpriv	(144)	The process does not have Read Privilege for the file.
errinvfnstr	(147)	The specified filename is syntactically incorrect.
errinvdirfle	(148)	The specified directory is not a directory type file.
errinvdirstr	(149)	The specified directory name is syntactically incorrect.
errimprdmnt	(164)	This device was improperly dismounted.
errinvcloper	(173)	The operation is inappropriate for the device class.
errdirnotfnd	(177)	The specified directory does not exist.
errdevnamexs	(179)	The specified device is already mounted.
errinvclass	(180)	The WMCS does not recognize the specified device class.
errnobbfound	(181)	The specified volume has no valid boot block.
errinvdmreq	(183)	The process requested more than 3964 bytes of dynamic memory.
errnoclass	(185)	The device class handler was not loaded when the system was booted.
errprevinit	(188)	The specified device is already mounted, and has another name.
errmntasync	(190)	The specified device has already been mounted for synchronous use.
errmntsync	(191)	The specified device has already been mounted for asynchronous use.
errinvdriver	(216)	The specified file does not contain a device driver.
errdevwrtprot	(269)	The specified device is write-protected.
errcantreaddsr	(308)	The size of the device driver does not match

```
                          its expected size.
errinvdrvnum    (311) A value in at least one field of the devicename
                          is disallowed.
errnohardware   (312) The PC board for the specified device is not
                          installed.
                          Device integrity errors
```

See Also:

```
_dismnt - Dismount a logical device
_flush  - Flush I/O buffers to the device
_getdnam- Get devicename
_getdst - Get device status
_giodst - Get device status with lun
_setdst - Set device status
_siodst - Set device status with lun
```

Assembler Calling Sequence:

```
%%sys$disk/sysincl.sys/dstatdisp.asm
push    dname               ;address - devicename
push    driver              ;address - driver file name
push    class               ;value - device class
push    dstat               ;address - initial device status
push    label               ;address - device label
push    status              ;address - result of the operation
jsr     _mount              ;mount a logical device
```

C Function Declaration:

```
#include "sys$disk/sysincl.sys/dstatdisp.h"
                            /* mount a logical device */
long                        /* returns result of the operation */
_mount(dname, driver, class, dstat, label)
        char dname[94];         /* devicename */
        char driver[94];        /* driver file name */
        long class;             /* device class */
        devicestatus *dstat;    /* initial device status */
        char label[17];         /* device label */
```

FORTRAN Subroutine Declaration:

```
c                              ! mount a logical device
      subroutine mount(dname, driver, class, dstat, label, status)
          character*94 dname    ! devicename
          character*94 driver   ! driver file name
          integer*4 class       ! device class
          character*(*) dstat   ! initial device status
          character*17 label    ! device label
          integer*4 status      ! result of the operation
```

Pascal Procedure Declaration:

Note - If passing a device status block, use the following expression: cast(vloc(devicestatus),longint)

```
%%sys$disk/sysincl.sys/dstatdisp.pas
procedure _mount(                    {** mount a logical device}
        dname   : string[93];        {** devicename}
        driver  : string[93];        {** driver file name}
        class   : longint;           {** device class}
        dstat   : longint;           {** initial device status}
    var vlabel  : string[16];        {** device label}
    var status  : longint            {** result of the operation}
); external;
```

Write file control block.

Description:

This SVC allows the calling process to update the file control block for an open file on any disk class device. Note that this requires that the calling process have writephys privileges and have write access to the file for most fields, however, you can change the following fields without any privileges: fcbfiletype, fcbrecordsz, fcblogicalsz, fcbfileid, and fcbprotect. For security reasons the file should have been opened with write locked access.

CAUTION: The FCB file is the heart of the file system. Careless tampering with the FCB file can cause severe damage to the file system's integrity.

NOTE: The format of the file control block may change with each release. The current definition is included in each release in the file /SYSINCL.SYS/FCBDISP.*. The name of the FCB record is "fcbtype," i.e., in your program you can declare a variable whose type is "fcbtype."

There are several variations on the format of file control blocks, depending on the class of device which contains the file. Disk files contain "primary" FCBs and "continuation" FCBs. Tape files have "tape" FCBs. All other files have "tty" FCBs. You can only set the FCB for disk class devices.

The format of the first 14 bytes of the FCB record is the same for all types of FCBs. The fomat of this common type is:

Name	Length (bytes)	Description
fcbnum	4	FCB number for this FCB. The record number of this record within the FCB file. For tty FCBs, the value of this field is zero. This field may not be changed.
fcbseqnum	2	FCB sequence number. This number is unique for each usage of this FCB. For tty FCBs,. the value of this field is zero. This field may not be changed.
fcbcontfcbnum	4	FCB number of continuation FCB. The record number of the next FCB for this same file. For tape and tty FCBs, the value of this field is zero. This field may be zeroed (remove a continuation) but no other values may be set (add a continuation).
fcbcontfcbseq	2	Sequence number of the continuation FCB. For tape and tty FCBs, the value of this field is zero. This field may be zeroed (remove a continuation) but no other values may be set (add a continuation).
fcbusageid	1	Usage ID field. The type of FCB. Values are: fcbunalloc Ø This FCB is unused. The data in this record is invalid. fcballocroot 1 This record contains a root FCB. fcballoccont 2 This record contains a continuation FCB.
fcbextusecnt	1	Number of extent fields in use within this FCB.

The format of the last 242 bytes of the FCB is different for "primary" FCBs as opposed to "continuation" FCBs. For primary FCBs (disk, tape and tty) the format is as follows:

fcbfiletype	2	File type. This can be changed without privileges. For tty files, it is always set to zero (a data file). Valid file types are:

File Type	Value	Description
fcbftdata	0	data file
fcbftdir	1	directory file
fcbftimage	2	image file
fcbftksamdata	3	KSAM data file
fcbftksamkey	4	KSAM key file
fcbftllimage	5	LL image file
fcbftarchcont	6	archive file continuation
fcbftencrypt	7	encrypted file
fcbftsystem	8	system file
fcbftarchive	9	archive file
	20-255	reserved
	256-65535	user-defined file types

fcbfilename	9	Filename. For disk and tape files it contains the filename portion of the file designation. For tty files it contains the devicename.
fcbfileext	3	File extension. For tty FCBs this field is set to zero.
fcbfilevers	2	File version number. For tty FCBs this field is set to zero.
fcbdirfcbnum	4	Directory FCB number. The FCB number of the directory file containing this file. For tape and tty FCBs it contains zero.
fcbdirseqnum	2	Directory sequence number. The sequence number of the directory file containing this file. For tty FCBs this field contains zero.
fcbrecordsz	2	Default record size. This can be changed without privileges. For tty FCBs this field is set to 1.
fcbuserid	2	Owner ID of the file's owner.
fcbgroupid	2	Group ID of the file's owner.
fcbprotect	2	File protection field. This can be changed without privileges. For tty FCBs it contains the device protection.
fcbcreatemstim	4	The most significant 32 bits of the file creation date in system time format (year and day). For tty FCBs, it contains the year and day that the device was mounted.
fcbcreatelstim	4	The least significant 32 bits of the file creation date in system time format (hour, minute, ...). For tty FCBs, it contains the hour, minute, ... that the device was mounted.

fcbmodmstim	4	The most significant 32 bits of the date the file was last modified (year and day). For tty FCBs, it contains the year and day that the device was mounted.
fcbmodlstim	4	The least significant 32 bits of the date the file was last modified (hour, minute, second, tick). For tty FCBs, it contains the hour, minute, ... that the device was mounted.
fcbreserved	4	Reserved space.
fcbphysicalsz	4	The physical size of the file in bytes. For tty FCBs this field is set to zero.
fcblogicalsz	4	The logical size of the file in bytes. This can be changed without privileges. For tty FCBs this field is set to zero.
fcbfileid	2	File ID of the file. This can be changed without privileges. For tty FCBs this field is set to zero.
fcbrootextblk	180	File extent fields. There are 30 extent fields in a primary FCB. Each extent field is composed of 6 bytes. The first two bytes represent the number of sectors in that extent. The last four bytes are the logical sector number of the first sector in that extent.
fcbnotcksum	2	The FCB's NOTted checksum.

The format of the last 242 bytes of the FCB for "continuation" FCBs (disk only) is as follows:

fcbcontextblk	240	File extent fields in a continuation FCB. There are 40 extent fields in a continuation FCB. Each extent field is composed of 6 bytes. The first two bytes represent the number of sectors in that extent. The last four bytes are the logical sector number of the first sector in that extent.
fcbnotcksum	2	The FCB's NOTted checksum.

Related Privileges:

none	- Cannot write the FCB
writephys	- Allows the process to update the FCB if the process also has write access to the file.

Parameters:

lun	- Logical unit number of the file whose FCB is being updated.
cont	- Which part of the FCB for this file is to be updated. 0=root FCB, 1=first continuation FCB, etc.
fcbuff	- Address of a 256-byte buffer containing the FCB to be written. This buffer must be word aligned.
status	- Address of a long word to receive the result of the operation.

Diagnostics:

errinsufpriv	(1)	The process lacks the privileges required to perform the operation.
erridxrange	(56)	The table ends before the specified occurrence.
errinvlfn	(132)	The logical unit number does not correspond to an open file.
errnowriteacc	(142)	The process does not have write-access to the specified file.

See Also:

_create	- Create a file
_getfcb	- Get file control block
_open	- Open a file
_setfprt	- Set file protection

Assembler Calling Sequence:

```
push    lun             ;value - logical unit number
push    cont            ;value - continuation FCB number
push    fcbuff          ;address - buffer containing FCB
push    status          ;address - result of the operation
jsr     _setfcb         ;write file control block
```

C Function Declaration:

```
#include "sys$disk/sysincl.sys/fcbdisp.h"
                         /* write file control block */
long                     /* returns result of the operation */
_setfcb(lun, cont, fcbuff)
         long lun;              /* logical unit number */
         long cont;             /* continuation FCB number */
         fcbtype fcbuff;        /* buffer containing FCB */
```

FORTRAN Subroutine Declaration:

```
c                                 ! write file control block

          subroutine setfcb(lun, cont, fcbuff, status)
              integer*4 lun       ! logical unit number
              integer*4 cont      ! continuation FCB number
              character*(*) fcbuff ! buffer containing FCB
              integer*4 status    ! result of the operation
```

Pascal Procedure Declaration:

```
%%sys$disk/sysincl.sys/fcbdisp.pas
procedure _setfcb(                  {** write file control block}
        lun      : longint;         {** logical unit number}
        cont     : longint;         {** continuation FCB number}
        fcbuff   : ^array_of_char; {buffer containing FCB}
    var status   : longint          {** result of the operation}
); external;
```

_wake

Wake a hibernated process.

Description:

Zeros the hibernate count and clears the hibernate status bit in the process control block of the specified process. In other words the process will be awakened no matter how many times it has been hibernated. No error occurs if the process being awakened is not hibernating. Note that a process cannot wake itself since a hibernating process cannot make the call.

Related Privileges:

none	- Allows waking any process with the same owner id and group id as the calling process.
group	- Allows waking any process with the same group id as the calling process.
world	- Allows waking any process.

Parameters:

pid	- This is a long word uniquely specifying the process id of the process to wake up. The high word refers to the site ID, with Ø representing the node you are on. The low word refers to the process: a process id of -1 refers to the parent of the calling process, -2 is undefined, and -3 wakes all processes.
status	- Address of a long word to receive the result of the operation.

Diagnostics:

errinsufpriv	(1)	The process lacks the privileges required to perform the operation.
errprcsnotfnd	(2)	The specified process is not in the system process table.

See Also:

```
_hibern - Hibernate a process
_wakec  - Wake a hibernated process with count
```

Assembler Calling Sequence:

```
push    pid                 ;value - process id
push    status              ;address - result of the operation
jsr     _wake               ;wake a hibernated process
```

C function declaration:

```
                            /* wake a hibernated process */
long                        /* returns result of the operation */
_wake (pid)
        long pid;                   /* process id */
```

Fortran Subroutine Declaration:

```
c                           ! wake a hibernated process
        subroutine wake(pid, status)
            integer*4 pid           ! process id
            integer*4 status        ! result of the operation
```

Pascal Procedure Declaration:

```
procedure _wake(              {** wake a hibernated process}
        pid     : longint;          {** process id}
    var status  : longint           {** result of the operation}
); external;
```

WICAT Systems, Inc.

Product-documentation Comment Form

We are constantly improving our documentation, and we welcome specific comments on this manual.

Document Title: ____________________

Part Number: ____________________

Your Position:

☐ Novice user	☐ System manager
☐ Experienced user	☐ Systems analyst
☐ Applications programmer	☐ Hardware technician

Questions and Comments **Page No.**

Briefly describe examples, illustrations, or information that you think should be added to this manual.

____________________ ________

____________________ ________

____________________ ________

What would you delete from the manual and why?

____________________ ________

____________________ ________

____________________ ________

What areas need greater emphasis?

____________________ ________

____________________ ________

____________________ ________

List any terms or symbols used incorrectly.

____________________ ________

____________________ ________

____________________ ________

178-001-009 B

First Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 00028 OREM, UTAH

POSTAGE WILL BE PAID BE ADDRESSEE

WICAT Systems, Inc.
Attn: Corporate Communications
1875 S. State St.
Orem, UT 84058

Second Fold

Tape